RTVision Basic Programming Manual

V1.3.0

Contents

Chapter I ZVision Basic Quick Start

1.1.Linux Motion Controller

1.1.1.Motion Control Products Introduction

The motion controller based on the Linux operating system can manage and run programs in a systematic way, also can optimize resource calls and organize work processes reasonably. Compared with the "bare machine" (without software configuration), the Linux motion controller supports rich functions for users and provides a wealth of open interfaces for users. Namely, systematic management, multi-task operation, batch jobs, pipeline processes and multi-functional services make the Linux motion controller have huge advantages that "bare machine" does not have in terms of efficiency, real-time response and human-computer interaction.

1.1.2.Products Advantages

The Linux motion controller can divide an application program into multiple tasks during program execution, each task completes a part of the work, and each task can be written as an infinite loop. According to the priority of the task, the operating system executes each task in a time-sharing manner by the CPU to ensure that each task can be run, which can make each task execute in parallel, then the idle time of the CPU can be reduced and the utilization rate of the CPU can be promoted.

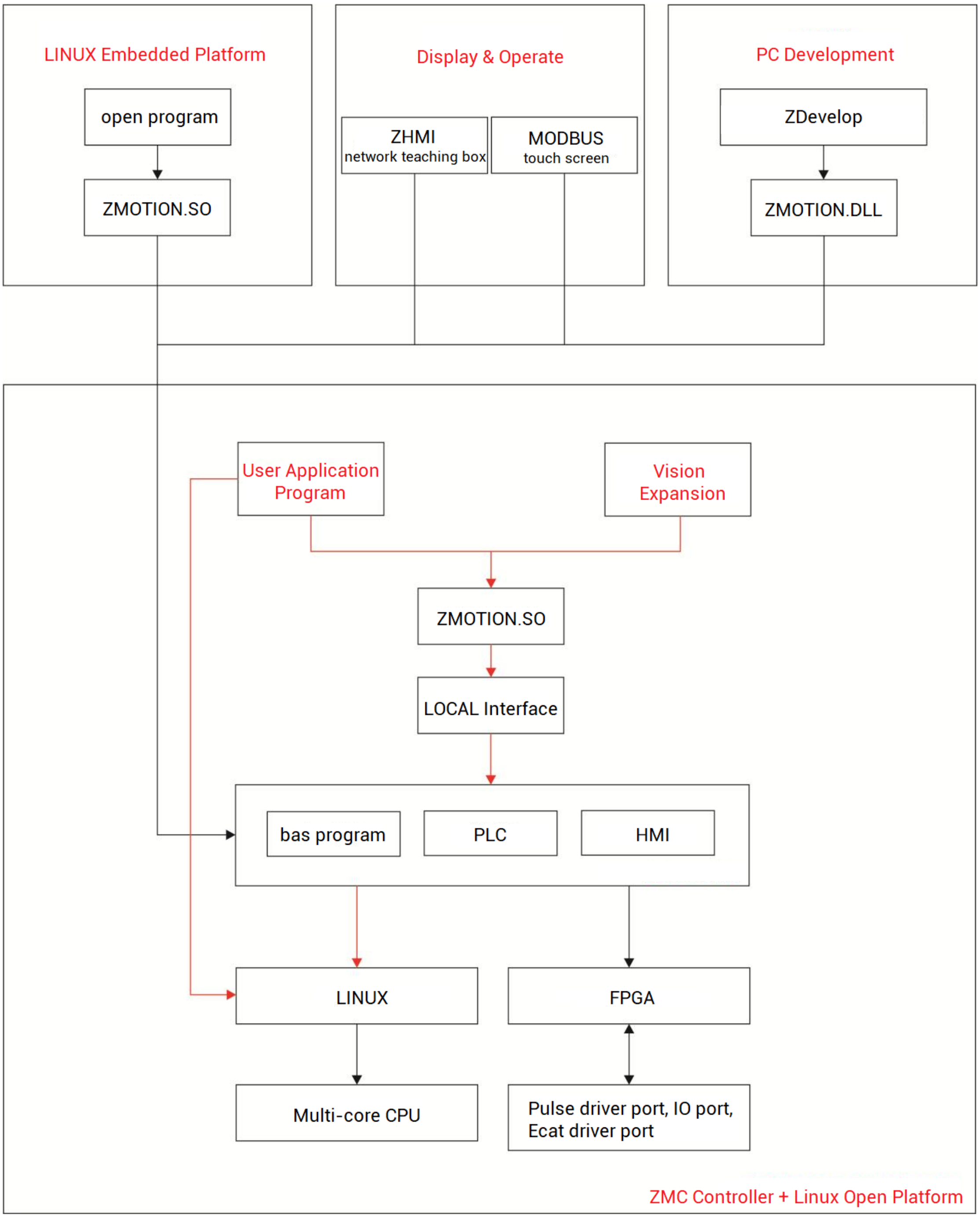
Zmotion provides powerful ZDevelop (RTSys) software development environment for motion control products, it is easy to use.

Followings are advantages of Linux controller:

- It can be developed by several kinds of software that are built in PC.
- The code versatility and portability of the motion control software are good.
- There are many engineers who can carry out development work, then development can be carried out without much training.

- Software under the Linux platform can be added freely to the controller.
- The controller can connect to the camera directly through Ethernet, and it supports the secondary development of various machine vision applications.
- Support motion and vision mixed programming.
- Low power consumption and high utilization.
- Excellent management for task scheduling.
- It not only supports ZBasic, ZPLC, ZHMI programming, but also supports Linux platform open programming (c, c++, java, python.).

1.2. Development Framework



1.3.Data Type

1.3.1.ZVOBJECT Type

The visual custom object type is defined by the ZVOBJECT keyword, and the related data is managed by Basic. The specific type can be obtained through ZV_TYPE. And following form shows types:

Type	Description
0	Undefined
1	Image
2	Rectangle
3	Region
4	Contour
5	List
6	Calibration parameters
7	Measurer
8	NCC template
9	Shape template
10	Color model

ZVOBJECT defines a variable that only declares the variable identifier. If the variable is not used, there is no type. When a variable is used in an instruction, if it is an output variable, no need to consider the type. For variables of different types, the original variable will be released automatically. For the same type, the original data will be released, and the corresponding initialization will be performed to receive the output result. If it is an input variable, then the specific type needs to adapt to the requirements of the command. For example, if the command receives a variable of image type, the input variable must also be of image type, otherwise an error will be reported.

If the same variable is used as input and output variables at the same time, the input and output variables should be of the same type, otherwise the data of the input variable will be cleared due to the automatic release of the aforementioned variables, and an error will also be reported in this case. The case where the input and output are of the same type is supported to facilitate the use of variables.

Notes:

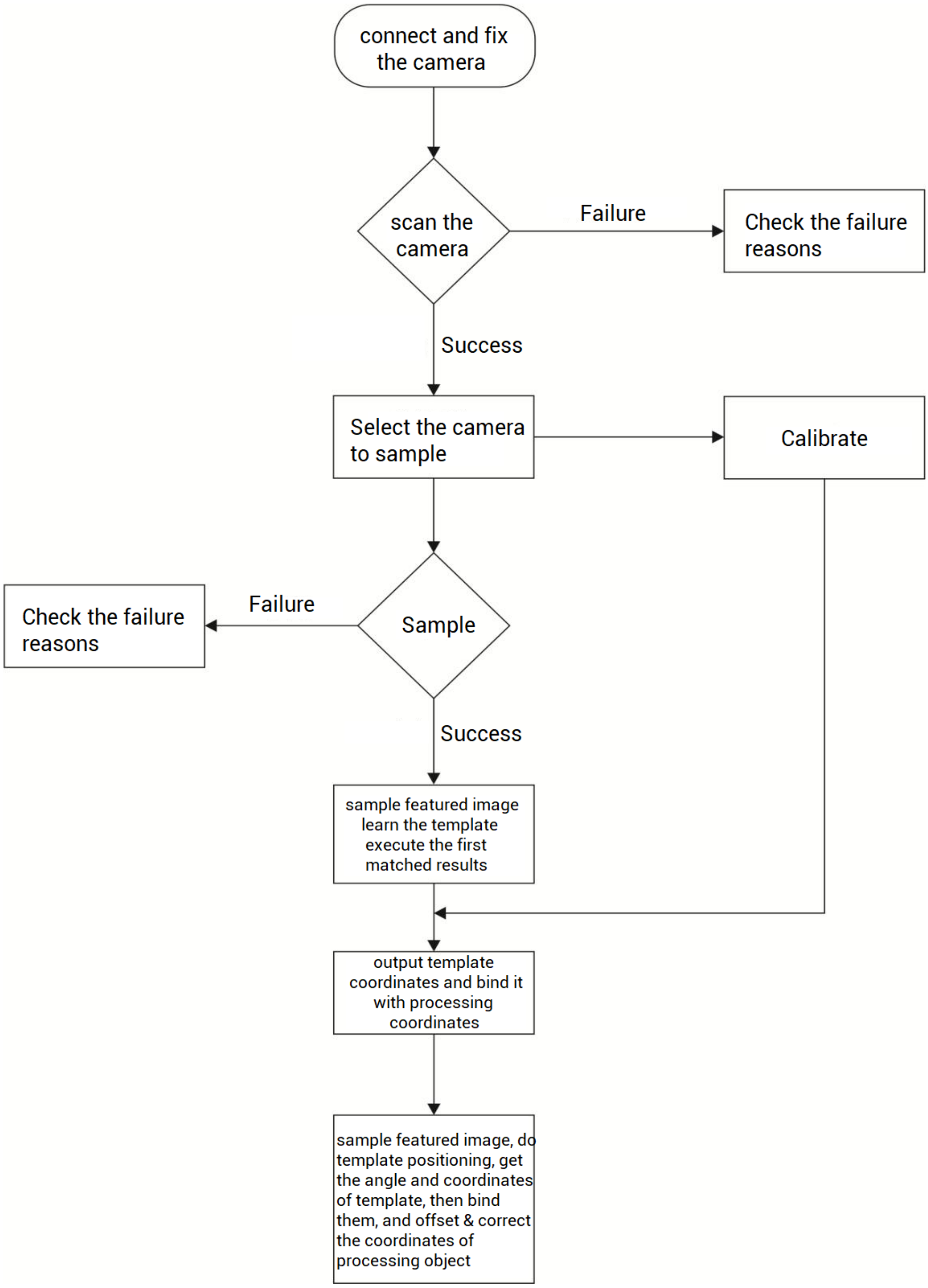
- When different types of variables are used as instruction parameters, variables' names can't be the same, otherwise, data will be cleared.
- ZVOBJECT variable doesn't support thread safety, please don't operate the same variable for multi-task.

1.3.2.ZVOBJECT General Operations

Type	Instruction
Get the type	ZV_TYPE
Whether is blank or not	ZV_ISEMPTY
Copy the data	ZV_COPY
clear the data	ZV_CLEAR

1.4. Vision Positioning

1.4.1. Vision Positioning Process



1.4.2. Calibration Method

1. Calibrate through calibration plate

Put the calibration board on the same plane as the measured object on site, and obtain the pixel coordinates of n feature points in the calibration board image by shooting the calibration board with the camera. If the world coordinates of each feature point are known, then input the world coordinates of n feature points accordingly. The way to obtain the world coordinates of feature points can also use the machine to control probe for the center of corresponding feature points, so as to obtain the world coordinates. Then use these n pairs of pixel coordinates and world coordinates to calibrate the camera, and calibrate the conversion relationship between the camera coordinate system and the world coordinate system.

2. Calibrate through 9-point

Use the method of locating feature points to obtain the pixel coordinates of image feature points. The methods that can be used for locating feature points include shape matching, Blob positioning and circle positioning.

First, ensure that the target does not move. The machine controls the camera to move nine times in the form of a nine-square grid and take pictures to collect nine images to obtain the pixel coordinates of nine feature points. Move once to take a picture and locate once. And take a picture once, make sure that the target is within the camera's field of view, at the same time read out the world coordinates of the machine. Then use these nine pairs of pixel coordinates and world coordinates to calibrate the camera, and calibrate the conversion relationship between the pixel coordinate system and the world coordinate system.

For more details, please refer to vision positioning routines.

1.5. Commonly Used Commands

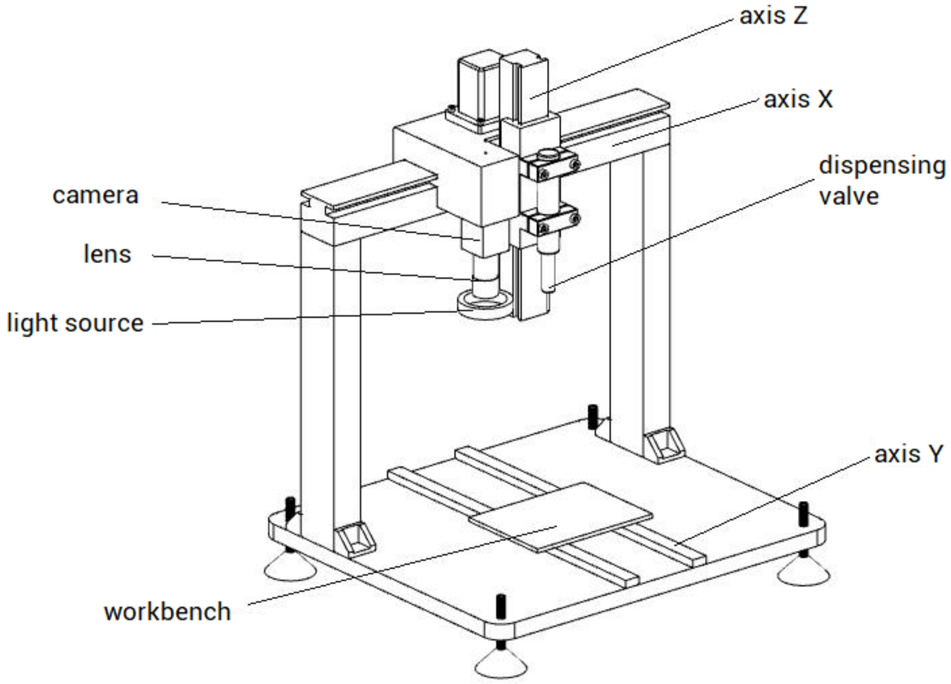
Type	Function	Instruction
Image	Read the image	ZV_READIMAGE
	Show the image	ZV_LATCH
	Save the image	ZV_WRITEIMAGE

	Image information	ZV_LATCHINFO
	Clear the image	ZV_LATCHCLEAR
Image operation	Affine transformation	ZV_AFFINE
	Scale factor scaling	ZV_ZOOM
	Target size scaling	ZV_RESIZE
	Mirror	ZV_MIRROR
	Rotate	ZV_ROTATE
Camera	Camera information	CAM_GETINFO
	Scan the camera	CAM_SCAN
	Select the camera	CAM_SEL
	Sample the camera	CAM_GRAB
	The number of cameras	CAM_COUNT
	Camera exposure	CAM_SETEXPOSURE
	Get the exposure	CAM_GETEXPOSURE
Template	Create the template	ZV_SHAPECREATERE
	Read the template	ZV_READSHAPEMOD
	Match the template	ZV_SHAPEFINO
	Save the template	ZV_WRITESHAPEMOD
	Template parameters	ZV_SHAPEDEFPARAM
Coordinates	From HMI into image	ZV_POSTOTIMG
	From image into HMI	ZV_POSFROMIMG
Measurement	Rectangle measurement region	ZV_MREGNRECT
	Rotation measurement region	ZV_MREGNRECT2
	Ring measurement region	ZV_MRGENARC
	Circle measurement	ZV_MRCIRCLE
Draw	Color	ZV_COLOR
	Mark the “Mark” point	ZV_MARKER
	Circle	ZV_CLRCLE
	Mask image	ZV_MASK
	Text	ZV_TEXT
	Rectangle	ZV_RECT
	Region	ZV_REGION
Matrix	Get the matrix information	ZV_MATINFO
	Create the matrix data	ZV_MATGENDATA

	Matrix corrosion	ZV_ERODE
	Matrix expansion	ZV_DILATE
	Open operation	ZV_OPENING
	Close operation	ZV_CLOSTING
Region	Generate matrix region	ZV_REGENRECT
	Generate circle region	ZV_REGENCIRCLE
	Generate ring region	ZV_REGENANNULAR
Region operation	Intersection	ZV_REITSEC
	Union	ZV_REUNION
	Difference set	ZV_REDIFF
Transformation	Affine transformation	ZV_AFFINETRANS
	Rigid transformation	ZV_GETRIGIDVECTOR
Filter	Gaussian filter	ZV_GAUSSBLUR
	Median filter	ZV_MEDIANBLUR
	Mean filtering	ZV_MEANBLUR
	Canny	ZV_CANNY
Color transformation	From RGB to gray image	ZV_RGBTOGRAY
	From gray to RGB	ZV_GRAYTORGB
	From other colors to RGB	ZV_COLORTORGB
	From RGB to other colors	ZV_RGBTOCOLOR
Calibration	The solid circle calibration plate extracts the pixel coordinates of the mark point	ZV_CALGETSCAPTS
	Calibration	ZV_CALCAM
	Calibrate the error	ZV_CALERROR

1.6. Applications

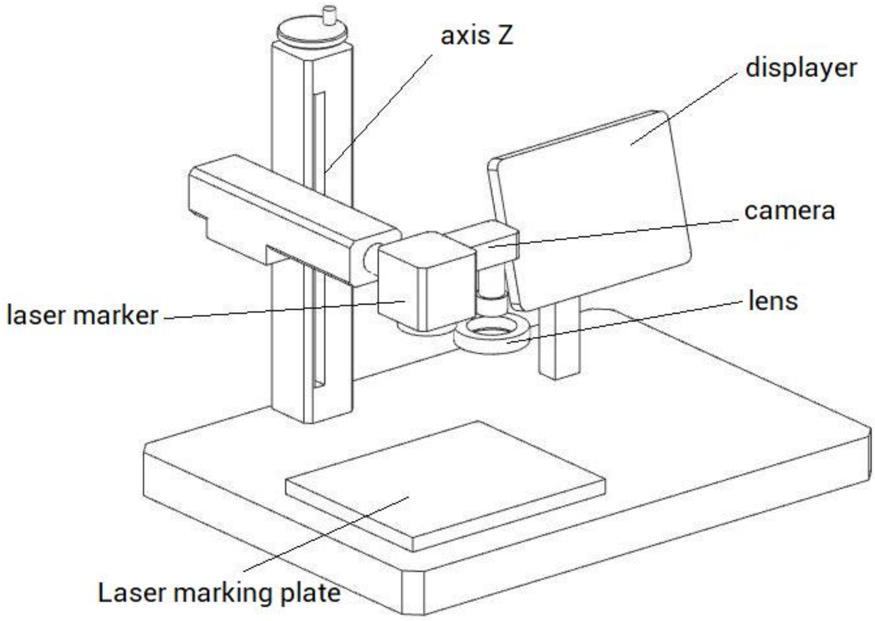
1.6.1. Dispensing



The visual dispensing machine is a kind of automatic dispensing machine. It can work cyclically, that is, long working hours and automatic operation of high precision glue-out, which saves manpower and improves the production efficiency of the enterprise. First, the compressed air is sent into the glue bottle, and then the glue is pressed into the feeding tube connected to the piston chamber so that the glue is pressed out of the needle mouth. The entire operation steps are controlled in the software, and the entire workflow is automated.

For the manual dispensing machine, it basically controls the entire working process manually. First, the glue is output from the pressure tank into the syringe, and then the controller is used to control the flow rate. Finally, the glue is applied by holding the syringe.

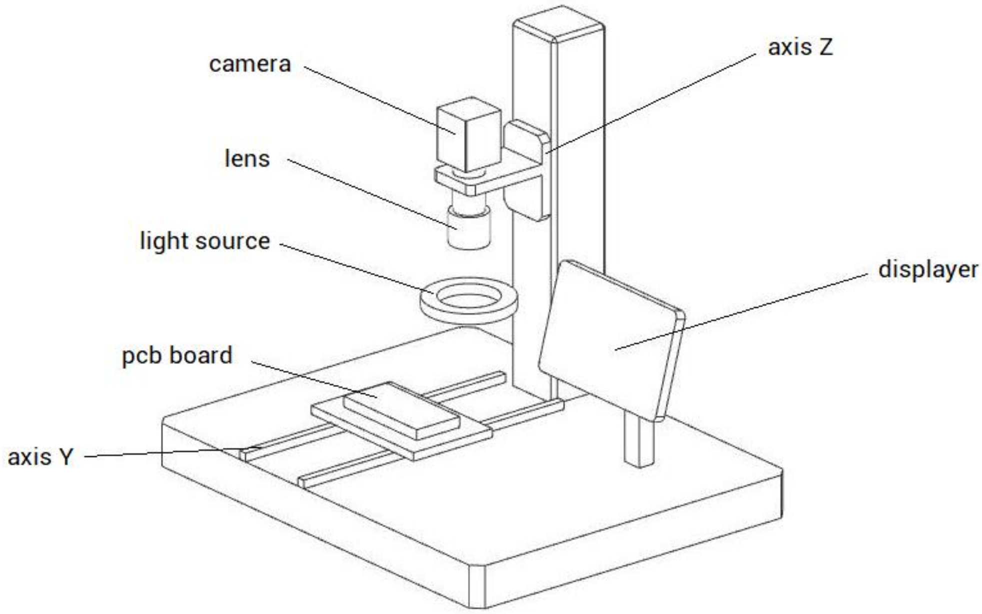
1.6.2.Laser Marking



Laser marking is a non-contact precision machining method that can be laser etched on the surface of any irregular workpiece without causing deformation of the workpiece due to internal stress caused by clamping, extrusion or impact. Then, high-precision and high-quality processing quality can be achieved.

Through visual inspection technology, assist laser marking operations, which frees laser marking from the limitations of fixtures and reduces processing costs while improving system applicability. In addition, it can achieve high-precision positioning with the help of vision detection, and positioning technology based on image processing can realize micron-level accuracy. Another obvious benefit is that the automation process and stability of the product line have greatly improved efficiency due to the reduction of manual involvement.

1.6.3.PCB Board Detection



PCB board detection originates from the traditional manual visual judgment method, instrument online detection method and functional testing method, machine vision detection technology is with more obvious advantages in terms of efficiency, labor cost, stability and accuracy. And there is a natural advantage in data collection for machine vision detection. While the amount of visual data is increasing, the efficiency of machine inspection technology can be further improved, which is what other inspection technologies can not possess.

Visual inspection can quickly detect defects in PCB manufacturing accurately, non-contact and highly flexible. Then scan the PCB board through the camera to obtain the image of the solder joints on the PCB board, extract the characteristic solder joints, and compare them with the information in the database. And with the help of embedded ARM, DSP, FPGA, etc. to carry out high-speed operations, so as to quickly and efficiently detect and classify welding defects, improve efficiency and greatly reduce time costs.

1.7. Common Problems

1.7.1. No Camera Scanned

- Check whether the wiring of the camera is loose, and whether the network LED of the camera is normal.
- Check whether the IP of the camera is occupied.
- Check whether the camera IP is in the same network segment.
- Check whether the camera is the type supported by the controller.
- Check the parameters that are to scan the camera whether are correct.

1.7.2. Blurry Image

- Set the display size (ZV_LATCHSETSIZE) corresponding to the latch channel corresponding to the image display. If the setting is not correct, the image may be blurred. The latch size setting is used to zoom in and out of the image, and an appropriate size needs to be set.
- Manually adjust the camera lens and adjust the focus.
- Set the appropriate exposure time in the camera.
- Manually adjust the aperture of the camera and adjust the brightness.
- Adjust the light source of environment to achieve a suitable brightness.

1.7.3. Camera Network

- When the camera is with higher pixel, it is necessary to ensure the network and use a 1000M USB switch, otherwise it will cause the camera parameter to fail to write, resulting in abnormal image acquisition.
- The camera network cable is directly connected to the network port of the controller, and the IP of the camera should be consistent with the network segment of the

network port of the controller

1.7.4. Abnormal Homing

- Check whether the origin sensor detects a signal.
- Check whether the indicator light of the sensor is ON when returning to the origin.
- Adjust the position so that it can sense the signal and check whether the sensor wire has fallen off.

1.7.5. Motor Doesn't Move

- **Reason of Driver:**

The factory settings of the drive generally do not reverse the IO level, which will cause the drive limit alarm. And the limit level should be set according to the drive manual. For example, Panasonic servo needs to set the parameters of pr4.01 and pr4.02 to 010101h (65793) and 020202h (131586) respectively. For other brands of drivers, please operate according to the relevant driver manual.

Corresponding parameter	Factory setting values (decimal system)	Position control / full closed loop control	
		Signal name	Logic
Pr4.00	00323232h (3289650)	SI-MON5	Commonly-ON (ON)
Pr4.01	00818181h (8487297)	POT	Commonly-OFF (NC)
Pr4.02	00828282h (8553090)	NOT	Commonly-OFF (NC)

Signal Name	Mark	Set values	
		Commonly-ON (ON)	Commonly-OFF (NC)
Invalid	-	00h	-
Prohibit driver inputting positively	POT	01h	81h
Prohibit driver inputting negatively	NOT	02h	82h

- **Reason of Program:**

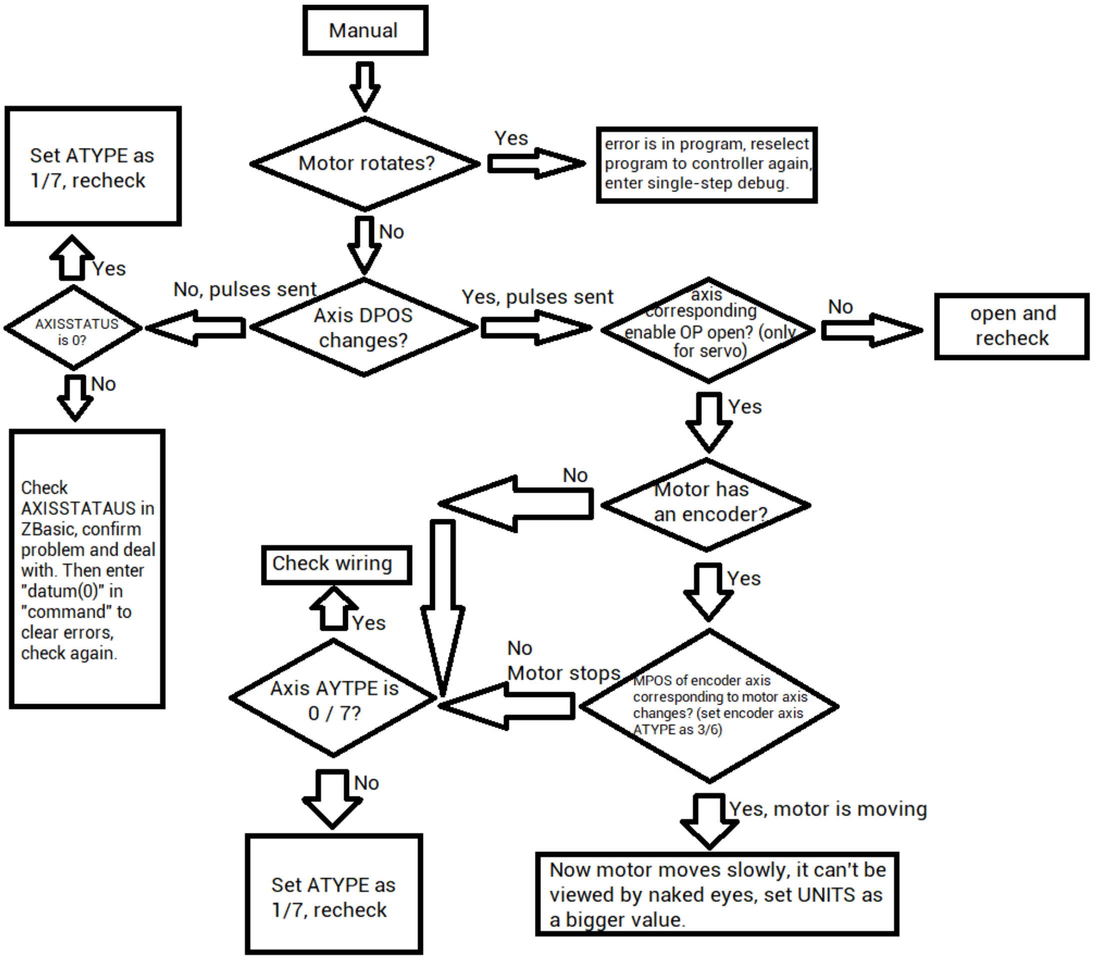
- If the UNITS setting is too small, the motor moves very slowly, which cannot be distinguished by the naked eye.
- The motor is in an abnormal state (limit, alarm...), unable to move, judge AXISSTATUS.
- The wiring of the motor is wrong, and the pulse cannot be transmitted correctly.
- The axis OP port is not enabled (only the servo motor needs to be open).
- The program processing prevents the motor from moving, download the empty program for confirmation.
- The driver alarms.

Below reasons mainly for bus axes:

- Fail to open bus scanning, print return values to confirm.
- wdog switch enable doesn't open, through axis_enable command.
- Wrong drive status setting, please refer to driver manual.

Problem checking steps:

- Open ZDevelop software to check problems.
- Close other software or programs that are connected to controller, except ZDevelop, to avoid external influences.
- Use ZDevelop to download one empty program into controller to avoid internal influences.
- Open ZDevelop software, click "VIEW" – "Manual" and "VIEW" – "Axis Parameters" for viewing and operating.
- If it is pulse axis, check according to below steps.



1.7.6. Motor Only Moves in One Single Direction

- The motor is in the limit state, check AXISSTATUS to confirm.
- The motor control mode is wrong, set INVERT_STEP to the corresponding mode (double pulse or pulse + direction).
- There is a problem with the motor wiring, check the wiring.

Chapter II Environment

2.1.Environment Description

2.1.1.Basic Limit

Name	Max Limit	Unit
Channel Numbers	4	
Max Dimension	2	Space
Image Size	8192	Pixel
Camera Numbers	4	
System Parameter Name Length	31/15	English Characters / Chines Characters
Camera Parameter Name Length	63/31	
File Path Length	255/127	

2.1.2.Image Data Type

0	8-bit without symbol 8U
1	16-bit without symbol 16U
2	32-bit with symbol 32S
3	64-bit with symbol 64F
4	32-bit with symbol 64F

The image is the single channel by default, if there are multiple channels, it will be specified. Multiple data types are mainly for efficiency, precision, application environment, etc., 8U is the data type of the usual camera, only a small number of cameras that support high dynamic range can support the data type of 16U. For 32S, 64F and 32F, they are based on processing and storage for the intermediate results of image operations, but they are valid in few cases.

2.1.3.ZVOBJECT Type

Type Value	Type	Description
0	Undefined	The variable that is new defined, no type information
1	Image	Multiple types of two-dimensional array structure, support multi-channel. Ordinary image, intermediate operation result, binary/difference/integral image, frequency domain image, etc.
2	Matrix	Two-dimensional array 64F type, only single channel is supported. General matrix, various transformation matrices, point sets, etc., distinguish single columns from images based on computing efficiency.
3	Region	Form a coded area, it is similar to a binary image, indicating the mask area of the image operation or the BLOB information of the image.
4	Contour	One-dimensional point set, it saves contour (closed) or edge data, and it can cache various feature parameters of contour to improve the efficiency of some applications.
5	List	One-dimensional indefinite type structure, an array of various ZVOBJECT types, such as outline list, area list, etc.
6	Calibration parameter	Dedicated data structure, result of linear or nonlinear calibration, result of distortion calibration or result of comprehensive calibration.
7	Measurer	Dedicated data structures for measurements of many types of geometric parameters.
8	NCC template	Dedicated data structure, data of NCC matching template. It can read template image, region, parameters.
9	Shape template	Dedicated data structure, data of shape matching template. It can read template image, region, contour, parameters.

2.2.Initialization

2.2.1.ZV_ENVINIT – Initialization of Running Environment

Type	Initialization
Description	It is used to initialize ZVision running environment, all ZVOBJECT variables will be cleared. And in some special situation, it is only called when Basic environment needs to be initialized again.
Grammar	ZV_ENBINIT()
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_ENBINIT() 'initialize ZVision running environment

2.3.Performance Mode

2.3.1.SYSTEM_ZVTASKS – Controller Task Mode

Type	Performance mode
Description	It is used to set the number of tasks of visual module, the default value is 1. If 2 or 3 are set, ZV commands for multi-task will be accelerated. For large images, it is recommended to set a bigger value, otherwise, Led may shrink or be incomplete. Like, for image above 2 million, 2 is set best, for image above 5 million, 3 is set best.
Grammar	SYSTEM_ZVTASKS = task task: task is the number of tasks for visual module
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	SYSTEM_ZVTASKS = 2 'set vision module tasks as 2

2.4.General Operations

2.4.1.ZV_OBJTYPE – Get the Type

Type	ZVOBJECT general operation
Description	It is used to get the type of ZVOBJECT variables. Please refer to 2.1.3. for types. Online command function is supported, using parameters that don't need to pass in TABLE index. Alias: ZV_TYPE
Grammar	ZV_OBJTYPE(obj,tabId) or number = ZV_OBJTYPE(obj) obj: parameter defined by ZVOBJECT tabId: TABLE index that outputs the result
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat ZV_OBJTYPE(mat,0) ?TABLE(0) 'the result is 0 ZV_MATGENCONST(mat,3,3,0) 'generate a 0 matrix of 3 rows and 3 columns ZV_TYPE(mat,1) ?TABLE(1) ' the result is 2

2.4.2.ZV_OBJISEMPTY – Whether is Empty

Type	ZVOBJECT general operation
Description	It is used to judge whether ZVOBJECT variable is blank or not. If it is blank, which means variable has no type (such as, new defined variables) or variable data area is blank (such as, ZV_CLEAR is called). Online command function is supported, using parameters that don't need to pass in TABLE index. Alias: ZV_ISEMPTY
Grammar	ZV_OBJISEMPTY(obj,tabId) or value = ZV_OBJISEMPTY(obj)

	obj: parameter defined by ZVOBJECT tabId: TABLE index, output the result, 1 means blank, 0 means non-blank.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT obj
	ZV_OBJISEMPTY(obj,0)
	IF TABLE(0)=1 THEN
	PRINT "obj variable is empty"
	ELSE
	PRINT "obj variable is not empty"
	ENDIF

2.4.3.ZV_OBJCOPY – Copy Object Data

Type	ZVOBJECT general operation
Description	It is used to copy ZVOBJECT variable data, and all data are copied through deep copy except list, if it copies list, list element is still as the “quote” type. Alias: ZV_COPY
Grammar	ZV_OBJCOPY(src,dst) src: ZVOBJECT type, copy source object dst: ZVOBJECT type, target object that is copied
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src, dst ZV_MATGENCONST(src,3,3,0) 'generate a 0 matrix of 3 rows and 3 columns ZV_OBJCOPY(src,dst) 'copy data of obj to dst

2.4.4.ZV_OBJCLEAR – Clear

Type	ZVOBJECT general operation

Description	It is used to clear ZVOBJECT variables data, only data part is cleared, structural definition of ZVOBJECT still holds, and type can be obtained. For variables that are quoted from LIST, data also is cleared, namely, corresponding list element data will be cleared. Alias: ZV_CLEAR
Grammar	ZV_CLEAR(obj) obj: object parameter defined by ZVOBJECT
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT obj ZV_MATGENCONST(obj,3,3,0) 'generate a 0 matrix of 3 rows and 3 columns ZV_OBJCLEAR(obj) 'clear object data

2.4.5.ZV_OBJDETACH – Detach Quote & Connection

Type	ZVOBJECT general operation
Description	It is used to detach the quote or connection of ZVOBJECT variable. If there is no other variables' quote or connection, all are detached, including variable structure. For variables that are quoted from LIST, quote is disconnected, which means variables become blank, then its operations will not influence object that is quoted in LIST before. Alias: ZV_DETACH
Grammar	ZV_OBJDETACH(obj) obj: object parameter defined by ZVOBJECT
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

2.4.6.ZV_OBJTYPEFILE – Get File ZVOBJECT Variable Type

Type	ZVOBJECT general operation

Description	It is used to get the file's ZVOBJECT variables types. Please refer to 2.1.3. for types. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_OBJTYPEFILE(name,tabId) or type = ZV_OBJTYPEFILE(name) name: file path that saves ZVOBJECT object's file tabId: TABLE index that outputs the result type: variable type obtained directly
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	type = ZV_OBJTYPEFILE("mat.zvb") ?type 'ZVOBJECT variable type saved by mat.zvb

2.4.7.ZV_OBJREAD – Read ZVOBJECT Object

Type	ZVOBJECT general operation
Description	Read the ZVOBJECT object from the file. The file extension is zvb. And it is the binary type. Support region, matrix, contour, list, calibration, NCC template, shape template, measurement defect detector (contour pair), OCR classifier, color model, OCR sample.
Grammar	ZV_OBJREAD(obj,name) obj: the object to be read, ZVOBJECT type, the variable type is determined by the file content, refer to the ZV_OBJTYPEFILE instruction name: the path to read the file, extension zvb, if not, it will be added automatically
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT obj ZV_OBJREAD(obj,"mod.zvb") 'read the mod.zvb file in the path into the obj object

2.4.8.ZV_OBJWRITE – Save ZVOBJECT Object

Type	ZVOBJECT general operation
Description	Save the ZVOBJECT object into specified path. The file extension is zvb. And it is the binary type. Support region, matrix, contour, list, calibration, NCC template, shape template, measurement defect detector (contour pair), OCR classifier, color model, OCR sample.
Grammar	ZV_OBJWRITE(obj,name) obj: the object to be saved, ZVOBJECT type name: the path to save the file, extension zvb, if not, it will be added automatically
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT obj ZV_OBJREAD(obj,"mod.zvb") 'save the color mode mod as zvb format file

2.4.9.ZINDEX_LABEL – Get Index

Type	ZVOBJECT
Description	It is used to get the type of ZVOBJECT object index, object index is a digital No., which is similar to a pointer that points to data.
Grammar	index = ZINDEX_LABEL (obj) obj: object parameter defined by ZVOBJECT
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	**Example 1:** ZVOBJECT img ZV_READIMAGE(img,"logo.png",0) 'read one image index = ZINDEX_LABEL(img) 'get the index of image **Example 2:** ZVOBJECT zvarray(100)

	ZVOBJECT aa,bb,cc DIM zind zind = ZINDEX_LABEL(zvarray) ZV_READIMAGE(ZINDEX_ZVOBJ(zind)(1), "logo.png", 0) 'read image ZV_LATCH(ZINDEX_ZVOBJ(zind)(1),0) 'show image in latch channel 0 **Example 3:** GLOBAL SUB objindx(BYREF obj(100) as ZVOBJECT) ZV_LATCH(obj(1),0) 'show the image DELAY(2000) ZV_LATCH(obj(10),0) 'show the image DELAY(2000) ZV_LATCH(obj(1),0) 'show the image ENDSUB GLOBAL SUB objtest () ZVOBJECT zvarray(100) DIM zind zind = ZINDEX_LABEL(zvarray) ZV_READIMAGE(ZINDEX_ZVOBJ(zind)(1), "img1.bmp", 0) ZV_READIMAGE(ZINDEX_ZVOBJ(zind)(10), "img1.bmp", 0) objindx(ZINDEX_ZVOBJ(zind) ENDSUB

2.4.10. ZINDEX_ZVOBJ – Get Index Data

Type	ZVOBJECT
Description	It is used to get the data of index.
Grammar	ZVINDEX_ZVOBJ (index) index: index No.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	Refer to ZINDEX_LABLE example.

2.5.Parameters Related

2.5.1.Parameters Description

The system parameter retains the initial value, and the beginning part of the program can be modified as needed. “name” is the parameter name, the type is character string type, and it is case-sensitive, “value” is the parameter value. After the controller is started, all parameters are the initial values, and all modifications will remain until they are modified again or the controller is restarted.

The parameters of this summary can be set through the instructions in 2.5.2, and the corresponding type of read and write instructions can be selected according to the type of the parameter.

Note: the "C:\" path is automatically converted to "flash\", and "A:\" is the U disk path.

2.5.1.1. Data Catalogue

Name	“DataDir”
Type	Character string
Description	This is the default catalogue for file operation. When file path belongs to relative path, path automatically adds this catalog. For absolute path, no influence. When blank character string is passed, it will automatically modify as the value of parameter “DefDataDir”. Relative path is used best for file operation, in this way, simulator and controller can be compatible.
Initial Value	Controller path is “/zmc/flash/”, for simulator, it is under path of “flash”.

2.5.1.2. Default Data Directory

Name	“DefDataDir”
Type	Character string

Description	It means default value of data catalog, when blank character string is passed, it will automatically modify as initial value.
Initial Value	Controller path is “/zmc/flash/", for simulator, it is under path of “flash".

2.5.1.3. List Parameter Name

Name	“NameList" / “"
Type	Character string
Description	List all parameter names, separated by ";", integer, floating point and string parameters, with additional newlines between types. This function is also implemented if the parameter name is empty.
Permission	Read-only

2.5.1.4. List Parameter Information

Name	“ParamList"
Type	Character string
Description	List all parameter information, one parameter per row, including parameter name, type, permission, current value information, type I-integer, D-floating point, S-string, permission R-readable, W-writable, C-naming, command permission is a kind of write-only permission, and is also executed by writing parameter instructions.
Permission	Read-only

2.5.1.5. All Parameters Resume Default Values

Name	“ResetAll"
Type	Integer
Description	For command parameters with write-only permission, after executing the write command, all parameters except "DefDataDir" will be restored to their default values. The parameter values will have no

	effect.
Permission	Read-only & Command

2.5.1.6. Grab Timeout

Name	"GrabTimeout"
Type	Floating type
Range	>0
Unit	ms
Initial Value	2000

2.5.1.7. Image Getting Timeout

Name	"CamGetTimeout"
Type	Floating type
Range	>0
Unit	ms
Initial Value	30000

2.5.1.8. Shape Template Creating Level

Name	"ShapeCreateLevel"
Type	Integer
Range	0~4. 0~3 means some parts of template features are created. The value is bigger, created feature of template is more. 4 means template is created fully. 0 means self-adaption method. According to memory occupied by template, select 1 or 2. Parts of features are created through 1-3, remaining template features will be generated automatically during matching instruction, which means matching efficiency will be lower, but the memory size occupied by template features will be less.

Initial Value	0

2.5.1.9. Shape Template Creating Timeout

Name	"ShapeCreateTimeout"
Type	Floating type
Range	>0
Unit	ms
Initial Value	2000

2.5.1.10. Shape Template Matching Timeout

Name	"ShapeFindTimeout"
Type	Floating type
Range	>0
Unit	ms
Initial Value	2000

2.5.1.11. NCC Template Creating Timeout

Name	"NccCreateTimeout"
Type	Floating type
Range	>0
Unit	ms
Initial Value	2000

2.5.1.12. NCC Template Matching Timeout

Name	"NccFindTimeout"
Type	Floating type
Range	>0

Unit	ms
Initial Value	2000

2.5.1.13. Image Distortion Correction Mode

Name	“CalibRectMode”
Type	Integer
Range	0 – correction without black border 1 – correction with full pixel, which means there may be with black border.
Default Value	0

2.5.1.14. Line

Name	“LineType”
Type	Integer
Range	0 -- anti-aliasing 1-4 – connection 2-8 – connection
Default Value	2

2.5.1.15. Line Width

Name	“LineWidth”
Type	Integer
Range	1~8192
Unit	Pixel
Initial Value	1

2.5.1.16. Graphic Drawing Fill

Name	"IsDrawFill"
Type	Integer
Range	When setting drawing graphics, whether the closed graphics is filled inside, the graphics drawing instructions are other drawing instructions except the text drawing instructions.
Range	0: not filled, only draw graphic edge, 1: fill
Initial Value	0

2.5.1.17. Text Drawing Fill

Name	"IsTextFill"
Type	Integer
Range	Set whether to fill the interior when drawing text. If not filled, only the outer outline of the text will be drawn.
Range	0: not filled, 1: fill
Initial Value	1

2.5.1.18. Text Drawing Base Position

Name	"TextBase"
Type	Integer
Range	Set the reference position of the text when drawing text, that is, the point on the text area corresponding to the coordinate point passed in when drawing text. If set to 0, the drawn text will be located at the upper right position of the coordinate point, and if it is 1, it will be located on the bottom right position of the coordinate point
Range	0: left bottom corner, 1: left upper corner
Initial Value	0

2.5.1.19. Whether Shape Matching Allows Exceeding Border

Name	"ShapeOnBorder"
Type	Integer
Description	When setting shape template matching, whether the contour of template exceeds matched ROI area. Please note this parameter has one certain influence on matching efficiency.
Range	0 – target doesn't exceed image border 1 – target can exceed image border
Initial Value	0

2.5.1.20. Shape Matching Expansion Interface

Name	"ExtensionShape"
Type	Integer
Description	Set whether the shape matching function uses the expansion pack algorithm. Turning on the expansion pack algorithm requires an expansion pack plug-in and plug-in related dependencies. After successful opening, the extended algorithm will be used to create a template, and the matching will be automatically selected based on the template type. If it fails to be opened, an error will be reported and the parameter values will not be modified.
Range	0: inner algorithm, 1: Halcon expansion pack algorithm
Initial Value	0

2.5.1.21. Measurement Threshold Mode

Name	"MrThreshMode"
Type	Integer
Description	Threshold mode that measures gradient (contrast). The relative threshold mode is normalized to 255 for the maximum gradient. A higher contrast threshold can also be obtained for low-contrast images. The compatibility mode uses different threshold modes for

	different functions, the relative threshold mode is used for area measurement and geometric measurement, and the absolute threshold mode is used for measurement defects.
Range	-1: compatible mode, 0: relative threshold, 1: absolute threshold
Initial Value	-1

2.5.1.22. Minimum Measured Gradient Threshold

Name	"MrMinThresh"
Type	Integer
Description	It means the minimal measured gradient threshold when in relative threshold mode. If one value is smaller than this one, it is considered as noise point, which means it will not be as measurement point, then it is only valid under relative threshold mode.
Range	>0
Default Value	12

2.5.1.23. Maximal Camera Numbers

Name	"MaxCameras"
Type	Integer
Permission	Read-only
Value	4

2.5.1.24. Image Maximal Dimension

Name	"MaxDims"
Type	Integer
Permission	Read-only
Value	2

2.5.1.25. Image Maximal Channel Numbers

Name	"MaxChannels"
Type	Integer
Permission	Read-only
Value	4

2.5.1.26. Image / Matrix Maximal Size

Name	"MaxSize"
Type	Integer
Permission	Read-only
Unit	Pixel
Value	32766

2.5.1.27. Version No.

Name	"Version"
Type	Character string
Permission	Read-only

2.5.1.28. Hardware Platform

Name	"Platform"
Type	Character string
Permission	Read-only
Value	Windows 64bit - "Win_x64" Windows 32bit - "Win_x86" Linux Arm64 - "Linux_aarch64" Linux 64bit - "Linux_x64"

2.5.2. Parameters Reading & Writing

2.5.2.1. ZV_SETSYSINT – Integer Type Setting

Type	Parameters reading and writing
Description	It is used to set parameter value in integer type.
Grammar	ZV_SETSYSINT (name, value) name: parameter name, character string type value: parameter value that is set, integer
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_SETSYSINT ("LineWidth", 1) 'set line width as 1
Related instruction	ZV_GETSYSINT (integer type reading)

2.5.2.2. ZV_GETSYSINT – Integer Type Reading

Type	Parameters reading and writing
Description	It is used to read parameter value in integer type. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_GETSYSINT (name, tabId) or value = ZV_GETSYSINT (name) name: parameter name, character string type tabId: TABLE index that saves read parameter value
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	Value = ZV_GETSYSINT ("LineWidth") 'read line width
Related instruction	ZV_SETSYSINT (integer type setting)

2.5.2.3. ZV_SETSYSDBL – Floating Type Setting

Type	Parameters reading and writing
Description	It is used to set parameter value in floating point type.

Grammar	ZV_SETSYSDBL (name, value) name: parameter name value: parameter value that is set, in floating point type
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_SETSYSDBL ("GrabTimeout", 5000) 'set sampling timeout as 5000ms
Related instruction	ZV_GETSYSDBL (faloting type reading)

2.5.2.4. ZV_GETSYSDBL – Floating Type Reading

Type	Parameters reading and writing
Description	It is used to read parameter value in floating type. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_GETSYSDBL(name, tab_value) or value = ZV_GETSYSDBL (name) name: parameter name tab_value: TABLE index that saves read parameter value
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	Value = ZV_GETSYSDBL ("GrabTimeout") 'read sampling timeout
Related instruction	ZV_SETSYSDBL (floating type setting)

2.5.2.5. ZV_SETSYSSTR – Character String Type Setting

Type	Parameters reading and writing
Description	It is used to set parameter value in character string type.
Grammar	ZV_SETSYSSTR (name, value) name: parameter name value: parameter value that is set, in character string type
Controller	It is valid in controllers that support ZV function or they belong

	to 5XX series or above.
Example	ZV_SETSYSSTR("DataDir","") 'set default data catalogue ZV_GETSYSSTR("DataDir",20,0) 'read value in default path data catalog into TABLE(0) FOR i =0 TO 20 PRINT CHR(TABLE(i)) 'convert value in table into character string NEXT
	[ZV_GETSYSSTR (character string type reading)]

2.5.2.6. ZV_GETSYSSTR – Character String Type Reading

Type	Parameters reading and writing
Description	It is used to read parameter value in character string type. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_GETSYSSTR(name, max_len, tab_value) or str = ZV_GETSYSSTR (name) name: parameter name max_len: the max available length of tab_value tab_value: TABLE index that saves read parameter value
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_SETSYSSTR("DataDir","D:/data/") 'set default data catalogue ZV_GETSYSSTR("DataDir",20,0) 'read value in default path data catalog into TABLE(0) FOR i =0 TO 20 PRINT CHR(TABLE(i)) 'print default path NEXT
Related instruction	[ZV_SETSYSSTR (character string type setting)]

2.6. Error Processing

2.6.1. ZV_LASTERR – Error Code of Last Time

Type	Error processing
Description	It is used to read get or modify the error code appeared at the last time. Parameter dedicates the task No. that needs to get error information, if it is not with parameter, the task current is used. Note: ZDevelop command line and HMI interface both have independent task No.
Grammar	Get the last error code of the task: error = ZV_LASTERR (taskid) Modify the last error code of the task: ZV_LASTERR (taskid) = 0 taskid: task No., no parameter is brought, current task is used.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	**Example 1:** IF (0<>ZV_LASTERR) THEN RETURN Example 2: ZV_LASTERR(taskid)=0 'clear errors of current task appeared at the last time

2.6.2. ZV_RUNERR – Error Code when Running

Type	Error processing
Description	It is used to read get or modify the running error code, and the recorded error code is the instruction that runs incorrectly. Parameter dedicates the task No. that needs to get error information, if it is not with parameter, the task current is used. **Note:** ZDevelop command line and HMI interface both have independent task No. And the vision instruction of first error

	reported of the task is recorded, if there is no clearing operation, this information will be held all the time.
Grammar	Get the running error code of the task: error = ZV_RUNERR (taskid) Modify the running error code of the task: ZV_RUNERR (taskid) = 0
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	**Example 1:** IF (0<>ZV_RUNERR) THEN RETURN 'if the current task error code is not 0, then return **Example 2:** ZV_RUNERR(taskid)=0 'clear running errors of taskid task.

2.6.3.ZV_RUNERRSTR – Running Error Code Information Description

Type	Error processing
Description	It is used to get information description of running errors.
Grammar	error = ZV_RUNERRSTR (taskId)
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	**Example:** error = ZV_RUNERRSTR() 'get running error code description information of current task

Chapter III File Operation

The file name can use absolute path or relative path. The relative path is relative to the data directory specified by the system configuration "DataDir", and the data directory can be modified as a new directory through the system configuration.

The default data directory in the simulator is the flash subdirectory of the directory where the simulator exe program is located.

3.1.Matrix

3.1.1.ZV_READMATRIX – Reading

Type	Matrix
Description	It is used to read matrix file. xml, yaml and zvb files are supported, zvb is customized binary system type. **Alias: ZV_MATREAD**
Grammar	ZV_READMATRIX(mat,name[,param=0]) mat: ZVOBJECT type, the matrix that is read name: matrix path that is to be read, the default type is zvb param: reserved parameter, it must be 0
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat ZV_READMATRIX(mat,"mat.zvb",0) 'read mat.zvb file of the path into mat
Related Instruction	ZV_WRITEMATRIX

3.1.2.ZV_WRITEMATRIX – Storage

Type	Matrix

Description	It is used to save matrix into specified path. The file expansion name is zvb, which is a kind of customized binary system. **Alias: ZV_MATWRITE**
Grammar	ZV_WRITEMATRIX(mat,name[,param=0]) mat: ZVOBJECT type, the matrix that is saved name: matrix path that is to be saved, the default type is zvb param: reserved parameter, it must be 0
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat ZV_WRITEMATRIX(mat,3, 3, 0) 'generate a 0 matrix of 3 rows and 3 columns ZV_WRITEMATRIX(mat,"mat.zvb",0) 'save mat matrix as zvb format file
Related Instruction	ZV_READMATRIX

3.2. Image

3.2.1. ZV_READIMAGE – Image Reading

<table>
<tr><th>Type</th><td>Image</td></tr>
<tr><th>Description</th><td>It is used to read image by getting the value according to "type".
Alias: ZV_IMGREAD</td></tr>
<tr><th>Grammar</th><td>ZV_READIMAGE(img,name[,type=0])
img: ZVOBJECT type, the image that is read
name: the path of the image to be read. The path can be set as an absolute path or a relative path. The relative path can be set in the default directory. The supported extension is bmp, jpg or png. If the file has no extension name, then add the "bmp" extension name.
type: control parameters
<table>
<tr><th>Type</th><th>Description</th></tr>
<tr><td>0</td><td>Original format image is read</td></tr>
</table>
</td></tr>
</table>

	1 — Gray image is read 2 — RGB image is read
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img,"logo.png",0) 'read log and png images in original image format
Related Instruction	ZV_LATCH (image showing)

3.2.2.ZV_WRITEIMAGE – Image Storage

Type	Image
Description	It is used to save image. Formats of bmp, jpg and png are supported, and for jpg format, single channel or three-channel 8-bit images can be saved. For bmp and png formats, single-channel, three channel or four-channel 8-bit images can be saved. In addition, 16-bit single channel images can be saved in png format. **Alias: ZV_IMGWRITE**
Grammar	ZV_WRITEIMAGE(img,name[,param=0]) img: ZVOBJECT type, the image that is read name: image path name. According to expansion name, confirm image format, bmp, jpg or png are supported. If the file has no extension name, then add the "bmp" extension name. param: image compression parameter, it is >0 and ≤ 100. The compression level is lower, image effect is better. 0 is the default value.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_IMGGENCONST(img,100,100,2,2,0) 'generate one 100*100 image "img" of 32-bit short type with symbol from TABLE (0), and the channel is 2. ZV_WRITEIMAGE(img,"1.bmp",0)

	'save the image into default path, and name it as 1.bmp
Related Instruction	ZV_READIMAGE (image reading)

3.3.Region

3.3.1.ZV_READREGION – Read Region

Type	Region
Description	It is used to read region from the file, the file expansion name is zvb, it is binary type. **Alias: ZV_REREAD**
Grammar	ZV_READREGION (re, name) re: region to be read, ZVOBJECT type name: region file path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT re ZV_REREAD(re,"re.zvb") 'read re. zvb file into re
Related Instruction	ZV_WRITEREGION

3.3.2.ZV_WRITEREGION – Save Region

Type	Region
Description	It is used to save region into the file, the file expansion name is zvb, it is binary type. **Alias: ZV_REWRITE**
Grammar	ZV_WRITEREGION (re, name) re: region to be saved, ZVOBJECT type name: region file path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT re ZV_REGENLINE(re,50,50,200,200) ZV_REWRITE(re,"re.zvb") 'save region re as zvb format file
Related Instruction	ZV_READREGION

3.4. Template

3.4.1. ZV_READNCCMOD – NCC Mode Reading

Type	Template
Description	It is used to read NCC template file, the file expansion file is zvb, which is customized binary system. **Alias: ZV_NCCREAD**
Grammar	ZV_READCNNMOD (mod, name) mod: ZVOBJECT type, NCC template that is read name: NCC template path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mod ZV_READNCCMOD(mod,"mod.zvb") 'read mod. zvb file into mod.
Related Instruction	ZV_WRITECNNMOD

3.4.2. ZV_WRITENCCMOD – NCC Mode Storage

Type	Template
Description	It is used to save NCC template into specified path, the file expansion file is zvb, which is customized binary system. **Alias: ZV_NCCWRITE**
Grammar	ZV_READCNNMOD (mod, name) mod: ZVOBJECT type, NCC template that is saved

	name: NCC template path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mod ZV_WRITENCCMOD(mod,"mod.zvb") 'save template mod as zvb format file
Related Instruction	ZV_READCNNMOD

3.4.3.ZV_READSHAPEMOD – Shape Mode Reading

Type	Template
Description	It is used to read shape template file, the file expansion file is zvb, which is customized binary system. **Alias: ZV_SHAPEREAD**
Grammar	ZV_READSHAPEMOD (mod, name) mod: ZVOBJECT type, shape template that is read name: shape template path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mod ZV_READSHAPEMOD(mod,"mod.zvb") 'read mod. zvb file into mod.
Related Instruction	ZV_WRITESHAPEMOD

3.4.4.ZV_WRITESHAPEMODE – Shape Mode Storage

Type	Template
Description	It is used to save shape template into specified path, the file expansion file is zvb, which is customized binary system. **Alias: ZV_SHAPEWRITE**

Grammar	ZV_READCNNMOD (mod, name) mod: ZVOBJECT type, shape template that is saved name: shape template path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mod ZV_WRITESHAPEMOD(mod,"mod.zvb") 'save template mod as zvb format file
Related Instruction	ZV_READSHAPEMOD

3.5.Calibration

3.5.1.ZV_CALREAD – Calibration Parameters Reading

Type	Calibration
Description	It is used to read calibration parameter file, the file expansion file is zvb, which is customized binary system.
Grammar	ZV_CALREAD(calParam, name) calParam: ZVOBJECT type, calibration parameter that is read name: calibration parameter path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT calParam ZV_CALREAD(calParam,"calParam.zvb") 'read calParam. zvb file into calibration parameter calParam.
Related Instruction	ZV_CALWRITE

3.5.2.ZV_CALWRITE – Calibration Parameters Storage

Type	Calibration

Description	It is used to save calibration parameters into specified path, the file expansion file is zvb, which is customized binary system.
Grammar	ZV_CALWRITE(calParam, name) calParam: ZVOBJECT type, calibration parameter that is saved. name: calibration parameter path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT calParam ZV_CALWRITE(calParam,"calParam.zvb") 'save calibration parameter calParam as zvb format file.
Related Instruction	ZV_CALREAD

3.6.Color

3.6.1.ZV_CLRMODREAD – Color Mode Reading

Type	Color
Description	It is used to read color mode file, the file expansion file is zvb, which is customized binary system.
Grammar	ZV_CLRMODREAD(mod, name) mod: ZVOBJECT type, color mode that is read name: color mode path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mod ZV_CLRMODREAD(mod,"mod.zvb") 'read mod. zvb file into color mode "mod".
Related Instruction	ZV_CLRMODWRITE

3.6.2.ZV_CLRMODWRITE – Color Mode Storage

Type	Color
Description	It is used to save color mode into specified path, the file expansion file is zvb, which is customized binary system.
Grammar	ZV_CLRMODWRITE(mod, name) mod: ZVOBJECT type, color mode that is saved name: color mode path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mod ZV_CLRMODWRITE(mod,"mod.zvb") 'save color mode "mod" as zvb format file
Related Instruction	ZV_CLRMODREAD

3.7.OCR

3.7.1.ZV_OCRREADSVM – SVM Classifier Reading

Type	OCR
Description	It is used to read SVM classifier file that supports vector machine and to identify characters, the file expansion file is zvb, which is customized binary system.
Grammar	ZV_OCRREADSVM(svm, name) svm: ZVOBJECT type, svm classifier that is read name: svm classifier path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT svm ZV_OCRREADSVM(svm,"svm.zvb") 'read svm. zvb file into classifier "svm".

Related Instruction	ZV_OCRWRITESVM

3.7.2.ZV_OCRWRITESVM – SVM Classifier Storage

Type	OCR
Description	It is used to save SVM classifier that supports vector machine into specified path, the file expansion file is zvb, which is customized binary system.
Grammar	ZV_OCRREADSVM(svm, name) svm: ZVOBJECT type, svm classifier that is saved name: svm classifier path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT svm ZV_OCRWRITESVM(svm,"svm.zvb") 'save classifier "svm" as zvb format file
Related Instruction	ZV_OCRREADSVM

3.7.3.ZV_OCRREADMLP – MLP Classifier Reading

Type	OCR
Description	It is used to read neural network MLP classifier file and to identify characters, the file expansion file is zvb, which is customized binary system.
Grammar	ZV_OCRREADMLP(mlp, name) svm: ZVOBJECT type, mlp classifier that is read name: mlp classifier path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mlp ZV_OCRREADSVM(mlp,"mlp.zvb")

	'read mlp. zvb file into classifier "mlp".
Related Instruction	ZV_OCRWRITEMLP

3.7.4.ZV_OCRWRITEMLP – MLP Classifier Storage

Type	OCR
Description	It is used to save MLP classifier into specified path, the file expansion file is zvb, which is customized binary system.
Grammar	ZV_OCRWRITEMLP(mlp, name) svm: ZVOBJECT type, mlp classifier that is saved name: mlp classifier path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mlp ZV_OCRWRITESVM(mlp,"mlp.zvb") 'save classifier "mlp" as zvb format file
Related Instruction	ZV_OCRREADMLP

3.8.Contour

3.8.1.ZV_CONTREAD – Contour Reading

Type	Contour
Description	It is used to read contour file, and the file expansion file is zvb, which is customized binary system.
Grammar	ZV_CONTREAD(cont, name) svm: ZVOBJECT type, contour that is read name: contour file path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT cont

	ZV_OCRREADSVM(cont,"cont.zvb") 'read cont. zvb file into contour "cont".
Related Instruction	ZV_CONTWRITE

3.8.2.ZV_CONTWRITE – Contour Storage

Type	Contour
Description	It is used to save contour file into specified path, the file expansion file is zvb, which is customized binary system.
Grammar	ZV_CONTWRITE(cont, name) svm: ZVOBJECT type, contour file that is saved name: contour file path, expansion name is zvb, if there is no expansion name, add it automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT cont ZV_CONTWRITE(cont,"cont.zvb") 'save contour "cont" as zvb format file
Related Instruction	ZV_CONTREAD

3.9.List

3.9.1.ZV_LISTREAD – List Reading

Type	List
Description	It is used to read list file, and the file expansion file is zvb or zpk, zvb is customized binary system, and zpk is the package type.
Grammar	ZV_LISTREAD(list, name) list: ZVOBJECT type, list that is read name: list file path, expansion name is zvb or zpk, if there is no expansion name, zvb is added automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT list_param ZV_LISTREAD(list_param, "list.zvb") 'read list. zvb file into list "list_param".
Related Instruction	ZV_LISTWRITE

3.9.2. ZV_LISTWRITE – List Storage

Type	List
Description	It is used to save list file into specified path, the file expansion file is zvb or zpk. zvb is customized binary system, and zpk is the package type, which can be released as directory structure through UNPACK command. Element is named by the No., and the image is saved as BMP format. List is the subdirectory, the saved form is zvb.
Grammar	ZV_LISTWRITE(list, name) list: ZVOBJECT type, contour file that is saved name: path, expansion name is zvb or zpk, if there is no default zvb form, zvb expansion name is added automatically.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT list_param ZV_LISTWRITE(list_param,"list.zvb") 'save list as list.zvb format file
Related Instruction	ZV_LISTREAD

3.10. Compression Package

3.10.1. PACK – File / Directory Packing & Compressing

Type	File
Description	It is used to pack file or directory as compressed file with a suffix of .zpk.

Grammar	PACK(pack_name, mode, path) pack_name: the file name after packing and compression, with the extension of .zpk, which cannot be omitted mode: the processing method of the existing compressed file: 0 - report an error directly, 1 - replace the existing file in the package, and append the file that does not exist in the package. 2 - update the existing file in the package, that is, append files that do not exist in the package are appended path: directory / file name to be compressed
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	PACK("123.zpk", 1, "123") 'compress the 123 folder to generate the 123.zpk file. PACK("A:/123.zpk", 1, "123") 'generate the compressed file 123. zpk and that is located in the U disk directory
Related Instruction	UNPACK

3.10.2. UNPACK – Packed File Decompressing

Type	File
Description	It is used to decompress the compressed file with the .zpk suffix
Grammar	UNPACK(pack_name, mode[, path]) pack_name: pack compressed file, .zpk extension, which cannot be omitted mode: the processing method for the existing files to be decompressed: 0-report an error directly. 1-decompress all files and overwrite existing files, 2-skip existing files and decompress the remaining files. path: decompress destination path, if it is empty or default, it will be decompressed to the current directory.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	UNPACK("123.zpk", 1) 'Unpack 123.zpk to the directory
Related Instruction	PACK

Chapter IV Matrix

4.1.Generate the Matrix

4.1.1.ZV_MATGENCONST – Constant Creating

Type	Generate
Description	It means that matrix mat is generated through constant value “value”, the max memory is 512M.
Grammar	ZV_MATGENCONST(mat,rows,cols,value) mat: ZVOBJECT type, matrix that is generated rows: rows of the matrix, range is (0, 32766] cols: columns of the matrix, range is (0, 32766] value: constant value
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	<table><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td></tr></table> ZVOBJECT mat ZV_MATGENCONST(mat,3, 3, 0) ‘generate 3*3 constant matrix, matrix values are 0

4.1.2.ZV_MATGENEYE – Size of Matrix

Type	Generate
Description	It is used to generate a matrix with a size of (size, size), the max memory is 512M.
Grammar	ZV_MATGENEYE(mat, size) mat: ZVOBJECT type, matrix that is generated size: rows and columns of the matrix, the range is (0,

<table>
<tr><td></td><td>32766]</td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><td>Example</td><td><table><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td></tr></table>ZVOBJECT mat
ZV_MATGENEYE(mat,3)
‘generate a matrix of 3*3 size</td></tr>
</table>

4.1.3.ZV_MATGENDATA – Data Creating

<table>
<tr><td>Type</td><td>Generate</td></tr>
<tr><td>Description</td><td>It means that matrix mat is generated through data of tab_data, the max memory is 512M.</td></tr>
<tr><td>Grammar</td><td>ZV_MATGENDATA(mat,rows,cols,tabId)
mat: ZVOBJECT type, matrix that is generated
rows: rows of the matrix, range is (0, 32766]
cols: columns of the matrix, range is (0, 32766]
tabId: TABLE index, which is used to generate data for matrix, the number of rows*cols data. The maximum allowable data of controller is 250 TABLE data, and same as simulator.</td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><td>Example</td><td><table><tr><td>0</td><td>1</td><td>2</td></tr><tr><td>3</td><td>4</td><td>5</td></tr><tr><td>6</td><td>7</td><td>8</td></tr></table>ZVOBJECT mat
TABLE(0, 0, 1, 2, 3, 4, 5, 6, 7, 8)
'store the numbers 0-8 into TABLE(0)</td></tr>
</table>

	ZV_MATGENDATA(mat,3,3,0) 'save data generated in TABLE into the matrix mat

4.2. Basic Parameters

4.2.1. ZV_MATINFO – Basic Information

Type	Basic parameters
Description	It is used to get basic information of matrix.
Grammar	ZV_MATGENEYE(mat, tabld) mat: ZVOBJECT type, matrix tabld: TABLE index that outputs matrix information, TABLE type, they are rows, columns, element size (memory control occupied by one single element, the unit is byte) in order.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat ZV_MATGENEYE(mat,3) 'generate a matrix of 3*3 size ZV_MATINFO(mat,0) 'output array information to TABLE (0), the sequence is rows, columns, element size.

4.2.2. ZV_MATISVALID – Whether is Valid

Type	Basic parameters
Description	It is used to judge whether the matrix is valid. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_MATISVALID(mat,tabld) or value = ZV_MATISVALID(mat) mat: ZVOBJECT type, source matrix tabld: TABLE index, output result is saved into TABLE (tabld), 0 – invalid, 1 – valid
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT mat 'define one ZVOBJECT variable
	ZV_MATGENEYE(mat,3) 'generate a matrix of 3*3 size
	ZV_MATISVALID(mat,0)'judge whether mat is valid, then save result into TABLE(0)
	IF TABLE(0)=0 THEN
	PRINT "invalid"
	END IF

4.2.3.ZV_MATROWS – Get Rows of Matrix

Type	Basic parameters
Description	It is used to get the matrix's rows. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_MATROWS(mat,tabId) or count = ZV_MATROWS(mat) mat: ZVOBJECT type, source matrix tabId: TABLE index, output result is saved into TABLE (tabId)
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat ZV_MATGENEYE(mat,3) 'generate a matrix of 3*3 size ZV_MATROWS(mat,0) 'get the rows of the matrix, then save result into TABLE(0)

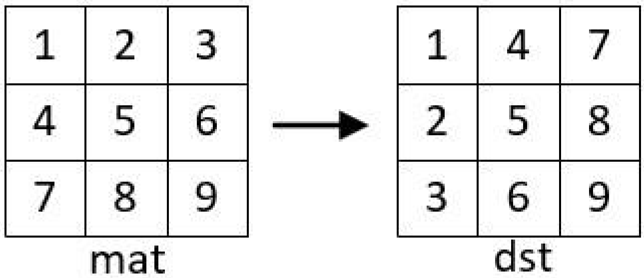
4.2.4.ZV_MATCOLS – Get Columns of Matrix

Type	Basic parameters
Description	It is used to get the matrix's columns. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_MATCOLS (mat,tabId) or count = ZV_MATCOLS (mat) mat: ZVOBJECT type, source matrix

	tabId: TABLE index, output result is saved into TABLE (tabId)
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat ZV_MATGENEYE(mat,3) 'generate a matrix of 3*3 size ZV_MATCOLS(mat,0) 'get the columns of the matrix, then save result into TABLE(0)

4.3. Matrix Operation

4.3.1. ZV_TRANSPOSE – Transpose

<table>
<tr><td>Type</td><td>Matrix operation</td></tr>
<tr><td>Description</td><td>It is used to find matrix transpose.</td></tr>
<tr><td>Grammar</td><td>ZV_TRANSPOSE(src,dst)
src: ZVOBJECT type, source matrix
dst: ZVOBJECT type, matrix after transposed</td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><td>Example</td><td>

ZVOBJECT mat, dst
TABLE(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
ZV_MATGENDATA(mat,3,3,0)
'generate the data in TABLE to the matrix mat
ZV_TRANSPOSE(mat,dst)
‘transpose the source matrix src into dst matrix</td></tr>
</table>

4.3.2.ZV_INVERT – Inverse Matrix

<table>
<tr><td>Type</td><td>Matrix operation</td></tr>
<tr><td>Description</td><td>It is used to find inverse matrix, method is matrix factorization algorithm.</td></tr>
<tr><td>Grammar</td><td>ZV_INVERT(src, dst, method)
src: ZVOBJECT type, source matrix, matrix must be phalanx.
dst: ZVOBJECT type, matrix after transposed
method: algorithm for inverse matrix
<table>
<tr><th>method</th><th>Description</th></tr>
<tr><td>0</td><td>LU decomposition, the matrix must be phalanx.</td></tr>
<tr><td>1</td><td>Feature values decomposition, the matrix must be symmetric.</td></tr>
<tr><td>2</td><td>Cholesky decomposition, the matrix must be positive definite.</td></tr>
<tr><td>3</td><td>SVD decomposition</td></tr>
</table></td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><td>Example</td><td>ZVOBJECT mat, dst
ZV_MATGENEYE(mat,3) 'generate a matrix of 3*3 size
ZV_INVERT (mat, dst, 3) ‘use SVD decomposition to find the inverse matrix, and it can find the generalized inverse matrix.</td></tr>
</table>

4.3.3.ZV_MATRIXMULT – Matrix Multiple

Type	Matrix operation
Description	Two matrices execute matrix multiple, and the column of the first matrix needs to be same as the rows of the second matrix.

Grammar	ZV_MATRIXMULT (mat1, mat2, dst) mat1: ZVOBJECT type, the multiplier of matrix multiplication mat2: ZVOBJECT type, the multiplicand of matrix multiplication dst: ZVOBJECT type, the result of matrix multiplication
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat1, mat2, dst ZV_MATGENEYE(mat1,3) ZV_MATGENCONST(mat2,3,3,2) ZV_MATRIXMULT(mat1,mat2,dst) 'multiply two matrices, dst is the result

4.4. Access

4.4.1. ZV_MATGETVAL – Get the Value

Type	Access
Description	It is used to get the matrix value of specified position into TABLE. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_MATGETVAL(mat,row,col,tabId) or val= ZV_MATGETVAL(mat,row,col) mat: ZVOBJECT type, source matrix row: get the row coordinates of the value col: get the column coordinates of the value tabId: TABLE index, obtained value
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \boxed{1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$ ZVOBJECT mat ZV_MATGENEYE(mat,3) 'generate a matrix of 3*3 ZV_MATGETVAL(mat,1,1,0) 'get the value of coordinate (1,1) and store it in TABLE(0)

4.4.2.ZV_MATSETVAL – Set the Value

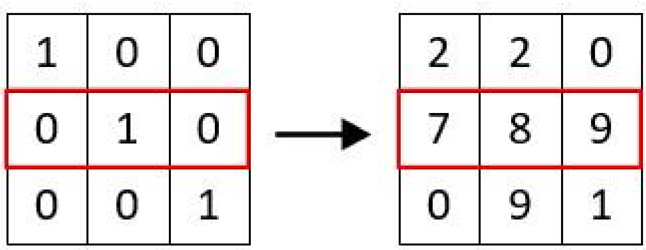
Type	Access
Description	It is used to modify the matrix value of specified position.
Grammar	ZV_MATSETVAL(mat,row,col,val) or val= ZV_MATGETVAL(mat,row,col) mat: ZVOBJECT type, source matrix row: modify the row coordinates of the value col: modify the column coordinates of the value val: value after modification
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \boxed{1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & \boxed{5} & 0 \\ 0 & 0 & 1 \end{bmatrix}$ ZVOBJECT mat ZV_MATGENEYE(mat,3) 'generate a matrix of 3*3 ZV_MATSETVAL(mat,1,1,5) 'modify the value of specified position as 5

4.4.3.ZV_MATGETROW – Get One Row

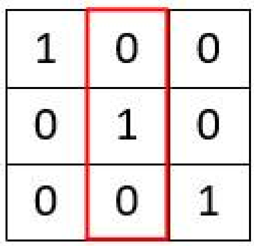
Type	Access
Description	It is used to get the data of specified row into TABLE.
Grammar	ZV_MATGETROW(mat,row,tabLen,tabId) mat: ZVOBJECT type, source matrix row: get the row index of data tabLen: max available length of result in TABLE, it should be ≥ matrix column tabId: TABLE index, obtained row data
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	$\begin{array}{\|c\|c\|c\|} \hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 1 \\ \hline \end{array}$ ZVOBJECT mat ZV_MATGENEYE(mat,3) 'generate a matrix of 3*3 ZV_MATGETROW(mat,1,3,0) 'save the data in row 2 in order into TABLE (0) (starting index of TABLE)

4.4.4.ZV_MATSETROW – Set the Row

Type	Access
Description	It is used to modify the data of specified row.
Grammar	ZV_MATSETROW(mat,row,tabNum,tabId) mat: ZVOBJECT type, the matrix to be modified row: the row to be modified, 0 is the starting row number tabNum: the number of data in tabId, which must be equal to the number of matrix columns tabId: TABLE index, row data

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT mat ZV_MATGENEYE(mat,3) 'generate a matrix of 3*3 ZV_MATSETVAL(mat,1,3,0) 'save 3 data of TABLE starting index into the first row of the matrix in order

4.4.5.ZV_MATGETCOL – Get One Column

Type	Access
Description	It is used to get the data of specified column into TABLE
Grammar	ZV_MATGETCOL(mat,row,tabLen,tabId) mat: ZVOBJECT type, source matrix col: obtained id of column, 0 is the starting column tabLen: max available length of result in TABLE, it should be ≥ matrix row tabId: TABLE index, obtained column data
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT mat ZV_MATGENEYE(mat,3) 'generate a matrix of 3*3 ZV_MATGETCOL(mat,1,3,0) 'save the data in column 2 in order

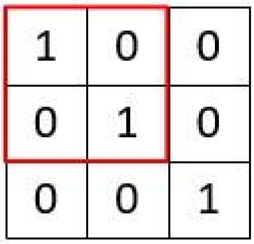
	into TABLE (0) (starting index of TABLE)

4.4.6.ZV_MATSETCOL – Set the Col

Type	Access
Description	It is used to modify the data of specified column.
Grammar	ZV_MATSETCOL(mat,col,tabNum,tabId) mat: ZVOBJECT type, the matrix to be modified col: the column to be modified, 0 is the starting column number tabNum: the number of data in tab_val, which must be equal to the number of matrix rows tabId: TABLE index, column data
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 7 & 0 \\ 0 & 8 & 0 \\ 0 & 9 & 1 \end{bmatrix}$ ZVOBJECT mat ZV_MATGENEYE(mat,3) 'generate a matrix of 3*3 ZV_MATSETCOL(mat,1,3,0) 'save 3 data of TABLE starting index into the first column of the matrix in order

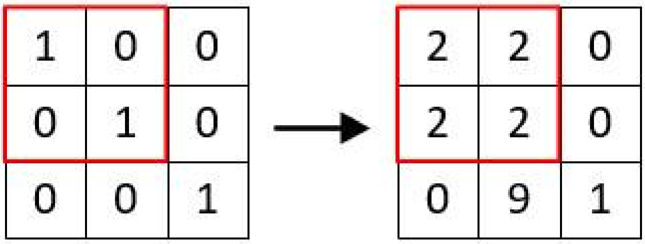
4.4.7.ZV_MATGETRANGE – Get Sub-Region Value

Type	Access
Description	It is used to get the matrix data of specified position into TABLE

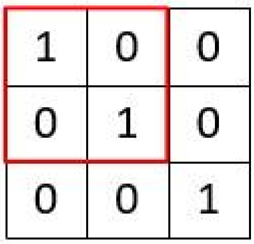
Grammar	ZV_MATGETRANGE(mat,y1,y2,x1,x2,tabId) mat: ZVOBJECT type, source matrix y1: starting row of the matrix, including the current row y2: ending row of the matrix, including the current row x1: starting column of the matrix, including the current column x2: ending column of the matrix, including the current column tabId: TABLE index, obtained sub-region data
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT mat ZV_MATGENEYE(mat,3) 'generate a matrix of 3*3 ZV_MATGETRANGE(mat,0,1,0,1,0) 'get the data of the region that includes row 1 – row 2 and column 1 – column 2, then save them into TABLE starting index TABLE (0) in order.

4.4.8.ZV_MATSETRANGE – Set Sub-Region Value

Type	Access
Description	It is used to set the data of TABLE into specified area of matrix.
Grammar	ZV_MATSETRANGE(mat,y1,y2,x1,x2,tabId) mat: ZVOBJECT type, source matrix y1: starting row of the matrix, including the current row y2: ending row of the matrix, including the current row x1: starting column of the matrix, including the current column x2: ending column of the matrix, including the current column

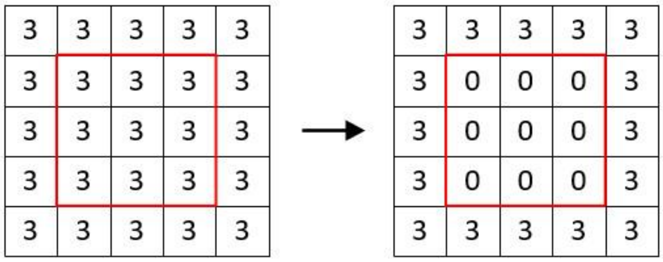
	tabId: TABLE index, obtained sub-region data
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT mat ZV_MATGENEYE(mat,3) 'generate a matrix of 3*3 ZV_MATSETRANGE(mat,0,1,0,1,0) 'set the data of TABLE (0) to the region that includes row 1 – row 2 and column 1 – column 2

4.4.9.ZV_MATGETSUB – Get Sub-Region Matrix

Type	Access
Description	It is used to get the sub-region of matrix.
Grammar	ZV_MATGETSUB(mat, sub, x, y, width, height) mat: ZVOBJECT type, source matrix sub: ZVOBJECT type, obtained sub matrix x: x coordinate of subregion y: y coordinate of subregion width: width of subregion height: height of subregion
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT sub_mat, mat

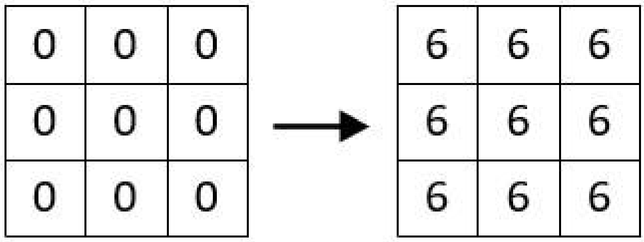
	ZV_MATGENEYE(mat,3) 'generate a matrix of 3*3 ZV_MATGETSUB(mat, sub_mat, 0, 0, 2, 2) 'in position mat (0,0), select one subregion matrix into sub

4.4.10. ZV_MATSETSUB – Set Sub-Region

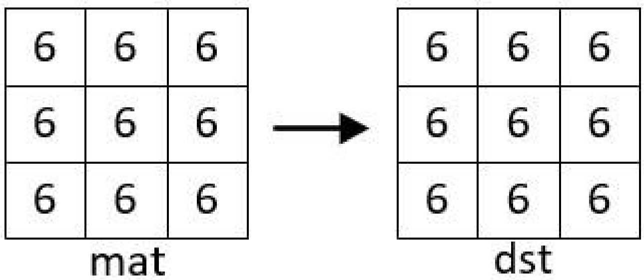
Type	Access
Description	It is used to modify subregion.
Grammar	ZV_MATSETSUB(mat, sub, x, y) mat: ZVOBJECT type, matrix to be modified sub: ZVOBJECT type, matrix for modification x: x coordinate of subregion that is to be modified y: y coordinate of subregion that is to be modified
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT sub_mat, mat ZV_MATGENCONST(sub_mat,3,3,0) 'generate a 3*3 constant matrix, the matrix values are all 0 ZV_MATGENCONST(mat,5,5,3) 'generate a 5*5 constant matrix, the matrix values are all 3 ZV_MATSETSUB(mat,sub_mat,1,1) 'set the value of sub_mat matrix at mat(1,1) position

4.4.11. ZV_MATSETCONST – Constant Filling

Type	Access
Description	Fill the matrix through constant val.

Grammar	ZV_MATSETCONST(mat, val) mat: ZVOBJECT type, matrix val: filling value
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT mat ZV_MATGENCONST(mat,3,3,0) 'generate a 3*3 constant matrix, the matrix values are all 0 ZV_MATGENCONST(mat,6) 'set the matrix values are all 6

4.4.12. ZV_MATCOPY – Copy

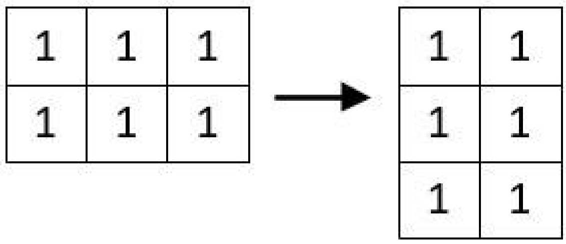
Type	Access
Description	It is used to copy the matrix.
Grammar	ZV_MATSETCONST(mat, dst) mat: ZVOBJECT type, source matrix dst: ZVOBJECT type, copied matrix
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT mat ZV_MATGENCONST(mat,3,3,6) 'generate a 3*3 constant matrix, the matrix values are all 6 ZV_MATCOPY(mat,dst) 'copy mat matrix into dst

4.4.13. ZV_MATSORT – Sorting

<table>
<tr><td>Type</td><td>Access</td></tr>
<tr><td>Description</td><td>It is used to sort for matrix rows and columns.</td></tr>
<tr><td>Grammar</td><td>ZV_MATSORT(mat,sortMat,idx,isAsec,isSortCol)
mat: ZVOBJECT type, matrix
sortMat: ZVOBJECT type, sorted matrix
idx: used to specify the row number or column number for sorting, which means the property value of corresponding row or column of idx is for sorting, for example, the row 0 of shape matched result represents the fraction.
isAsec: 1 - ascending, 0 - descending
isSortCol: 1 - sort column, 0 - sort row</td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><td>Example</td><td>
<table>
<tr><td></td><td></td><td></td><td>idx</td><td></td></tr>
<tr><td>score</td><td>x</td><td>y</td><td>angle</td><td>scale</td></tr>
<tr><td>0.9</td><td>10</td><td>70</td><td>-5</td><td>1.5</td></tr>
<tr><td>0.5</td><td>1</td><td>0</td><td>15</td><td>1</td></tr>
<tr><td>0.01</td><td>154</td><td>200</td><td>170</td><td>0.8</td></tr>
</table>
mat
→
<table>
<tr><td></td><td></td><td></td><td>idx</td><td></td></tr>
<tr><td>score</td><td>x</td><td>y</td><td>angle</td><td>scale</td></tr>
<tr><td>0.01</td><td>154</td><td>200</td><td>170</td><td>0.8</td></tr>
<tr><td>0.5</td><td>1</td><td>0</td><td>15</td><td>1</td></tr>
<tr><td>0.9</td><td>10</td><td>70</td><td>-5</td><td>1.5</td></tr>
</table>
dst
ZVOBJECT mat 'assume each column corresponds to an attribute for matched result matrix
ZVOBJECT dst
TABLE(0,0.5,1,0,15,1,0.9,10,70,-5,1.5,0.01,154,200,170,0.8)
ZV_MATGENDATA(mat,3,5,0) 'use the data in TABLE(0) to generate a 3*5 matrix
ZV_MATSORT(mat,dst,0,0,0) 'idx selects the first column (score attribute), and arranges the score attribute of the matrix mat in descending order by row</td></tr>
</table>

4.5. Transformation

4.5.1. ZV_MATRESHAPE – Adjust Rows & Columns

Type	Transformation
Description	It is used to adjust rows and columns. Note: the adjusted total size needs to be strictly equal.
Grammar	ZV_MATRESHAPE(mat,row) mat: ZVOBJECT type, matrix row: rows after adjusted, if it is 0, getting the original value
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT mat, dst ZV_MATGENCONST(mat,2,3,1) 'generate a 2*3 constant matrix, the matrix values are all 1 ZV_MATRESHAPE(mat,3) 'adjust the 2x3 matrix to 3x2 matrix

Chapter V Image

5.1.Image Generation

5.1.1.ZV_IMAGECONST – Image Generating from Data

<table>
<tr><th>Type</th><td>Generate</td></tr>
<tr><th>Description</th><td>It is used to generate the image through constant, and for the image size, it is recommended to be a multiple of 4, and the maximum memory is 512M.</td></tr>
<tr><th>Grammar</th><td>ZV_IMAGECONST(dst, w, h, ch, type, tabId)
dst: ZVOBJECT type, the generated image
w: the width of the generated image, range is (0,32766]
h: the height of the generated image, range is (0,32766]
cn: the number of channels of the generated image
type: the type of generated image<table><tr><th>Type</th><th>Description</th></tr><tr><td>0</td><td>8-bit without symbol 8U</td></tr><tr><td>1</td><td>16-bit without symbol 16U</td></tr><tr><td>2</td><td>32-bit with symbol 32S</td></tr><tr><td>3</td><td>64-bit with symbol 64F</td></tr></table>tabId: TABLE index, constant values of each channel of the image, one channel occupies one TABLE space.</td></tr>
<tr><th>Controller</th><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><th>Example</th><td>ZVOBJECT img
TABLE(0,255,128,64) 'store the constant value of each channel in the TABLE whose starting index is 0 (TABLE (0))
ZV_IMGGENCONST(img,4,4,3,0,0) 'use the data of TABLE(0) to generate an 8-bit without symbol 4*4 image img, whose number of channels is 3</td></tr>
</table>

5.1.2.ZV_IMGTILE – Image Combination

<table>
<tr><th>Type</th><td>Generate</td></tr>
<tr><th>Description</th><td>It is used to combine image tiles in an image list into one large image.</td></tr>
<tr><th>Grammar</th><td>ZV_IMGTILE(imgs,img,numCols,type)
imgs: the image list, ZVOBJECT type and list type, the number and type of image channels in the list must be the same, and the size in the splicing direction must be the same
img: the image generated by tile combination, ZVOBJECT type
numCols: the number of columns of the image tile, >1
type: image tiling method, as follows
<table>
<tr><th>Type</th><th>Description</th></tr>
<tr><td>0</td><td>Horizontal to the right, Z order type</td></tr>
<tr><td>1</td><td>Horizontal to the right, S round-trip</td></tr>
<tr><td>2</td><td>Vertically down, Z-sequential</td></tr>
<tr><td>3</td><td>Vertically down, S round-trip</td></tr>
<tr><td>4</td><td>Horizontal to the left, Z order type</td></tr>
<tr><td>5</td><td>Horizontal to the left, S round-trip</td></tr>
<tr><td>6</td><td>Vertically up, Z-sequential</td></tr>
<tr><td>7</td><td>Vertically up, S round-trip</td></tr>
</table></td></tr>
<tr><th>Controller</th><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><th>Example</th><td>ZVOBJECT imgList 'define image list
ZVOBJECT img, dst
ZV_READIMAGE(img1,"logo.png",0) 'read image
ZV_LISTINSERT(img1,imgList,-1) 'insert list
ZV_IMGTILE(imgList,dst,2,0)
'tile the images in the image list into a large image in a horizontal to right Z order</td></tr>
</table>

5.2. Image Acquisition

5.2.1. Camera Scanning

5.2.1.1. CAM_SCAN – Scan All Cameras

Type	Image acquisition related.
Description	It is used to scan cameras connected to the system and record all cameras information. **Note:** ➢ The camera and the controller need to be connected directly or use a Gigabit HUB, otherwise the scan may fail or affect the stability of the acquisition (acquisition failure, frame loss, image loss, etc., the specific phenomenon is related to the camera) ➢ It should be noted that both display and acquisition may cause the effect of frame loss, which needs to be distinguished. Display may result in the entire image not being updated, not in missing parts. ➢ Notes for Dahua cameras: For the VPLC516E controller, after the bus axis is enabled, there will be an error when scanning Dahua cameras, such as printing "recv no pdo 2". If the object is a Panasonic or Raytheon drive, no problem, but if it is a Delta drive, the axis will appear abnormal.

Grammar	CAM_SCAN(type,flag=0) type: scan type includes "zmvcbase", "zmotion", "mvision", "basler", "mindvision", "huaray", "dvpcamera", "daheng" and other types, and it supports multiple types mixed, but different cameras may affect the stability. It is recommended to use the same type of camera. When the type is empty, the default type will be used for scanning. The default type is "zmvcbases". The corresponding cameras of the scanning type are as follows: "zmvcbase" USB drive free camera "zmotion" Zmotion camera "mvision" Hikvision camera "basler" basler camera "mindvision" mindvision camera "huaray" Dahua camera "dvpcamera" do3think camera "daheng" daheng camera flag: scan method, default value 0 means rescanning all, 1 means only new added camera is scanned.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	CAM_SCAN("mvision") 'scan the camera
	CAM_COUNT(0) 'the number of cameras are saved into TABLE (0)
	IF (0 = TABLE(0)) THEN
	PRINT "No Camera"
	RETURN
	ENDIF
	CAM_SEL (0) 'select the camera of No.0

Related Instruction	CAM_SEL (select acquisition devices)

5.2.1.2. CAM_COUNT – Camera Numbers

Type	Image acquisition related.
Description	It is used to get the number of scanned cameras, the upper limit

	of the scan index, that is, the value range of scanId is [0,CAM_COUNT()-1]. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	CAM_COUNT(tabId) or num = CAM_COUNT() tabId: TABLE index, a kinds of output parameter, the number of cameras
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	CAM_COUNT (0) 'get the number of scanned cameras
Related Instruction	CAM_SCAN (scan the camera)

5.2.1.3. CAM_LISTLIB – Get Camera Library Type that are Loaded

Type	Image acquisition related.
Description	It is used to get the camera type that has been scanned and loaded, if it is successful, which means camera library plug-in is installed normally.
Grammar	CAM_LISTLIB(maxLen, tabId) or libs = CAM_LISTLIB() maxLen: the maximum length allowed tabId: output the camera type as a string and store it in the TABLE whose starting index number is tabId
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	CAM_LISTLIB(32,0) 'get the camera library type that has been loaded successfully
Related Instruction	CAM_SCAN (scan the camera)

5.2.1.4. CAM_FINDLIB – Get Available Camera Library Type

Type	Image acquisition related.
Description	It is used to get available camera library type, that is, find out all camera library types by file scanning method.

Grammar	CAM_FINDLIB(maxLen, tabId) maxLen: the maximum length allowed tabId: output the camera type as a string and store it in the TABLE whose starting index number is tabId
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	CAM_FINDLIB(64,0) 'get available camera library types
Related Instruction	CAM_SCAN (scan the camera)

5.2.1.5. CAM_QUERYLIB – Check Camera Library Information

<table>
<tr><td>Type</td><td>Image acquisition related.</td></tr>
<tr><td>Description</td><td>It is used to check camera library information, the returned result is multiple character strings, parting through “;”.</td></tr>
<tr><td>Grammar</td><td>Function syntax:
infos = CAM_QUERYLIB(camType,queryType)
camType: the camera type string to be queried, such as "zmvcbase" , "mindvision" , "basler" , "huaray" , "mvision" , "zmotion" , "dvpcamera"
queryType: query type, specifically related to the camera library, may report unsupported errors, as shown in the following table:
<table>
<tr><td>Type</td><td>Description</td><td></td></tr>
<tr><td>0</td><td>All parameters</td><td rowspan="3">In the case of supporting multiple cameras, it is the parameter of the Gigabit camera</td></tr>
<tr><td>1</td><td>Read-only parameters</td></tr>
<tr><td>2</td><td>Write-only parameters</td></tr>
<tr><td>3</td><td>In execution status</td><td></td></tr>
<tr><td>4</td><td>Error information</td><td>Details of the last time error reporting</td></tr>
<tr><td>5</td><td>USB3 camera parameters</td><td rowspan="2">If camera doesn't support USB3, error of unsupported will</td></tr>
<tr><td>6</td><td>USB3 camera read-only parameters</td></tr>
</table>
</td></tr>
</table>

7	USB3 camera write-only parameters	report.
8	Interface version No.	
9	Modification version No.	
10	The number of opened cameras	

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	? CAM_QUERYLIB("zmotion",4)　　‘print error information
Related Instruction	CAM_SCAN (scan the camera)

5.2.2. Camera Using

5.2.2.1. CAM_SEL – Select Acquisition Devices

Type	Image acquisition related.
Description	It is used to select the camera scanId as the serial No. of the currently operating camera, and the camera will automatically open. When there are multiple tasks and each task is responsible for one camera, such as two or more cameras, use CAM_SEL in each task to select the camera for following acquisition.
Grammar	CAM_SEL(scanId) 　　scanId: camera serial number when scanning Function syntax: scanId= CAM_SEL(), get the camera selected by the current task
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	CAM_SCAN("mvision")　　‘scan the camera CAM_COUNT(0)　　‘the number of cameras is saved into TABLE (0) IF (0 = TABLE(0)) THEN 　　PRINT "No Camera"

	RETURN ENDIF CAM_SEL(0) 'select the camera of No.0
Related Instruction	CAM_SCAN (scan the camera)

5.2.2.2. CAM_GETINFO – Camera Information

Type	Image acquisition related.
Description	It is used to obtain the information of the scanned camera, such as manufacturer name, SN number, IP, MAC, user-defined name (UserID), device type (GIGE), etc. Different cameras are with different parameters.
Grammar	CAM_GETINFO(prop,maxLen,tabId) prop: enumeration value of property to be obtained, please refer to below form: Value — Description -1 — Scanning type of camera 0 — SN 1 — Model 2 — Device ID 3 — Device name 4 — Display name 5 — Interface type 6 — Port No. 7 — Mac address 8 — IP address 9 — Host IP address 10 — Name defined by user 11 — Parameter defined by user maxLen: the maximum length allowed for obtained result, including terminator tabId: TABLE index, starting position of obtained character string property value
Controller	It is valid in controllers that support ZV function or they belong

	to 5XX series or above.
Example	DIM tmp (32) 'define variable CAM_SCAN() 'scan the camera CAM_COUNT(0) 'the number of cameras is saved into TABLE (0) FOR i = 0 TO TABLE(0)-1 CAM_SEL(i) 'select the camera of No.i CAM_GETINFO (0, 16, 0) 'get SN No. of camera, and save it into TABLE (0) DMCPY tmp(0), TABLE(0), 32 IF 0 = STRCOMP(tmp, "22533411") THEN EXIT FOR ENDIF NEXT
Related Instruction	CAM_SCAN (scan the camera)

5.2.2.3. CAM_GRAB – Grab One Frame

Type	Image acquisition related.
Description	The current camera index captures one frame, and the corresponding trigger mode is the capture mode (-1). When using the CAM_GRAB command to capture images, it is not necessary to call the start capture command CAM_START before this command, and the CAM_GRAB command will automatically open the "start capture command CAM_START". In order to avoid using the last image or to reduce the error checking processing, it is recommended to clear or zero the image before acquisition through ZV_CLEAR(img) or ZV_IMGSETCONST(img,0)
Grammar	CAM_GRAM (img) img: ZVOBJECT type, sampled image
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV0BJECT img

	CAM_SCAN("basler") CAM_COUNT(0)
	IF (0 = TABLE(0)) THEN
	        PRINT "No Camera"
	        RETURN
	ENDIF
	CAM_SEL(0)
	CAM_GRAB(img)

Related Instruction	CAM_SCAN (scan the camera)

5.2.2.4. CAM_SETMODE – Set Trigger Mode

Type	Image acquisition related.
Description	It is used to set camera triggering mode. While setting, it may stop camera acquisition actively.
Grammar	CAM_SETMODE(mode) mode: trigger mode: -1 is the acquisition mode; 0 is the soft trigger mode; 1~N is the external trigger mode, corresponding to different trigger sources of external trigger in turn, and they are relative to the external camera wiring. For example, 1 corresponds to the first trigger source Line0, 2 corresponds to the second trigger source Line1, and 3 corresponds to Line2. Please note that the trigger source of the external trigger mode needs to be used in conjunction with the "LineMode" parameter, see the camera parameter for details. When the camera uses the external trigger mode, the camera cable needs to be correctly connected to the external device. For the connection method of the external line "Line0", "Line1", "Line2" and the external device, please refer to the IO cable connection instructions of the corresponding camera, such as Hikvision camera Please refer to the "Hikvision Camera IO Cable Wiring Instructions" document.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	CAM_SETMODE(0) 'set camera as soft trigger mode

	/ CAM_SETMODE(1) 'set camera as external trigger mode, and the trigger source is Line0

5.2.2.5. CAM_TRIGGER – Camera Soft Trigger

Type	Image acquisition related.
Description	This is the soft trigger signal. Send it to trigger acquisition in the camera's soft trigger mode. The function is similar to CAM_SETPARAM("TriggerSoftware", 0), but there is additional processing inside the command to enhance protection, and the execution time will be longer It is recommended to use CAM_TRIGGER(), which can provide better protection.
Grammar	CAM_TRIGGER ()
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT img
	CAM_SCAN("basler") 'scan basler camera
	CAM_COUNT(0) 'get the number of scanned cameras
	IF (0 = TABLE(0)) THEN
	PRINT "No Camera"
	RETURN
	ENDIF
	CAM_SEL(0) 'select camera of No.0
	CAM_SETMODE(0) 'set trigger mode as soft trigger mode
	CAM_START(1) 'open camera acquisition, and assign the cache number as 1
	CAM_TRIGGER() 'use soft trigger command to trigger camera shooting, and trigger once, take picture once
	CAM_GET(img,0) 'get the image of assigned id No.0 from camera cache into "img"

5.2.2.6. CAM_STRAT – Start to Capture

Type	Image acquisition related.
Description	It is used to start the acquisition of the camera. The parameter is the number of buffers. After the acquisition starts, the images will be placed in the buffer in turn, and some parameters of the camera will not be able to be modified. This command is usually used when the camera acquires images in soft trigger mode or hard trigger mode. Please note it needs to use this command to start camera acquisition firstly, usually specifying the number of buffers as 1, and multiple buffers are usually used for multiple triggers in a short period of time, the number of buffers is equal to the number of triggers, and the triggered images are placed in the buffer in turn, and then processed separately.
Grammar	CAM_START (bufCnt) bufCnt: specified buffer numbers
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	CAM_START(1) 'start to capture, the buffer number is 1 / CAM_START(2) 'start to capture, the buffer number is 2
Related Instruction	CAM_STOP (stop acquisition)

5.2.2.7. CAM_STOP – Stop Acquisition

Type	Image acquisition related.
Description	It is used to stop acquisition.
Grammar	CAM_STOP ()
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	CAM_START() 'stop capturing
Related Instruction	CAM_START (start acquisition)

5.2.2.8. CAM_GET – Get the Image

Type	Image acquisition related.
Description	It is used to read the image from the specified buffer of the camera. The corresponding trigger mode is soft trigger mode (0) or external trigger mode (1~N). Before using the CAM_GET command to acquire the image, call the start acquisition command "CAM_START". Since the trigger mode is multi-threaded processing, in order to avoid image confusion, the triggered image can only be read once, and the cached image will be cleared after reading. If the cached image has not been read, the image captured by the next trigger will overwrite the old image, but it should be noted that the old image may still be read by using CAM_GET before the acquisition is completed, that is, the image read after the trigger is the last acquired image, then reading the image after each triggered to avoid this problem. In order to avoid using the last image or to reduce the error checking processing, it is recommended to clear or zero the image before acquisition through ZV_CLEAR(img) or ZV_IMGSETCONST(img,0).
Grammar	CAM_GET(img，bufId,timeout=-1) img: ZVOBJECT type, obtained image bufId: specified buffer id, starting from No.0 timeout: timeout time, ≥0, default parameter -1 uses system timeout parameter.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	**Example 1: obtain one single image through soft trigger** ZVOBJECT img CAM_SCAN("basler") 'scan basler camera CAM_COUNT(0) 'obtained the number of scanned camera IF (0 = TABLE (0)) THEN 'if the scanned camera number is 0, then return PRINT "No Camera"

	RETURN
	ENDIF
	CAM_SEL(0) 'select camera of No.0
	CAM_SETMODE(0) 'set trigger mode as soft trigger mode
	CAM_START(1) 'open camera acquisition, and assign the buffer number as 1
	CAM_SETPARAM ("TriggerSoftawre", 0)
	'use soft trigger command to trigger camera shooting, and trigger once, take picture once
	CAM_GET(img,0) 'get the image of assigned id No.0 from camera cache into "img"
	
	Example 2: achieve continuous acquisition through soft trigger
	ZVOBJECT img
	CAM_SCAN("basler") 'scan basler camera
	CAM_COUNT(0) 'obtained the number of scanned camera
	IF (0 = TABLE (0)) THEN
	'if the scanned camera number is 0, then return
	PRINT "No Camera"
	RETURN
	ENDIF
	CAM_SEL(0) 'select camera of No.0
	CAM_SETMODE(0) 'set trigger mode as soft trigger mode
	CAM_START(1) 'open camera acquisition, and assign the buffer number as 1
	WHILE (1)
	CAM_SETPARAM ("TriggerSoftawre", 0)
	'use soft trigger command to trigger camera shooting, and trigger once, take picture once
	CAM_GET(img,0) 'get the image of assigned id No.0 from camera cache into "img"
	WEND
	
	Example 3: single acquisition through external trigger mode (move_op mode)

	ZVOBJECT img
	CAM_SCAN("basler") 'scan basler camera
	CAM_COUNT(0) 'obtained the number of scanned camera
	IF (0 = TABLE (0)) THEN
	'if the scanned camera number is 0, then return
	PRINT "No Camera"
	RETURN
	ENDIF
	CAM_SEL(0) 'select camera of No.0
	CAM_SETMODE(1) 'set trigger mode as external trigger
	CAM_START(1) 'open camera acquisition, and assign the buffer number as 1
	AXIS_ZSET(0)=2 'select precision output function
	Base(0) 'select axis 0
	MOVEABS(100) 'move to machine coordinate 100
	MOVE_OP(0,ON)
	MOVE_OP(0,OFF) 'falling edge shooting, operate OUT (OP), trigger to take the picture
	MOVEABS(200) 'move to machine coordinate 200
	CAM_GET(img,0) 'get the image of assigned id No.0 from camera cache into "img"
	CAM_STOP() 'stop acquisition
	
	Example 4: single acquisition through external trigger (hw mode)
	ZVOBJECT img
	CAM_SCAN("basler") 'scan basler camera
	CAM_COUNT(0) 'obtained the number of scanned camera
	IF (0 = TABLE (0)) THEN
	'if the scanned camera number is 0, then return
	PRINT "No Camera"
	RETURN
	ENDIF
	CAM_SEL(0) 'select camera of No.0
	CAM_SETMODE(1) 'set trigger mode as external trigger

	CAM_START(1) 'open camera acquisition, and assign the buffer number as 1 AXIS_ZSET(0)=2 'select precision output function Table (100, 99, 100, 101) 'trigger OP to invert the machine coordinates HW_PSWITCH (2) Hw_pswitch (1, 0, 0, 100, 102, 1) 'operate OP0 to invert the position Base(0) 'select axis 0 MOVEABS(200) 'axis 0 moves to machine coordinate 200, here, camera is triggered to take the picture by falling edge, that is, at position of 100-101, image is saved into buffer 0 automatically. CAM_GET(img,0) 'get the image of assigned id No.0 from camera cache into "img" CAM_STOP() 'stop acquisition

5.2.3. Camera Parameters

Camera parameters include camera-related functions such as exposure time, packet sending delay, etc., which can be obtained or set through the CAM_GETPARAM and CAM_SETPARAM commands.

Camera parameters are divided into 7 types of parameters, including Boolean parameters, enumeration parameters, command parameters, string parameters, integer parameters, floating point parameters, and register parameters.

The controller supports Hikvision (mvision), Basler, mindvision, an Dahuang (huaray). Different cameras have their own camera parameters. For details, please refer to Appendix II, which lists the commonly used function parameters of the four cameras.

5.2.3.1. CAM_GETEXPOSURE – Get Exposure Time

Type	Image acquisition related.

Description	It is used to get current camera's exposure time, the unit is us. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	CAM_GETEXPOSURE(tabId) or number = CAM_GETEXPOSURE() tabId: TABLE index, get the camera exposure time
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	CAM_GETEXPOSURE 'save exposure time into TABLE (0)
Related Instruction	CAM_SETEXPOSURE (set exposure time)

5.2.3.2. CAM_SETEXPOSURE – Set Exposure Time

Type	Image acquisition related.
Description	It is used to set the exposure time of the current camera, the unit is us, the exposure time controls the exposure time of the photosensitive element of the camera chip to the light, the greater the exposure time, the brighter the image
Grammar	CAM_SETEXPOSURE(time) time: camera exposure time
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	CAM_SETEXPOSURE (3000) 'set current camera's exposure time as 3000us
Related Instruction	CAM_GETEXPOSURE (get exposure time)

5.2.3.3. CAM_GETPARAM – Get Parameters

Type	Image acquisition related.
Description	It is used to get camera's parameter values or expansion information, the result is in string. And it supports extended syntax, that is, add a suffix after the name such as paramName: suffix, please refer to Appendix II

	for details of parameter names
Grammar	CAM_GETPARAM(paramName, maxLen, tabId) paramName: the parameter name is a string, which can be input directly or the extended form of the parameter name maxLen: the maximum length allowed for tabId tabId: the TABLE starting index where the read parameter value information is placed paramName is explained as follows: ➢ **paramName** -- the normal form of the parameter name, get the value corresponding to the parameter name ➢ **paramName:Range** -- the expanded form of the parameter name with Range as the suffix, get the value range of the parameter. And the Range suffix only supports integer, floating point, Boolean, enumeration and other parameters. For integer and floating-point parameters, the values are separated by ":" colons. For example, if the parameter has a step size limit, it will output the form "minValue:maxValue:step", and if there is no step size limit, it will output "minValue:maxValue ". For boolean parameter values, it is separated by "," commas, such as "0,1". For enumeration parameter values, it is also separated by "," commas, such as "enum1, enum2, enum3,..." this form ➢ **paramName:Num** -- the extended form of the parameter name with Num as the suffix, the Num suffix only supports enumeration type parameters, and obtains the number of enumeration values. ➢ **paramName:Len** -- the extended form of the parameter name with Len as the suffix, the Len suffix only supports string type parameters, get the length of the output string.

<table>
<tr><td></td><td>

➢ **paramName:Str** -- the extended form of the parameter name with Str as the suffix, the Str suffix only supports enumeration type parameters, get the enumeration symbol
➢ **paramName:Type** -- the extended form of the parameter name with Type as the suffix. The Type suffix supports all parameter types, and the data type of the parameter is obtained, as shown in the following table

<table>
<tr><td>Type value</td><td>Description</td></tr>
<tr><td>-1</td><td>Unsupported or error</td></tr>
<tr><td>0</td><td>Boolean type</td></tr>
<tr><td>1</td><td>Integer type</td></tr>
<tr><td>2</td><td>Floating type</td></tr>
<tr><td>3</td><td>Character string type</td></tr>
<tr><td>4</td><td>Enumeration type</td></tr>
<tr><td>5</td><td>Command type</td></tr>
<tr><td>6</td><td>Register type</td></tr>
</table>

</td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><td>Example</td><td>

Example 1:
'get the value of the integer parameter "Width", that is, the width of the image
DIM tmp(32)
CAM_GETPARAM("Width", 32, 0)   'If the value is 1280
DMCPY tmp(0), TABLE(0), 32
?"Image width = "tmp

Example 2:
'get the number of enumeration values of the enumeration parameter "TriggerMode"
DIM tmp(32)
CAM_GETPARAM("TriggerMode:Num", 20, 0) 'If the value is 2
DMCPY tmp(0), TABLE(0), 32
?"Trigger Mode = "tmp

</td></tr>
</table>

	Example 3: 'get the range of the floating point parameter " ExposureTime " DIM tmp(32) CAM_GETPARAM("ExposureTime:Range", 32, 0) 'If the value is 1:1000000 DMCPY tmp(0), TABLE(0), 32 ?"Exposure Time = "tmp
Related Instruction	CAM_SETPARAM (set parameters)

5.2.3.4. CAM_SETPARAM – Set Parameters

Type	Image acquisition related.
Description	It is used to set camera's parameter values, and values are string. And it supports extended syntax, that is, add a suffix after the name such as paramName: suffix, please refer to Appendix II for details of parameter names
Grammar	CAM_SETPARAM(paramName, value) paramName: parameter name, the parameter name is a string value: parameter values, in string paramName is explained as follows: ➢ **paramName** -- the normal form of the parameter name, get the value corresponding to the parameter name ➢ **paramName:Str** -- the extended form of the parameter name with Str as the suffix, the Str suffix only supports enumeration type parameters, set parameters in the format of enumeration symbol
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	**Example 1:**

	CAM_SETPARAM("GevSCPD", "4000") 'set the camera sending delay, indirectly set the frame rate **Example 2:** CAM_SETPARAM("PixelFormat", "17301505") 'set the camera pixel format to grayscale format by way of enumeration value **Example 3:** CAM_SETPARAM("PixelFormat:Str", "Mono8") 'set the camera pixel format to grayscale format by means of enumeration symbols
Related Instruction	CAM_GETPARAM (get parameters)

5.2.3.5. CAM_GETPARAMTYPE – Get Parameters Types

Type	Image acquisition related.
Description	it is used to get the type of camera parameters. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	CAM_GETPARAMTYPE(paramName,tabId) or type = CAM_GETPARAMTYPE() paramName: camera parameter name tabId: the TABLE index, getting the camera parameters' type
Controller	It is valid in controllers that support ZV function or they belong

Type value	Description
-1	Unsupported or error
0	Boolean type
1	Integer type
2	Floating type
3	Character string type
4	Enumeration type
5	Command type
6	Register type

	to 5XX series or above.
Example	CAM_GETPARAMTYPE("zmotion",0) 'get parameters' type

5.2.3.6. CAM_GETPARAMMODE – Get Parameters Access Mode

<table>
<tr><td>Type</td><td>Image acquisition related.</td></tr>
<tr><td>Description</td><td>it is used to get the access mode of camera parameters.
Online command function is supported, using parameters that don't need to pass in TABLE index.</td></tr>
<tr><td>Grammar</td><td>CAM_GETPARAMMODE(paramName,tabId)
or mode = CAM_GETPARAMMODE()
paramName: camera parameter name
tabId: the TABLE index, getting the camera parameters' access mode
<table>
<tr><td>Mode value</td><td>Description</td></tr>
<tr><td>-1</td><td>Unsupported or error</td></tr>
<tr><td>0</td><td>Failure</td></tr>
<tr><td>1</td><td>Writing & Reading</td></tr>
<tr><td>2</td><td>Read only</td></tr>
<tr><td>3</td><td>Write only, usually is command</td></tr>
<tr><td>4</td><td>Writing & Reading, but it is read-only when in acquisition state</td></tr>
</table>
</td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><td>Example</td><td>CAM_GETPARAMMODE("zmotion",0)
'get parameters' access mode</td></tr>
</table>

5.2.3.7. CAM_LOADCONFIG – Load Configured Parameters/Files

Type	Image acquisition related.
Description	It is used to load the camera's default parameters and built-in parameters, or load parameters from a file. The instruction parameter config_data indicates the source of the parameters.

	If it is empty, default parameters are loaded. The string at the beginning of the colon identifies the loaded user configuration set UserSet, such as ":UserSet1"; otherwise a file name is recognized, and the corresponding file parameters are loaded. If the camera has a lot of parameters to configure, it can save all the parameters in a file, and directly use this command to load the file, which can simplify the configuration code of the camera
Grammar	CAM_LOADCONFIG(configData) configData: character string, loaded is parameter configuration information
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	CAM_LOADCONFIG(":UserSet1")
Related Instruction	CAM_GETPARAM (get parameters)

5.2.3.8. CAM_SAVECONFIG – Save Configured Parameters/Files

Type	Image acquisition related.
Description	It is used to save the built-in parameters or parameter files of the camera, and the command "config_data" indicates the source of the parameters: The character string beginning with a colon is recognized as the saved user configuration set UserSet, such as ":UserSet1", which cannot be the default set 0; otherwise, it is recognized as the file name, and saves the parameters to the corresponding file.
Grammar	CAM_SAVECONFIG(configData) configData: character string, save parameter configuration into target position
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	CAM_SAVECONFIG(":UserSet1")

5.2.3.9. CAM_DATASIZE – Image Data Size

Type	Image acquisition related.
Description	It is used to get how size the camera image data occupies.
Grammar	CAM_DATASIZE(tabId) or size = CAM_DATASIZE() tabId: TABLE index, obtained image data size
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	CAM_DATASIZE(0) 'store the memory size of the camera image data in the array table(0)

5.3. Image Showing

5.3.1. Latch

5.3.1.1. ZV_LATCH – Latch Showing Image

Type	Latch
Description	According to the current latch information, convert the image img to be displayed to the latch area specified by the latch channel number latch_id, and the latch information is updated. The latch information may have changes such as legality checks and range restrictions. A latch refers to a protected buffer area, only single-channel and three-channel are supported, 8u type. Note: due to the limitation of the display mode, the display command cannot be called frequently and continuously, otherwise the latter command may be lost, that is, the command triggers the protection mechanism without updating the display and returns directly. Display commands include ZV_LATCH, ZV_LATCHTRANS and

	ZV_LATCHCLEAR.
Grammar	ZV_LATCH (img, latchId, type) img: ZVOBJECT image type, image to be shown latchId: latch channel No., which corresponds to the digital behind variable @ZV that is quoted by background picture of control, such as, channel 0 corresponds to @ZV0 type: showing method, 0 – keep current scaling ratio and translate, 1 – showing in the center, that is, the max size that can show
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img,"logo.png",0) 'read image in original image format ZV_LATCH(img,0) 'show image "img" that relates to channel 0 
Related Instruction	ZV_READIMAGE (read image)

5.3.1.2. ZV_LATCHINFO – Get Latch Information

Type	Latch
Description	It is used to get latch information specified by latch channel No., they are in the sequence of scale ratio, x offset, y offset, window width, window height, background color.
Grammar	Command syntax:

	ZV_LATCHINFO(latchId, tabId) latchId: latch channel number tabId: TABLE index, latch information, occupying 6 data spaces ZV_LATCHINFO(latchId, tabNum, tabId) latchId: latch channel No. tabNum: the amount of space available for the tabId tabId: TABLE index, latch information, output zoom ratio, x offset, y offset, window width, window height, background color, image width, image height, frame width, frame height in sequence according to the number of tab_num Function syntax: ?ZV_LATCHINFO(latchId) Output the above abbreviated parameter names and parameter values in readable text.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img,"logo.png",0) 'read image in original image format ZV_LATCH(img,0) 'display picture ZV_LATCHINFO(0,0) 'save the data of latch channel 0 from talbe(0) to store information sequentially ?TABLE(0) 'zoom ratio ?TABLE(1) 'x offset ?TABLE(2) 'y offset ?TABLE(3) 'window width ?TABLE(4) 'window height ?TABLE(5) 'background color
Related Instruction	ZV_READIMAGE (read image)

5.3.1.3. ZV_LATCHRANS – Latch Image Transformation

Type	Latch
Description	Perform scaling and translation of x, y coordinates on the currently displayed image. The process is to scale the sclaeFactor on the basis of the current image. When zooming, the image coordinate position corresponding to the center point of the screen remains unchanged, and then translate the tx and ty pixels on the zoomed image, when translating, the image boundary can be translated to the center of the field of view at most. For example, when panning to the lower right, the upper left corner of the image can be translated to the center of the field of view at most. Note: due to the limitation of the display mode, the display command cannot be called frequently and continuously, otherwise the latter command may be lost, that is, the command triggers the protection mechanism without updating the display and returns directly. Display commands include ZV_LATCH, ZV_LATCHTRANS and ZV_LATCHCLEAR.
Grammar	ZV_LATCHTRANS(latchId, sclaeFactor, tx, ty) latchId: latch channel number sclaeFactor: the zoom ratio on the previous basis, if the ratio can be 0, the display will be scaled according to the size of the window tx: the number of pixels to translate in the x direction. If it is greater than 0, it moves in the positive direction of x. The size count is based on the zoomed image ty: the number of pixels to translate in the y direction. If it is greater than 0, it moves in the positive direction of y. The size count is based on the zoomed image
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img

ZV_LATCHSETSIZE(0,400,112)   'size of display area

ZV_READIMAGE(img,"logo.png",0)   'read image in the format of original image

ZV_LATCH(img,0)

'button actions

ZV_LATCHTRANS(0,0.8,10,10)   'zoom out by 0.8 times on the basis of the displayed image, and then translate in the positive direction of the xy axis (bottom right) equivalent to a distance of 10 pixels of the zoomed image

Original image:

After transformed:

(zoom out by a factor of 0.8, translate by 10 pixels in the x direction, then translate by 10 pixels in the y direction)

5.3.1.4. ZV_LATCHCLEAR – Latch Data Clearing

Type	Latch
Description	It is used to clear all other parameters except latch background and image size, and uses the background to fill in latch area. Note: due to the limitation of the display mode, the display command cannot be called frequently and continuously, otherwise the latter command may be lost, that is, the command triggers the protection mechanism without updating the display and returns directly. Display commands include ZV_LATCH, ZV_LATCHTRANS and ZV_LATCHCLEAR.
Grammar	ZV_LATCHCLEAR (latchId) latchId: latch channel number
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img,"logo.png",0) 'read the image ZV_LATCH(img,0) 'show the image in latch channel 0 ZV_LATCHCLEAR(0) 'clear latch channel 0

5.3.1.5. ZV_LATCHSETSIZE – Set Latch Image Size

Type	Latch
Description	It is used set the size of the latch display image, that is, the size of the indicator. If the setting is wrong, the image will still be displayed according to the size of the display control, but it will cause the calculation error of the latch information, and then the coordinate conversion the interactive control will be wrong. And it is enabled when the latched image is changed next time, that is, the display command needs to be called to trigger the update. Display commands include ZV_LATCH, ZV_LATCHTRANS and ZV_LATCHCLEAR.

Grammar	ZV_LATCHSETSIZE(latchId,width,height) latchId: latch channel No. width: width of latch area height: height of latch area
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_LATCHSETSIZE(0,400,200) 'set width of height of channel 0: 400*200 ZV_READIMAGE(img,"1.bmp",0) 'read image in original image format ZV_LATCH(img,0) 'show the image in latch 0
Related Instruction	ZV_READIMAGE (read image)

5.3.1.6. ZV_LATCHSETBGC – Set Latch Background Color

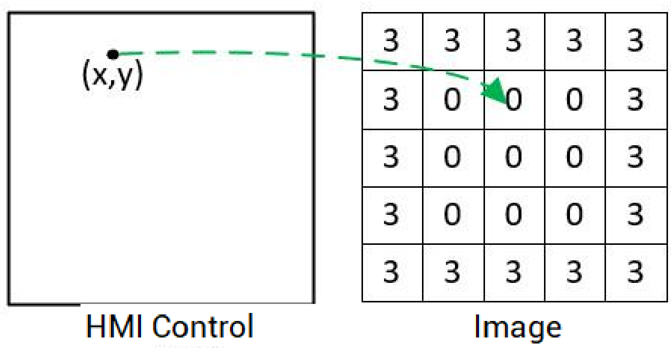
Type	Latch
Description	It is used set the background color of the latched image, which will be enabled when the latched image is changed next time, that is, the display command needs to be called to trigger the update Display commands include ZV_LATCH, ZV_LATCHTRANS and ZV_LATCHCLEAR.
Grammar	ZV_LATCHSETBGC(latchId,rgb) latchId: latch channel No. rgb: background color, color value generated by ZV_COLOR
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM red ZVOBJECT img Red = ZV_COLOR(255,0,0) ZV_READIMAGE(img,"logo.png",0) 'read image information in original image format ZV_LATCHSETBGC(0,red)

	'change latch channel 0 background color as red ZV_LATCH(img,0) Original image:  Color changed: 
Related Instruction	ZV_COLOR (set color)

5.3.2.Coordinates Conversion of Image & HMI

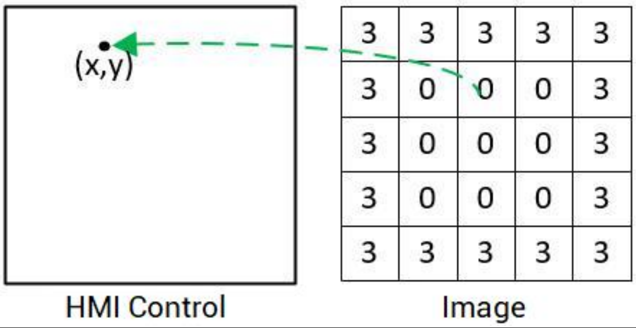
5.3.2.1. ZV_POSTOIMG – From HMI Control to Image Coordinate

Type	Coordinates conversion
Description	Convert HMI control coordinates to image pixel coordinates. In the conversion process, it uses latch information, which needs to be correct for coordinate transformation during interface

	interaction. The instruction is applied to the coordinate data, please note that it is different from the length data 
Grammar	ZV_POSTOIMG(latchId,num,tabIdIn,tabIdOut) latchId: latch channel No. num: coordinates numbers tabIdIn: store the TABLE index of the coordinate points before conversion, the data of num coordinate points are x, y, x, y... tabIdOut: store the TABLE index of the transformed coordinate point
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	TABLE(0, 0, 1) ZV_POSTOIMG(0,1,0,10) 'convert the control coordinates corresponding to latch channel 0 into image coordinates

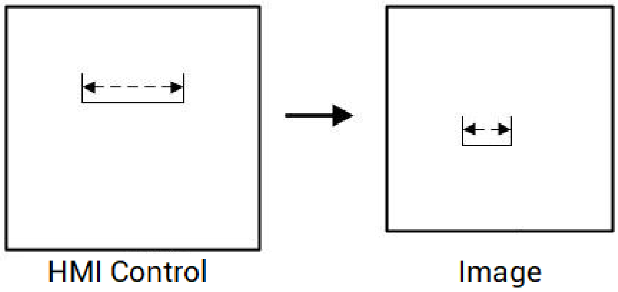
5.3.2.2. ZV_POSFROMIMG – From Image to HMI Control Coordinates

Type	Coordinates conversion
Description	Convert image pixel coordinates to HMI control coordinate. In the conversion process, it uses latch information, which needs to be correct for coordinate transformation during interface interaction.

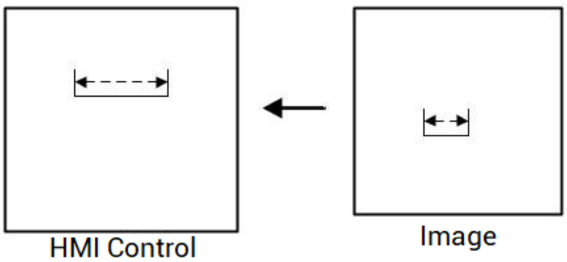
	The instruction is applied to the coordinate data, please note that it is different from the length data 
Grammar	ZV_POSFROMIMG(latchId,num,tabIdIn,tabIdOut) latchId: latch channel No. num: coordinates numbers tabIdIn: store the TABLE index of the coordinate points before conversion, the data of num coordinate points are x, y, x, y... tabIdOut: store the TABLE index of the transformed coordinate point
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	TABLE(0, 0, 1) ZV_POSFROMIMG(0,1,0,3) 'convert the image coordinates corresponding to latch channel 0 into HMI control coordinates.

5.3.2.3. ZV_LENTOIMG – From HMI Control to Image Length

Type	Coordinates conversion
Description	Convert length of HMI control to image pixel length. In the conversion process, it uses latch information, which needs to be correct for relative value transformation during interface interaction. The instruction is applied to the length data, only for zooming, please note that it is different from the coordinates data.

	
Grammar	imgLen = ZV_LENTOIMG(latchId,len) latchId: latch channel number len: the length in the coordinate system of the HMI control return value: imgLen: the length in the image coordinate system after conversion
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	imgLen= ZV_LENTOIMG(0,20) 'convert the length in control coordinates corresponding to latch channel 0 to the length in image coordinates

5.3.2.4. ZV_LENFROMIMG – From Image to HMI Control Length

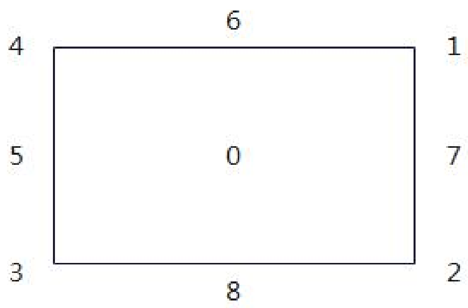
Type	Coordinates conversion
Description	Convert image pixel length to length of HMI control. In the conversion process, it uses latch information, which needs to be correct for relative value transformation during interface interaction. The instruction is applied to the length data, only for zooming, please note that it is different from the coordinates data. 

Grammar	imgLen = ZV_LENFROMIMG(latchId,len) latchId: latch channel number len: the length in the coordinate system of the image return value: hmiLen: the length in the HMI coordinate system
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	hmiLen =ZV_LENFROMIMG(0,10) 'convert the length in the image coordinate system to the length in the control coordinates corresponding to latch channel 0

5.3.3.HMI

5.3.3.1. ZV_HMIADJRECT – Adjust Rectangle ROI

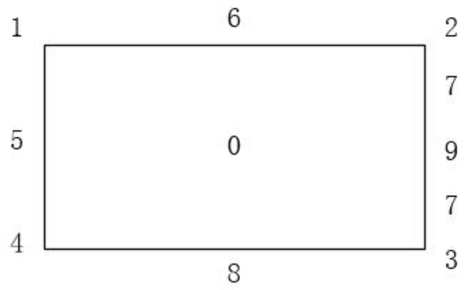
Type	HMI
Description	It is used for HMI control interaction to adjust the rectangle ROI, that is, the rectangular roi parallel to the horizontal axis. It is used in the refresh function of the HMI custom control to real-time adjust the position and size of the rectangular roi by the mouse. When the mouse is pressed, the adjustment function is different according to the mouse position in the roi area. And this area is the corresponding hit area with adjustment function, the rectangle roi contains 9 hit areas (numbered 0-8, corresponding to center, upper left point, upper right point, lower right point, lower left point, left, upper, right, lower) ➢ The center hit area is used to adjust the position. ➢ The four corner hit areas are used to adjust the two sides corresponding to each corner ➢ The four-side hit area is used to adjust the corresponding side. The schematic diagram of the numbering area hit by a regular

	rectangle is as follows: 
Grammar	hittype=ZV_HMIADJRECT (mousex, mousey, tabId, hitType, maxHitDist =0) mousex: the mouse x coordinate of the HMI control mousey: the mouse y coordinate of the HMI control tabId: save the TABLE index of the roi parameter of the rectangle, which are ltx, lty, rbx, rby in sequence, that is, the coordinates ltx, lty of the upper left corner of the rectangle, and the coordinates of rbx, rby of the lower right corner of the rectangle, corresponding to the values in the coordinate system of the hmi control, and the adjusted value will directly replace the unadjusted value hitType: specify the No. of the hit area, indicating the corresponding part of the rectangle to be adjusted by the command. When it is -1, it means an invalid No., and the command will judge the hit situation by itself. If it is a valid No., adjust the corresponding part of the rectangle. Normally, -1 is passed in when the left mouse button is pressed, and the command internally judges the hit position and returns. If the return value is -1, there is no adjustment action, and the return value is the effective area No., then in the subsequent mouse movement process, it uses this return value as the new one passed parameter and continues to call this command for continuous adjustment. maxHitDist: select the maximum distance of the "hit area", it is only selected if it is less than this distance, and it is not limited by the distance when it is 0 Return value:

	hittype: when a valid hit No. is passed in, the No. is returned. When an invalid hit number is passed in, calculate the hit area No. according to the mouse clicking position and return, -1 is returned if there is no hit area.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	hittype = ZV_HMIADJRECT(mousex,mousey,tabId,-1) 'get the hit area No. corresponding to the mouse click position ZV_HMIADJRECT(mousex,mousey,tabId,hittype) 'specify the hit area No. and adjust the corresponding rectangular part

5.3.3.2. ZV_HMIADJRECT2 – Adjust Rotate Rectangle ROI

Type	HMI
Description	It is used for HMI control interaction to adjust the rotate rectangle ROI. Specifically, it is used in the refresh function of the HMI custom control to real-time adjust the position and size of the rotate rectangular roi by the mouse. When the mouse is pressed, the adjustment function is different according to the mouse position in the roi area. And this area is the corresponding hit area with adjustment function, the rotate rectangle roi contains 10 hit areas (numbered 0-9, corresponding to center, upper left point, upper right point, lower right point, lower left point, left, upper, right, lower, near right center, namely, the angle hit area). ➢ The center hit area is used to adjust the position. ➢ The four corner hit areas are used to adjust the two sides corresponding to each corner. ➢ The four-side hit area is used to adjust the corresponding side (current side and corresponding side are adjusted). ➢ The angle hit area is used to adjust rotate angle.

	Note: rotate rectangle angle based on image coordinates system, clockwise is positive, the unit is degree. The schematic diagram of the numbering area hit by a rotate regular rectangle is as follows: 
Grammar	hittype=ZV_HMIADJRECT2 (mousex, mousey, tabId, hitType, maxHitDist =0) mousex: the mouse x coordinate of the HMI control mousey: the mouse y coordinate of the HMI control tabId: save the TABLE index of the roi parameter of the rotate rectangle, which are cx, cy, width, height, angle in sequence, that is, the rotate rectangle's center coordinates cx, cy, width, height, angle, corresponding to the values in the coordinate system of the hmi control, and the adjusted value will directly replace the unadjusted value. hitType: specify the No. of the hit area, indicating the corresponding part of the rectangle to be adjusted by the command. When it is -1, it means an invalid No., and the command will judge the hit situation by itself. If it is a valid No., adjust the corresponding part of the rectangle. Normally, -1 is passed in when the left mouse button is pressed, and the command internally judges the hit position and returns. If the return value is -1, there is no adjustment action, and the return value is the effective area No., then in the subsequent mouse movement process, it uses this return value as the new one passed parameter and continues to call this command for continuous adjustment. maxHitDist: select the maximum distance of the "hit area", it is only selected if it is less than this distance, and it is not

	limited by the distance when it is 0 Return value: hittype: when a valid hit No. is passed in, the No. is returned. When an invalid hit number is passed in, calculate the hit area No. according to the mouse clicking position and return, -1 is returned if there is no hit area.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	hittype = ZV_HMIADJRECT2(mousex,mousey,tabId,-1) 'get the hit area No. corresponding to the mouse click position ZV_HMIADJRECT2(mousex,mousey,tabId,hittype) 'specify the hit area No. and adjust the corresponding rectangular part

5.3.3.3. ZV_HMIADJRECT2S – Adjust Rotate Rectangle ROI (Single Side)

Type	HMI
Description	It is used for HMI control interaction to adjust the rotate rectangle ROI. Specifically, it is used in the refresh function of the HMI custom control to real-time adjust the position and size of the rotate rectangular roi by the mouse. When the mouse is pressed, the adjustment function is different according to the mouse position in the roi area. And this area is the corresponding hit area with adjustment function, the rotate rectangle roi contains 10 hit areas (numbered 0-9, corresponding to center, upper left point, upper right point, lower right point, lower left point, left, upper, right, lower, near right center, namely, the angle hit area). ➢ The center hit area is used to adjust the position. ➢ The four corner hit areas are used to adjust the two sides corresponding to each corner.

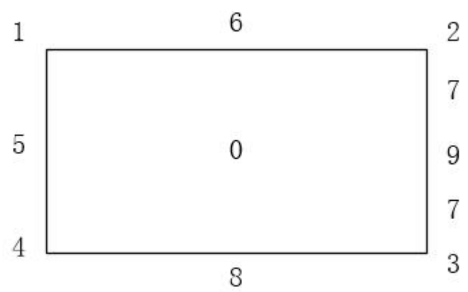
<table>
<tr>
<td></td>
<td>

- The four-side hit area is used to adjust the corresponding side.
- The angle hit area is used to adjust rotate angle.

The edge hit point only adjusts the hit edge, the opposite edge is unchanged.

Note: rotate rectangle angle based on image coordinates system, clockwise is positive, the unit is degree.

The schematic diagram of the numbering area hit by a rotate regular rectangle is as follows:



</td>
</tr>
<tr>
<td>Grammar</td>
<td>

hittype=ZV_HMIADJRECT2S (mousex, mousey, tabId, hitType, maxHitDist =0)

mousex: the mouse x coordinate of the HMI control

mousey: the mouse y coordinate of the HMI control

tabId: save the TABLE index of the roi parameter of the rotate rectangle, which are cx, cy, width, height, angle in sequence, that is, the rotate rectangle's center coordinates cx, cy, width, height, angle, corresponding to the values in the coordinate system of the hmi control, and the adjusted value will directly replace the unadjusted value.

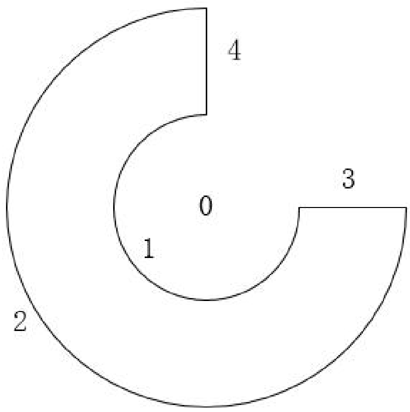
hitType: specify the No. of the hit area, indicating the corresponding part of the rectangle to be adjusted by the command. When it is -1, it means an invalid No., and the command will judge the hit situation by itself. If it is a valid No., adjust the corresponding part of the rectangle. Normally, -1 is passed in when the left mouse button is pressed, and the command internally judges the hit position and returns. If the return value is -1, there is no adjustment action, and the return

</td>
</tr>
</table>

	value is the effective area No., then in the subsequent mouse movement process, it uses this return value as the new one passed parameter and continues to call this command for continuous adjustment. maxHitDist: select the maximum distance of the "hit area", it is only selected if it is less than this distance, and it is not limited by the distance when it is 0 Return value: hittype: when a valid hit No. is passed in, the No. is returned. When an invalid hit number is passed in, calculate the hit area No. according to the mouse clicking position and return, -1 is returned if there is no hit area.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	hittype = ZV_HMIADJRECT2S(mousex,mousey,tabId,-1) 'get the hit area No. corresponding to the mouse click position ZV_HMIADJRECT2S(mousex,mousey,tabId,hittype) 'specify the hit area No. and adjust the corresponding rectangular part

5.3.3.4. ZV_HMIADJARC – Adjust Arc ROI

Type	HMI
Description	It is used for HMI control interaction to adjust the arc ROI. Specifically, it is used in the refresh function of the HMI custom control to real-time adjust the position and size of the arc ROI by the mouse. When the mouse is pressed, the adjustment function is different according to the mouse position in the ROI area. And this area is the corresponding hit area with adjustment function, the arc ROI contains 5 hit areas (numbered 0-4, corresponding to center, inner circle, outer circle, starting side, end side).

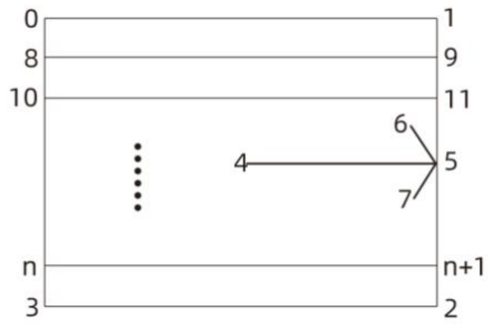
	➢ The center hit area is used to adjust the position. ➢ The inner circle hit area is used to adjust the size of the inner circle. ➢ The outer circle hit area is used to adjust the size of the outer circle. ➢ The starting side hit area is used to adjust starting angle of the circle. ➢ The end side hit area is used to adjust end angle of the circle. Note: arc starting angle is based on image coordinates system, clockwise is positive, the unit is degree. The schematic diagram of the numbering area hit by an arc is as follows: 
Grammar	hittype = ZV_HMIADJARC (mousex, mousey, tabId, hitType, maxHitDist = 0) mousex: the mouse x coordinate of the HMI control mousey: the mouse y coordinate of the HMI control tabId: save the TABLE index of the ROI parameter of the arc, which are cx, cy, radius, annR, angleStart, anglExtent in sequence, that is, the arc's center coordinates cx, cy, circle center arc, circle semi-width, starting angle, angle range, and the adjusted value will directly replace the unadjusted value. hitType: specify the No. of the hit area, indicating the corresponding part of the arc to be adjusted by the command. When it is -1, it means an invalid No., and the command will

	judge the hit situation by itself. If it is a valid No., adjust the corresponding part of the arc. Normally, -1 is passed in when the left mouse button is pressed, and the command internally judges the hit position and returns. If the return value is -1, there is no adjustment action, and the return value is the effective area No., then in the subsequent mouse movement process, it uses this return value as the new one passed parameter and continues to call this command for continuous adjustment. maxHitDist: select the maximum distance of the "hit area", it is only selected if it is less than this distance, and it is not limited by the distance when it is 0 Return value: hittype: when a valid hit No. is passed in, the No. is returned. When an invalid hit number is passed in, calculate the hit area No. according to the mouse clicking position and return, -1 is returned if there is no hit area.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	hittype = ZV_HMIADJARC(mousex,mousey,tabId,-1) 'get the hit area No. corresponding to the mouse click position ZV_HMIADJARC(mousex,mousey,tabId,hittype) 'specify the hit area No. and adjust the corresponding arc part.

5.3.3.5. ZV_HMIRECT2 – From Rotate Rectangle ROI to HMI Drawing Primitives

Type	HMI
Description	It is used to decompose the rotating rectangle ROI into HMI-supported drawing primitives and add control parameters for HMI drawing display.
Grammar	ZV_HMIRECT2(tabIdRect,tabIdOut[,maxElems=0])

	tabIdRect: the TABLE index that saves the parameters of the rotated rectangle, which are cx, cy, width, height, angle, subNum and subWidth in sequence, that is, the center coordinates of the rotated rectangle cx, cy, width, height, angle, number of sub-regions, and width of the sub-region, and these values are the values in the coordinate system of the hmi control, and the number of sub-regions can be 0 (default value of maxElems), if more than 80, the excess part will not be drawn. Usually, when drawing a general rotation rectangle, such as the ROI rectangle for creating a template and the ROI rectangle for generating a Region used in Blob analysis, subNum = 0 and subWidth = 0 are fine. When drawing the roi rectangle for measuring a straight line, sub_num and subWidth are the parameter values corresponding to the straight line measurement, please refer to the ZV_MRGENLINE command for parameter details. tabIdOut: the TABLE index of the primitive parameter, which are the corner coordinates of the four vertices of the rotating rectangle ROI, the start and end points of the arrow pointing from the center of the roi to the center of the right line, the coordinates of the line segments on both sides of the arrow, the number of sub-region segment lines, and the starting and ending coordinates of the segment line. And there is maximum and minimum output quantity limit, maximum 80 sub-areas, the excess part will not be output, the minimum needs to ensure the output of the parameters of the rotation rectangle and the indicator arrow, if there is no maxElems parameter, it is necessary to ensure enough space to receive the primitive parameter 8*(subNum- 1)+17 (subNum is greater than 1) maxElems: the available size of tabIdOut, the space occupied by the output parameters is ≤ to maxElems, and the output primitives are complete, the excess parts are not output tableOut output coordinates:

	
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	'construct a rotated rectangle centered at (100,100) with a width and height of 60 * 40 and an angle of 60 degrees. The number of sub-regions is 8, and the width of the sub-region is 5, and the graphic data is stored in the TABLE whose starting index is 0 TABLE(0,100,100,60,40,60,8,5) 'set the color of the drawn rectangle to blue SET_COLOR(RGB(0,0,255)) 'decompose the rotated rectangle into drawing primitives supported by hmi, and store the corresponding primitive coordinate data in the TABLE whose starting index is 300 ZV_HMIRECT2(0, 300) 'draw a rectangle DRAWLINE(TABLE(300), TABLE(301), TABLE(302), TABLE(303)) DRAWLINE(TABLE(302), TABLE(303), TABLE(304), TABLE(305)) DRAWLINE(TABLE(304), TABLE(305), TABLE(306), TABLE(307)) DRAWLINE(TABLE(306), TABLE(307), TABLE(300), TABLE(301)) 'draw an arrow from the center of the rectangle to the center of the right line DRAWLINE(TABLE(308), TABLE(309), TABLE(310), TABLE(311)) DRAWLINE(TABLE(312), TABLE(313), TABLE(310), TABLE(311)) DRAWLINE(TABLE(314), TABLE(315), TABLE(310), TABLE(311))

	'set the color of the drawn subregion line to green
	SET_COLOR(RGB(0,255,0))
	
	'If the number of sub-region dividing lines is greater than 0, draw sub-region segment lines
	IF TABLE(316) > 0 THEN
	DIM idx
	FOR idx = 0 TO TABLE(316)-1
	DRAWLINE (TABLE(317+idx*4), TABLE(318+idx*4), TABLE(319+idx*4), TABLE(320+idx*4))
	NEXT
	ENDIF

5.3.3.6. ZV_HMIARC – From Arc ROI to HMI Drawing Primitives

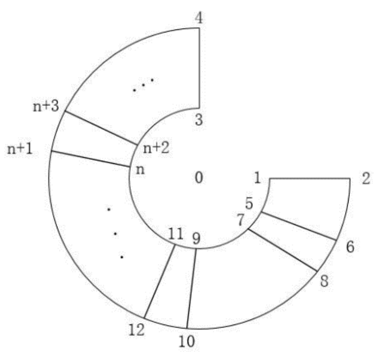
Type	HMI
Description	It is used to decompose the arc ROI into HMI-supported drawing primitives and add control parameters for HMI drawing display.
Grammar	ZV_HMIARC(tabIdArc,tabIdOut[,maxElems=0]) tabIdArc: the TABLE index that saves the parameters of the arc, which are cx, cy, radius, annR, angleStart, angleExtent, subNum and subWidth in sequence, that is, the center coordinates of the arc cx, cy, arc center radius, arc semi-width, starting angle, angle range, sub-region numbers, sub-region width, and these values are the values in the coordinate system of the hmi control, and the number of sub-regions can be 0, if more than 80, the excess part will not be drawn. Usually, when drawing a general arc, such as the ROI arc for generating a Region used in Blob analysis, subNum = 0 and subWidth = 0 are fine. When drawing the ROI arc for measuring a center, sub_num and subWidth are the parameter values corresponding to the arc measurement, please refer to the ZV_MRGENCIRCLE command for parameter details. tabIdOut: the TABLE index of the primitive parameters,

<table>
<tr>
<td></td>
<td>
which are the center of the arc, the inner and outer arc, the start and end angles, the number of arc edges, the start and end coordinates of the edges, the number of sub-region segment lines, and the start and end coordinates of the segment lines. Among them, the ring edge refers to the boundary line corresponding to the starting and ending angle of the ring.

- If it is a full circle, there will be 1 line.
- If it is a non-full circle, there will be two lines.
- If it is not needed, it will be 0 lines.
- When the edge line is 0, the dividing line must also be 0.
- The angle unit after conversion is radian, and there are maximum and minimum output quantity limits internally. The maximum number of sub-areas is 80, and the excess part is not output. The minimum parameter output of the ring and edge line needs to be guaranteed. If there is no maxElems parameter, it needs to ensure enough space to receive primitive parameters 8*(subNum-1)+20 (subNum is > 1)

maxElems: the available size of tabIdOut, the space occupied by the output parameters is ≤ to maxElems, and the output primitives are complete, the excess parts are not output

tableOut output coordinates:



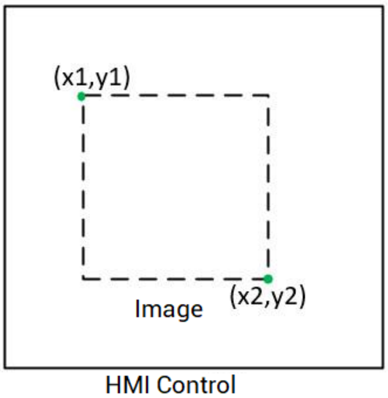
</td>
</tr>
<tr>
<td>Controller</td>
<td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td>
</tr>
<tr>
<td>Example</td>
<td>'construct an arc with a center at (100,100, a radius of 60, a half-width of 20, a starting angle of 0, and an angle range of 270, with</td>
</tr>
</table>

	8 sub-regions and a sub-region width of 5, and store the graphic data in a TABLE starting at index 0
	TABLE(0,100,100,60,20,0,270,8,5)
	
	'set the color of the drawn arc to blue
	SET_COLOR(RGB(0,0,255))
	'decompose the arc into the drawing primitives supported by hmi, and store the corresponding primitive coordinate data in the TABLE whose starting index is 300
	ZV_HMIARC(0,300)
	
	'draw the inner arc
	DRAWARC(TABLE(300), TABLE(301), TABLE(302), TABLE(304), TABLE(305))
	
	'draw the outer arc
	DRAWARC(TABLE(300), TABLE(301), TABLE(303), TABLE(304), TABLE(305))
	
	'draw a cross at the center of the circle
	DRAWLINE(TABLE(300),TABLE(301)-5,TABLE(300),TABLE(301)+5) DRAWLINE(TABLE(300)-5,TABLE(301),TABLE(300)+5,TABLE(301))
	
	'If the number of edges is greater than 0, draw the edges
	IF TABLE(306) > 0 THEN
	    DIM idx
	    FOR idx = 0 to TABLE(306)-1
	        DRAWLINE(TABLE(307+idx*4), TABLE(308+idx*4), TABLE(309+idx*4), TABLE(310+idx*4))
	    NEXT
	
	    'set the color of the drawn sub-region line to green
	    SET_COLOR(RGB(0,255,0))

	DIM startid 'sub-area dividing line startid = 307+TABLE(306)*4 FOR idx = 0 TO TABLE(startid)-1 DRAWLINE(TABLE(startid+1+idx*4), TABLE(startid+2+idx*4), TABLE(startid+3+idx*4), TABLE(startid+4+idx*4)) NEXT ENDIF

5.3.4. Custom Control Drawing

5.3.4.1. DRAWZVOBJ – HMI Custom Control Drawing

Type	HMI
Description	It is used to draw one image in the area specified by HMI custom control. Specifically, it is used in custom control drawing function. 
Grammar	DRAWZVOBJ (img, x1, y1, x2, y2) img: input image x1: the x coordinate of the upper left corner of the specified area y1: the y coordinate of the upper left corner of the specified area x2: the x coordinate of the lower right corner of the specified area, which can be omitted, and the default value is the X coordinate of the lower right corner of the control, that is, the

	width of the control -1 y2: the y coordinate of the lower right corner of the specified area, which can be omitted, and the default value is the Y coordinate of the lower right corner of the control, that is, the height of the control -1 The instruction zooms and matches the image to the area specified by x1, y1, x2, y2. The values of x1, y1, x2, and y2 can be beyond the range of the control. The instruction adapts the image according to the original value, and don't draw the exceed part of control after matching.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_READIMAGE(img,"logo.png",0) 'read image in original image format DRAWZVOBJ(img,0,0,639,479) 'draw an image img in the area specified by the upper left coordinates (0,0) and lower right coordinates (639,479) of the custom control, and the width and height directions of the image will be scaled and matched The lower right corner of the image can be drawn on the control as follows: GLOBAL SUB cust_draw() local width, height width = HMI_CONTROLSIZEX() height = HMI_CONTROLSIZEY() 'the left and upper sides of the image are beyond the scope of the control and will not be drawn 'the last two parameters can be omitted, and the default value is the correct coordinate DRAWZVOBJ(MAIN_IMG, -width, -height, width-1, height-1) END SUB

5.4. Basic Parameters

5.4.1. ZV_IMAGEFOR – Basic Information

Type	Basic parameters
Description	It is used to get image basic information.
Grammar	ZV_IMGINFO(img, tabId) img: ZVOBJECT type, source image tabId: TABLE index, the TABLE index that outputs position, image information, 5 pieces of data, which are width, height, channel number, data type and basic pixel size in turn. For the image data type, please refer to the ZV_IMGGENCONST command
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img,"1.bmp",0) 'get the image in original format ZV_IMGINFO(img,0) size = TABLE(0) * TABLE (1) * TABLE (2) * TABLE (4) PRINT "Image Data Size:" + tostr(size)
Related Instruction	ZV_READIMAGE (read image)

5.4.2. ZV_IMGISVALID – Whether the Image is Valid

Type	Basic parameters
Description	It is used to determine whether the image is valid. Online command function is supported, using parameters that don't need to pass in TABLE index. ➢ Divided into command syntax and function syntax ➢ Non-immediate instructions, for function syntax, some expressions are not supported now, and an error that non-immediate instructions are not supported will be printed at this time.

Grammar	Command syntax: ZV_IMGISVALID(img, tabId) img: ZVOBJECT type, source image tabId: TABLE index, whether the image is valid. 1 – valid, 0-invalid Function syntax: val = ZV_IMGISVALID (img)
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT img
	ZV_READIMAGE(img,"1.bmp",0) 'read the image in original image format
	IF ZV_IMGISVALID(img) = 0 THEN
	PRINT "Image Is Empty"
	ENDIF

Related Instruction	ZV_READIMAGE (read image)

5.4.3.ZV_IMGWIDTH – Get Image Width

Type	Basic parameters
Description	It is used to get image width. Online command function is supported, using parameters that don't need to pass in TABLE index. ➢ Divided into command syntax and function syntax ➢ Non-immediate instructions, for function syntax, some expressions are not supported now, and an error that non-immediate instructions are not supported will be printed at this time.
Grammar	Command syntax: ZV_IMGWIDTH(img, tabId) img: ZVOBJECT type, source image tabId: TABLE index, output results are saved into TABLE (tabId). Function syntax: val = ZV_IMGWIDTH (img)
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img

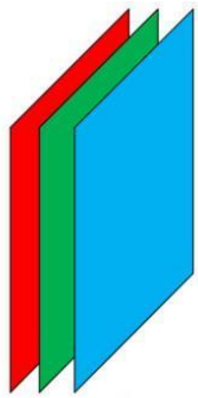
	ZV_READIMAGE(img,"1.bmp",0) ' read the image in original image format ZV_IMGWIDTH(mat,0) 'get image width, the result is saved into TABLE (0).

5.4.4.ZV_IMGHEIGHT – Get Image Height

Type	Basic parameters
Description	It is used to get image height. Online command function is supported, using parameters that don't need to pass in TABLE index. ➢ Divided into command syntax and function syntax ➢ Non-immediate instructions, for function syntax, some expressions are not supported now, and an error that non-immediate instructions are not supported will be printed at this time.
Grammar	Command syntax: ZV_IMGHEIGHT(img, tabId) img: ZVOBJECT type, source image tabId: TABLE index, output results are saved into TABLE (tabId). Function syntax: val = ZV_IMGHEIGHT (img)
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img,"1.bmp",0) ' read the image in original image format ZV_IMGHEIGHT(mat,0) 'get image height, the result is saved into TABLE (0).

5.4.5.ZV_IMGCNS – Get Image Channels

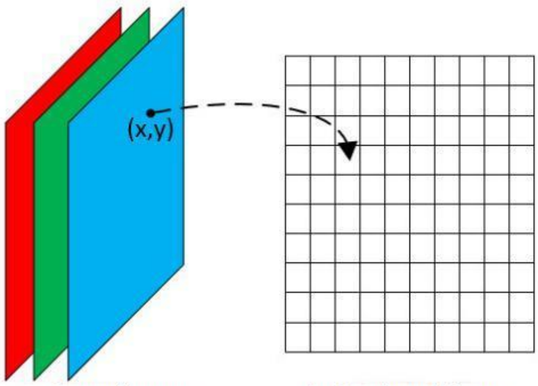
Type	Basic parameters
Description	It is used to get the number of image channels.

	Online command function is supported, using parameters that don't need to pass in TABLE index. ➢ Divided into command syntax and function syntax ➢ Non-immediate instructions, for function syntax, some expressions are not supported now, and an error that non-immediate instructions are not supported will be printed at this time.  3-channel image
Grammar	Command syntax: ZV_IMGCNS(img, tabId) img: ZVOBJECT type, source image tabId: TABLE index, output results are saved into TABLE (tabId). Function syntax: val = ZV_IMGCNS (img)
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img,"1.bmp",0) ' read the image in original image format ZV_IMGCNS(mat,0) 'get image channel numbers, the result is saved into TABLE (0).

5.5. Access

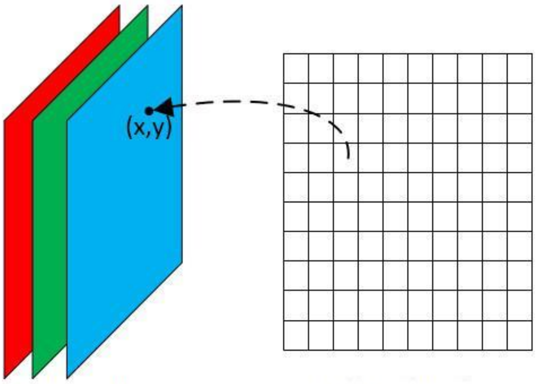
5.5.1. ZV_IMGGETVAL – Get the Value

Type	Access

Description	It is used to get the value of image specified value. Online command function is supported, using parameters that don't need to pass in TABLE index. 
Grammar	ZV_IMGGETVAL(img,x,y,cn,tabId) Or value = ZV_IMGGETVAL(img,x,y,cn) im: ZVOBJECT type, source image x: get the x-coordinate of the value y: get the y coordinate of the value cn: the specified channel No. tabId: TABLE index, output parameter, obtained value
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img,"1.bmp",0) ' read the image in original image format ZV_IMGGETVAL(img,x,y,0,0) 'get the value of image coordinates (x, y)
Related Instruction	ZV_IMASETVAL (modify value)

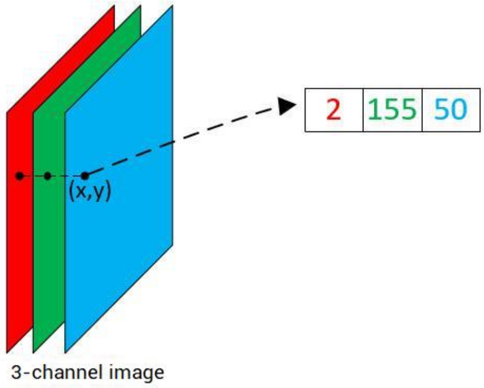
5.5.2.ZV_IMGSETVAL – Modify the Value

Type	Access
Description	It is used to modify the value of image specified value.

	
Grammar	ZV_IMGSETVAL(img,x,y,cn,value) im: ZVOBJECT type, source image x: modify the x-coordinate of the value y: modify the y coordinate of the value cn: modify the specified channel No. of the value value: the value after modification
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img,"1.bmp",0) ' read the image in original image format ZV_IMGGETVAL(img,0,0,0,100) 'modify channel 0 of image "img", the value of coordinate (0,0) is 100
Related Instruction	ZV_IMAGETVAL (get value)

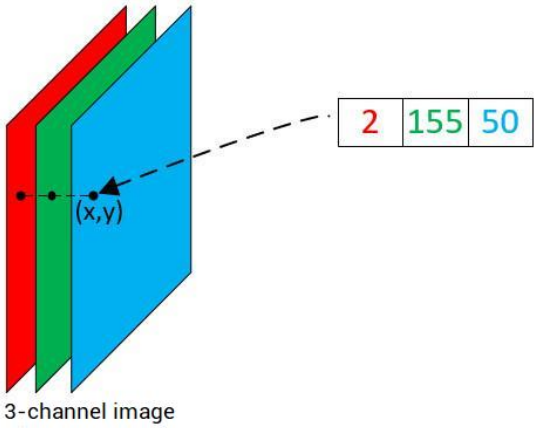
5.5.3.ZV_IMGGETELEM – Get Pixel Value

Type	Access
Description	It is used to get the pixel value of specified position, multi-channel is valid, and the tab_elememt length is 4.

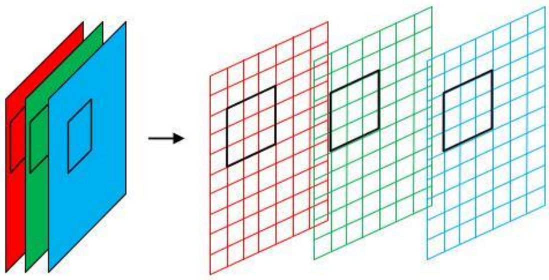
	
Grammar	ZV_IMGGETELEM(img,x,y,tabId) im: ZVOBJECT type, source image x: obtained pixel coordinate x y: obtained pixel coordinate y tabId: TABLE index, output parameter, obtained pixel value
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT show_img ZV_READIMAGE(img,"1.bmp",0) ' read the image in original image format ZV_IMGGETELEM(show_ima,100, 100, 0) 'get the pixel value of coordinates (100, 100) of image "show_img", and save them into TABLE (0)
Related Instruction	ZV_IMGSETELEM (modify pixel value)

5.5.4.ZV_IMGSETELEM – Modify the Pixel Value

Type	Access
Description	It is used to modify the pixel value of specified value, multi-channel is valid, and the tab_elememt length is 4.

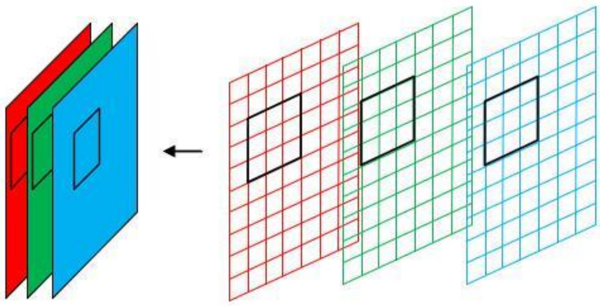
	
Grammar	ZV_IMGSETELEM(img,x,y,tabId) im: ZVOBJECT type, image to be modified x: modified pixel coordinate x y: modified pixel coordinate y tabId: TABLE index, each channel value after modification
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT show_img ZV_READIMAGE(img,"1.bmp",0) ' read the image in original image format ZV_IMGGETELEM(show_img,100,100,0) 'modify pixel value of image "show_img" coordinate (100,100) as each channel value of TABLE(0).
Related Instruction	ZV_IMGGETELEM (get pixel value)

5.5.5.ZV_IMGGETSUB – Get Sub-Region

Type	Access
Description	It is used to get the image sub-region, and it is recommended for the size of the subregion is the times of 4. 

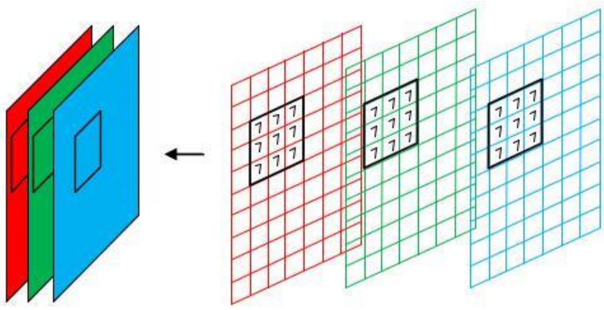
Grammar	ZV_IMGGETSUB(img,subImg,x,y,w,h) im: ZVOBJECT type, source image subImg: ZVOBJECT type, obtained subregion image x: obtained pixel coordinate x of subregion in source image y: obtained pixel coordinate y of subregion in source image w: obtained subregion width h: obtained subregion height
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src, dst ZV_READIMAGE(src,"1.bmp",0) 'read the image in original image format ZV_IMGGETSUB(src, dst, 0, 0, 100, 100) 'get the image with width pixel 100 and height pixel 100 based on original image coordinates 0,0
Related Instruction	ZV_IMGSETSUB (modify subregion)

5.5.6.ZV_IMGSETSUB – Modify the Sub-Region

Type	Access
Description	It is used to modify the image sub-region, and it is recommended for the size of the subregion is the times of 4. 
Grammar	ZV_IMGSETSUB(img, subImg, x, y) img: ZVOBJECT type, the big image to be modified subImg: ZVOBJECT type, the result image to be modified for subregion x: coordinate x of subregion to be modified y: coordinate y of subregion to be modified

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, dst ZV_READIMAGE(src, "1.bmp", 0) ' read the image in original image format ZV_IMGGETSUB(img, dst, x, y) 'modify the size of image subregion.
Related Instruction	ZV_IMGGETSUB (get subregion)

5.5.7.ZV_IMGSETCONST – Fill Constant Image

Type	Access
Description	Fill in the image by constant "val". 
Grammar	ZV_IMGSETCONST(img, val) img: ZVOBJECT type, the image to be filled val: the value is 0-255
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img, "1.bmp", 0) ' read the image in original image format ZV_IMGGETSUB(img,255) 'use constant 255 to fill in "img"

5.5.8.ZV_IMGCONVERT – Convert Specified Data Type

Type	Access
Description	It is used to convert image data types.

Grammar	ZV_IMGCONVERT(src, dst, type [,mult = 1, add = 0]) src: ZVOBJECT type, source image dst: ZVOBJECT type, converted image type: image data type after converted <table><tr><td>type</td><td>Description</td></tr><tr><td>0</td><td>8-bit without symbol 8U</td></tr><tr><td>1</td><td>16-bit without symbol 16U</td></tr><tr><td>2</td><td>32-bit with symbol 32S</td></tr><tr><td>3</td><td>64-bit with symbol 64F</td></tr></table>mult: value multiplier when converting add: value offset when converting
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src, dst ZV_READIMAGE(src, "1.bmp", 0) ' read the image in original image format ZV_IMGCONVERT (src, dst, 3, 1, 0) 'convert the image to 64F form

5.5.9.ZV_IMGCOPY – Copy

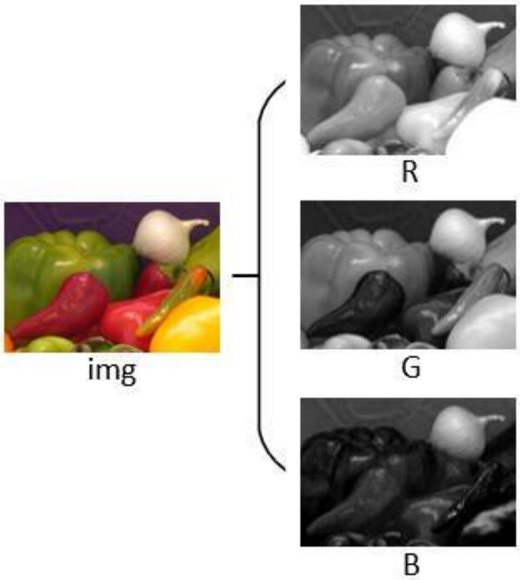
Type	Access
Description	It is used to copy images.
Grammar	ZV_IMGCOPY(src, dst) src: ZVOBJECT type, source image dst: ZVOBJECT type, copied image, if src and dst are the same one variable, instruction will return normally and directly.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src, dst ZV_READIMAGE(src, "test, jpg", 0) ' read the image in original image format ZV_IMGCOPY (src, dst) 'copy image in src variable into dst variable

5.5.10. ZV_IMGSPLIT2 – Split Dual-Channel

Type	Access
Description	It is used to split the dual-channel image into two independent channels.
Grammar	ZV_IMGSPLIT2(src, dst1, dst2) src: ZVOBJECT type, dual-channel image dst1: ZVOBJECT type, the first channel after decomposition dst2: ZVOBJECT type, the second channel after decomposition
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src, dst1, dst2 ZV_READIMAGE(src, "test, jpg", 0) ' read the image in original image format ZV_IMGSPLIT2 (src, dst1, dst2) 'split dual-channel image "src" into dst1 and dst2
Related Instruction	ZV_IMGMERGE2

5.5.11. ZV_IMGSPLIT3 – Split Three-Channel

Type	Access
Description	It is used to split the three-channel image into three independent channels.
Grammar	ZV_IMGSPLIT3(src, dst1, dst2, dst3) src: ZVOBJECT type, source three-channel image dst1: ZVOBJECT type, the first channel after decomposition dst2: ZVOBJECT type, the second channel after decomposition dst3: ZVOBJECT type, the third channel after decomposition
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	 ZVOBJECT src, r, g, b ZV_READIMAGE(src, “test, jpg”, 0) ’ read the image in original image format ZV_IMGSPLIT3 (src_img, r, g, b) ‘split original three-channel image into r, g, b three independent channels.
Related Instruction	ZV_IMGERGE3

5.5.12. ZV_IMGSPLIT4 – Split Four-Channel

Type	Access
Description	It is used to split the four-channel image into four independent channels.
Grammar	ZV_IMGSPLIT4(src, dst1, dst2, dst3, dst4) src: ZVOBJECT type, source four-channel image dst1: ZVOBJECT type, the first channel after decomposition dst2: ZVOBJECT type, the second channel after decomposition dst3: ZVOBJECT type, the third channel after decomposition dst4: ZVOBJECT type, the fourth channel after decomposition
Controller	It is valid in controllers that support ZV function or they belong

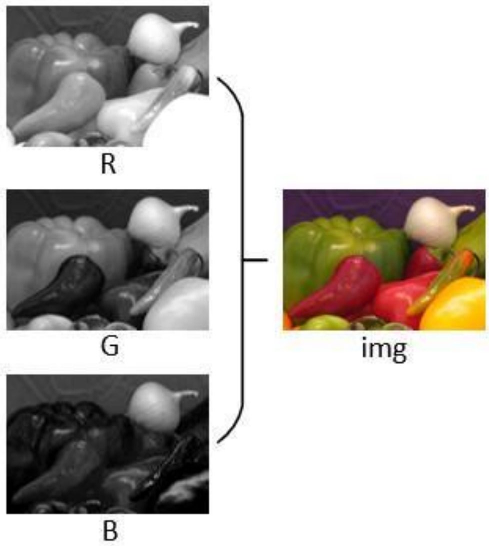
	to 5XX series or above.
Example	ZVOBJECT src, dst1, dst2, dst3, dst4 ZV_READIMAGE(src, "test, jpg", 0) ' read the image in original image format ZV_IMGSPLIT4 (src, dst1, dst2, dst3, dst4) 'split four-channel image into 4 independent channels.
Related Instruction	ZV_IMGERGE4

5.5.13. ZV_IMGMERGE2 – Merge Dual-Channel

Type	Access
Description	It is used to merge two single-channel images into a dual-channel image.
Grammar	ZV_IMGMERGE2(src1, src2, dst) src1: ZVOBJECT type, the first single-channel image src2: ZVOBJECT type, the second single-channel image dst: ZVOBJECT type, image after merged
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src1, src2, dst ZV_READIMAGE(src1, "test1, jpg", 0) ' read the image in original image format ZV_READIMAGE(src2, "test2, jpg", 0) ' read the image in original image format ZV_IMGMERGE2 (src1, src2, dst) 'merge into one dual-channel
Related Instruction	ZV_IMGSPLIT2

5.5.14. ZV_IMGMERGE3 – Merge Three-Channel

Type	Access
Description	It is used to merge three single-channel images into a 3-channel image.

Grammar	ZV_IMGMERGE2(src1, src2, src3, dst) src1: ZVOBJECT type, the first single-channel image src2: ZVOBJECT type, the second single-channel image src3: ZVOBJECT type, the third single-channel image dst: ZVOBJECT type, image after merged
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src1, src2, src3, dst ZV_READIMAGE(src1, "test1, jpg", 0) 'read the image in original image format ZV_READIMAGE(src2, "test2, jpg", 0) 'read the image in original image format ZV_READIMAGE(src3, "test3, jpg", 0) 'read the image in original image format ZV_IMGMERGE3 (src1, src2, src3, dst) 'merge into one 3-channel
Related Instruction	ZV_IMGSPLIT3

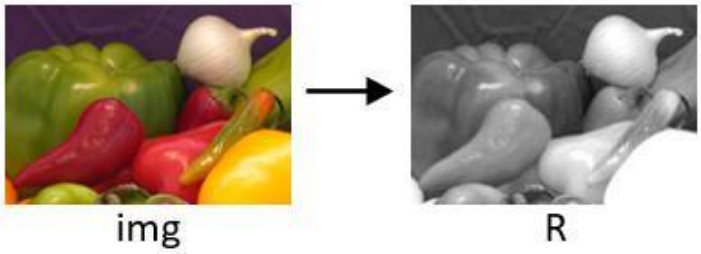
5.5.15. ZV_IMGMERGE4 – Merge Four-Channel

Type	Access
Description	It is used to merge two single-channel images into a dual-channel image.

Grammar	ZV_IMGMERGE2(src1, src2, src3, src4, dst) src1: ZVOBJECT type, the first single-channel image src2: ZVOBJECT type, the second single-channel image src3: ZVOBJECT type, the third single-channel image src4: ZVOBJECT type, the fourth single-channel image dst: ZVOBJECT type, image after merged
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src1, src2, src3, src4, dst ZV_READIMAGE(src1, "test1, jpg", 0) ' read the image in original image format ZV_READIMAGE(src2, "test2, jpg", 0) ' read the image in original image format ZV_READIMAGE(src3, "test3, jpg", 0) ' read the image in original image format ZV_READIMAGE(src4, "test4, jpg", 0) ' read the image in original image format ZV_IMGMERGE4 (src1, src2, src3, src4, dst) 'merge into one 4-channel
Related Instruction	ZV_IMGSPLIT4

5.5.16. ZV_IMGGETCN – Get Image in Specified Channel

Type	Access
Description	It is used to get the image that is in specified channel.
Grammar	ZV_IMGGETCN(src, dst, cn) src: ZVOBJECT type, input image dst: ZVOBJECT type, obtained channel image cn: channel No., ≥0, and it is smaller than channels of src itself
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	 ZVOBJECT src, R ZV_READIMAGE(src1, “test, jpg”, 0) ' read the image in original image format ZV_IMGGETCN(src, R, 0) ‘get image of channel 0

5.6. Operation

5.6.1. Algebra

5.6.1.1. ZV_SCALE -- Grayscale Extension

Type	Algebra
Description	It is used to remap grayscale of matrix or image, dst = src * mult + add. For images, when the pixel value of the target image dst is greater than 255, it takes 255, and when the pixel value is less than 0, it takes 0
Grammar	ZV_SCALE(src, dst, mult, add) src: ZVOBJECT type, source image or matrix dst: ZVOBJECT type, modified image or matrix, the same type of src mult: transformation scale factor, floating point value add: the offset of the transformation, adjust the grayscale
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0)'read the image in the original image format ZV_MINMAXLOC(src,0) 'output to TABLE(0) image src minimum

	value, minimum value x, y coordinates, maximum value, maximum value x, y coordinates mult=255/(TABLE(3)-TABLE(0)) add=-mult*TABLE(0) ZV_SCALE(src,dst,mult,add) 'remap grayscale to [0,255]

5.6.1.2. ZV_ABSDIFF -- Absolute Difference

Type	Algebra
Description	Absolute difference value of 2 images or matrixes.
Grammar	ZV_ABSDIFF(src1, src2, dst, mult) src1: ZVOBJECT type, image or matrix 1 src2: ZVOBJECT type, image or matrix 2 (size, channels and data type must be same as src 1) dst: ZVOBJECT type, calculated image or matrix, the same type of src1 mult: multiplier when calculating
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src1, src2, dst ZV_READIMAGE(src1,"test1.jpg",0) 'read the image in the original image format ZV_READIMAGE(src2,"test2.jpg",0) 'read the image in the original image format ZV_ABSDIFF (src1, src2, dst, 0.5) 'output calculated image into dst, calculation formula is dst = | src1 – src2 | * mult

5.6.1.3. ZV_ADDWEIGHTED -- Weighted Sum

Type	Algebra
Description	It is used to find the weighted sum of images or matrices element by element, dst = weight1*src1 + weight2*src2 + add
Grammar	ZV_ADDWEIGHTED(src1,src2,dst,weight1,weight2,add)

	src1: ZVOBJECT type, image or matrix 1 src2: ZVOBJECT type, image or matrix 2 (size, channels and data type must be same as src 1) dst: ZVOBJECT type, calculated image or matrix, the same type of src1 weight1: the weight of 1 weight2: the weight of 2 add: bias term
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src1, src2, dst ZV_READIMAGE(src1,"test1.jpg",0) 'read the image in the original image format ZV_READIMAGE(src2,"test2.jpg",0) 'read the image in the original image format ZV_ADDWEIGHT (src1, src2, dst, 0.8, 1-0.8, 0) 'image merging

5.6.1.4. ZV_MUL -- Multiple

Type	Algebra
Description	It is used to multiple two images or matrices element by element, dst = src1*src2*mult+add
Grammar	ZV_MUL (src1,src2,dst,mult,add) src1: ZVOBJECT type, image or matrix 1 src2: ZVOBJECT type, image or matrix 2 (size, channels and data type must be same as src 1) dst: ZVOBJECT type, calculated image or matrix, the same type of src1 mult: multiple when calculating add: bias when calculating
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT src1, src2, dst ZV_READIMAGE(src1,"test1.jpg",0) 'read the image in the original image format ZV_READIMAGE(src2,"test2.jpg",0) 'read the image in the original image format ZV_MUL (src1, src2, dst, 0.5, 2) 'dst = src1*src2*mult+add, two images or matrices are multiplied element by element

5.6.1.5. ZV_DIV -- Divide

Type	Algebra
Description	It is used to divide two images or matrices element by element, dst = src1*src2*mult+add
Grammar	ZV_DIV (src1,src2,dst,mult,add) src1: ZVOBJECT type, image or matrix 1 src2: ZVOBJECT type, image or matrix 2 (size, channels and data type must be same as src 1) dst: ZVOBJECT type, calculated image or matrix, the same type of src1 mult: multiple when calculating add: bias when calculating
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src1, src2, dst ZV_READIMAGE(src1,"test1.jpg",0) 'read the image in the original image format ZV_READIMAGE(src2,"test2.jpg",0) 'read the image in the original image format ZV_DIV (src1, src2, dst, 0.5, 2) 'dst = src1/src2*mult+ add

5.6.1.6. ZV_MAX – Maximum Value

Type	Algebra

Description	It is used to get the maximum of two images or matrices element by element.
Grammar	ZV_MAX (src1,src2,dst) src1: ZVOBJECT type, image or matrix 1 src2: ZVOBJECT type, image or matrix 2 (size, channels and data type must be same as src 1) dst: ZVOBJECT type, resulting image or matrix, it is composed of the larger value in src1 and src2, and the type, size, channels are same as src1.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src1, src2, dst ZV_READIMAGE(src1,"test1.jpg",0) 'read the image in the original image format ZV_READIMAGE(src2,"test2.jpg",0) 'read the image in the original image format ZV_DIV (src1, src2, dst) 'get the bigger one between 2 images to make the dst image

5.6.1.7. ZV_MIN – Minimal Value

Type	Algebra
Description	It is used to get the minimal of two images or matrices element by element.
Grammar	ZV_MIN (src1,src2,dst) src1: ZVOBJECT type, image or matrix 1 src2: ZVOBJECT type, image or matrix 2 (size, channels and data type must be same as src 1) dst: ZVOBJECT type, resulting image or matrix, it is composed of the smaller value in src1 and src2, and the type, size, channels are same as src1.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT src1, src2, dst ZV_READIMAGE(src1,"test1.jpg",0) 'read the image in the original image format ZV_READIMAGE(src2,"test2.jpg",0) 'read the image in the original image format ZV_MIN (src1, src2, dst) 'get the smaller one between 2 images to make the dst image

5.6.1.8. ZV_COMPARE – Comparison

Type	Algebra
Description	It is used to compare the size of src1 and src2, and output a binary image. If the logical condition is met, it is a white pixel, otherwise it is a black pixel
Grammar	ZV_COMPARE (src1,src2,dst,op) src1: ZVOBJECT type, image or matrix 1 src2: ZVOBJECT type, image or matrix 2 (size, channels and data type must be same as src 1) dst: ZVOBJECT type, result of comparison, 8U image type op: operator of comparison
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src1, src2, dst ZV_READIMAGE(src1,"test1.jpg",0) 'read the image in the original image format ZV_READIMAGE(src2,"test2.jpg",0) 'read the image in the

Operator table (within Grammar):

Value of comparison operator	Description
0	=
1	>
2	≥
3	<
4	≤
5	≠

	original image format ZV_COMPARE (src1, src2, dst, 1) 'compare two images values and output one binary image

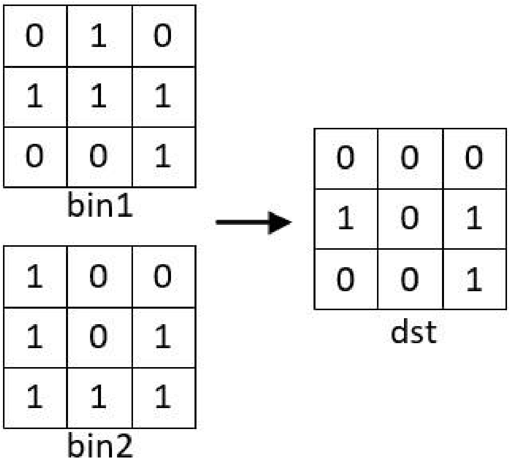
5.6.1.9. ZV_NORM – Norm

Type	Algebra
Description	It is used to calculate the norm of type. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_NORM (src, type, tab_norm) / number = ZV_NORM (src, type) src: ZVOBJECT type, image or matrix type: type of norm <table><tr><td>Type of norm</td><td>Description</td></tr><tr><td>0</td><td>Infinity norm — maximum absolute value</td></tr><tr><td>1</td><td>1-norm - sum of absolute values</td></tr><tr><td>2</td><td>2-norm – the root of the sum of squares</td></tr></table>If the input type is out of range, return 0 directly tab_norm: TABLE index, norm of matrix or image
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original image format ZV_COMPARE (src, 1, 0) 'output norm of matrix to TABLE (0)

5.6.2. Image Logic Operation

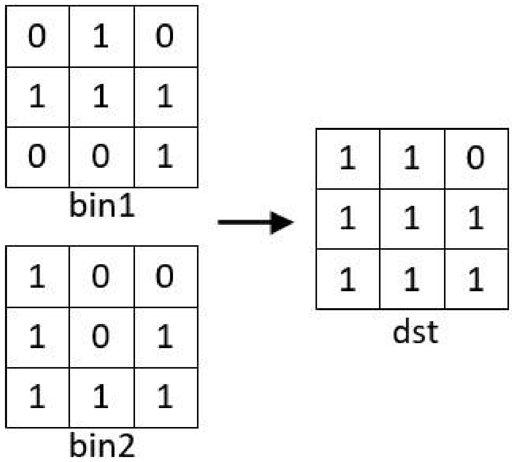
5.6.2.1. ZV_AND – Bitwise And

Type	Image logic operation
Description	It is used to calculate "and" image of image src1 and image src2.

Grammar	ZV_AND (src1, src2, dst) src1: ZVOBJECT type, single-channel image src2: ZVOBJECT type, single-channel image, the size, channel numbers, data type must be same as src1 dst: ZVOBJECT type, result image of bitwise AND per element of src1 and src2
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT bin1, bin2, dst TABLE(0, 0, 1, 0, 1, 1, 1, 0, 0, 1)'store data into TABLE(0) ZV_IMGGENCONST(bin1,3,3,1,0,0) TABLE(20, 1, 0, 0, 1, 0, 1, 1, 1, 1)'store data into TABLE(0) ZV_IMGGENCONST(bin2,3,3,1,0,20) ZV_AND(bin1,bin2,dst) 'find the common part of two binary images

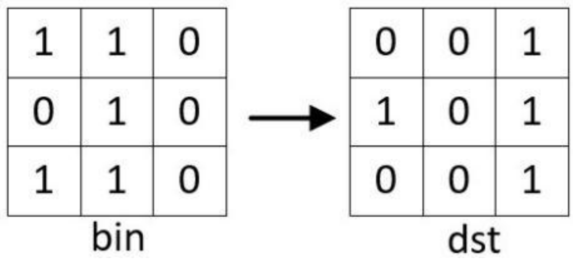
5.6.2.2. ZV_OR – Bitwise Or

Type	Image logic operation
Description	It is used to calculate "or" image of image src1 and image src2.

Grammar	ZV_OR (src1, src2, dst) src1: ZVOBJECT type, single-channel image src2: ZVOBJECT type, single-channel image, the size, channel numbers, data type must be same as src1 dst: ZVOBJECT type, result image of bitwise OR per element of src1 and src2
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT bin1, bin2, dst TABLE(0, 0, 1, 0, 1, 1, 1, 0, 0, 1)'store data into TABLE(0) ZV_IMGGENCONST(bin1,3,3,1,0,0) TABLE(20, 1, 0, 0, 1, 0, 1, 1, 1, 1)'store data into TABLE(0) ZV_IMGGENCONST(bin2,3,3,1,0,20) ZV_OR(bin1,bin2,dst) 'the dst image is the result image obtained by bitwise OR each element of the src1 and src2 images

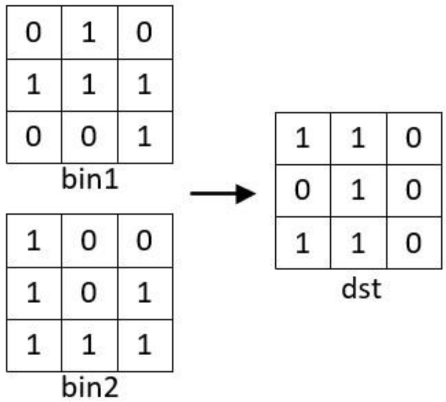
5.6.2.3. ZV_NOT – Bitwise Not

Type	Image logic operation
Description	It is used to calculate the bitwise inverse of image src.

Grammar	ZV_NOT (src, dst) src1: ZVOBJECT type, single-channel image dst: ZVOBJECT type, the result image after bitwise inversion of each element of src
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT bin dst TABLE(0, 0, 1, 0, 1, 1, 1, 0, 0, 1)'store data into TABLE(0) ZV_IMGGENCONST(bin,3,3,1,0,0) ZV_NOT (bin, dst) 'image bitwise inversion

5.6.2.4. ZV_XOR – Bitwise Exclusive OR

Type	Image logic operation
Description	It is used to calculate the exclusive or image of image src1 and image src2.
Grammar	ZV_XOR (src1, src2, dst) src1: ZVOBJECT type, single-channel image dst: ZVOBJECT type, the result image after bitwise inversion of each element of src dst: ZVOBJECT type, result image of each element of XOR src1 and src2
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	 ZVOBJECT bin1, bin2, dst TABLE(0, 0, 1, 0, 1, 1, 1, 0, 0, 1)'store data into TABLE(0) ZV_IMGGENCONST(bin1,3,3,1,0,0) TABLE(20, 1, 0, 0, 1, 0, 1, 1, 1, 1)'store data into TABLE(0) ZV_IMGGENCONST(bin2,3,3,1,0,20) ZV_XOR(bin1, bin2,dst) 'bitwise XOR image of images src1 and src2

5.6.3. Statistics

5.6.3.1. ZV_NONZEROCOUNT – Non 0 Element Numbers

Type	Statistics
Description	It is used to count the number of non-zero elements in src. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_NONZEROCOUNT(src,tabId) or count = ZV_NONZEROCOUNT(src) src: ZVOBJECT type, image or matrix tabId: TABLE index, the number of src non-zero elements
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img TABLE(0, 0, 1, 0, 1, 1, 1, 0, 0, 1) 'save data into TABLE (0) ZV_IMGGENCONST(img,3,3,1,0,0) ZV_NONZEROCOUNT(img,0) 'count the number of non-0

	elements in src type information, and save the counted result into TABLE (0).

5.6.3.2. ZV_SUM – Sum for Elements

Type	Statistics
Description	It is used to sum in all elements of each independent channel.
Grammar	ZV_SUM(src,tabId) src: ZVOBJECT type, image or matrix tabId: TABLE index, sum each channel's all elements
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT rgb ZV_READIMAGE(rgb,"test.jpg",0) 'read image in the original format ZV_SUM(rgb, 0) 'r channel sum value is TABLE(0), g channel sum value is TABLE(1), b channel sum value is TABLE(2)

5.6.3.3. ZV_STATROW – Row Element Statistic

Type	Statistics
Description	It is used to count each row's elements, and calculate the counted value.
Grammar	ZV_STATROW(src,dst,type) src: ZVOBJECT type, image or matrix of single-channel tabId: TABLE index, matrix, counted result, row N, column 1 type: type of statistic, 0 – sum, 1- average, 2 – max, 3 – min
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src, dst TABLE(0, 0, 1, 0, 1, 1, 1, 0, 0, 1) 'save data into TABLE(0) ZV_IMGGENCONST(src,3,3,1,0,0) ZV_STATROW(src,dst,0) 'count row elements in matrix

	in sum method to dst.

5.6.3.4. ZV_STATCOL – Column Element Statistic

Type	Statistics
Description	It is used to count each column's elements, and calculate the counted value.
Grammar	ZV_STATCOL(src,dst,type) src: ZVOBJECT type, image or matrix of single-channel tabId: TABLE index, matrix, counted result, row 1, column N type: type of statistic, 0 – sum, 1- average, 2 – max, 3 – min, when type is 0 and 1, data type is 64F, when type is 2 and 3, dst type is image, so data type is same as src. If the type is matrix, the data type is 64F.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src， dst TABLE(0, 0, 1, 0, 1, 1, 1, 0, 0, 1) 'save data into TABLE(0) ZV_IMGGENCONST(src,3,3,1,0,0) ZV_STATCOL(src,dst,0) 'count column elements in matrix in sum method to dst.

5.6.3.5. ZV_MEAN – Average Value

Type	Statistics
Description	It is used to count the average value of each channel.
Grammar	ZV_MEAN(src,tabId) src: ZVOBJECT type, image or matrix tabId: TABEL index, average value of each channel
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img TABLE(0, 0, 1, 0, 1, 1, 1, 0, 0, 1) 'save data into TABLE(0)

	ZV_IMGGENCONST(img,3,3,1,0,0) ZV_MEAN(img, 0) 'calculate average of image src's each channel into TABLE(0)

5.6.3.6. ZV_MEANSDEV – Average Value and Standard Deviation

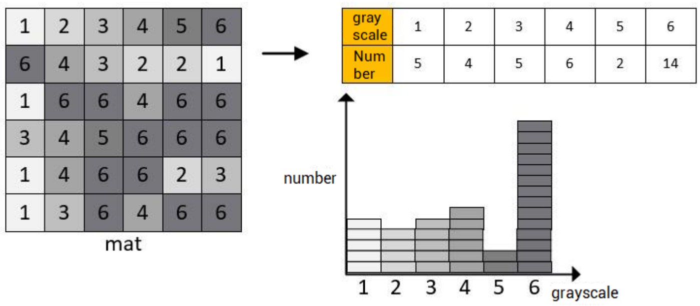
Type	Statistics
Description	It is used to count the average value and standard deviation of each channel.
Grammar	ZV_MEAN(src,tabId) src: ZVOBJECT type, image or matrix tabId: TABEL index, they are average value and standard deviation of each channel in order
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img TABLE(0, 0, 1, 0, 1, 1, 1, 0, 0, 1) 'save data into TABLE(0) ZV_IMGGENCONST(img,3,3,1,0,0) ZV_MEANSDEV(img, 0) 'calculate average value of standard deviation of each channel and save them into TABLE(0).

5.6.3.7. ZV_MINMAXLOC – Location of Minimal & Maximum

Type	Statistics
Description	It is used to the value and position of minimal and maximum, values and position are saved continuously, the minimal value is in the front.
Grammar	ZV_MINMAXLOC(src,tabId) src: ZVOBJECT type, single-channel image or matrix tabId: TABEL index, 6 output parameters, the positioning result is min, x,y coordinates of min, max, x,y coordinates of max.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT img TABLE(0, 0, 1, 0, 1, 1, 1, 0, 0, 1) 'save data into TABLE(0) ZV_IMGGENCONST(img,3,3,1,0,0) ZV_MEANSDEV(img, 0) 'image minimal value in TABLE (0), the coordinate is (TABLE(1), TABLE(2)), the maximum value is TABLE(3), the coordinate is (TABLE(4), TABLE(5))

5.6.3.8. ZV_HIST -- Histogram

Type	Statistics
Description	It is used to calculate the histogram of grayscale image. Histogram: the number of pixels of each kind of grayscale image, and it reflects the frequency of each kind of grayscale. 
Grammar	ZV_HIST(src,hist,size,lower,upper) src: ZVOBJECT type, single-channel image or matrix hist: ZVOBJECT type, calculated histogram result, matrix type size: the number of histogram data segments, the maximum value is 8192, the minimum value is 1 lower: the minimum value of src included in the histogram, pixels with a grayscale lower than this value will not be included in the statistics upper: the maximum value of src included in the histogram, pixels with a grayscale greater than this value will not be included in the statistics
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img

	TABLE(0, 0, 1, 0, 1, 1, 1, 0, 0, 1) 'save data into TABLE(0) ZV_IMGGENCONST(img,3,3,1,0,0) ZV_HIST(src,hist,256,0,255) 'calculate the histogram of the image hist

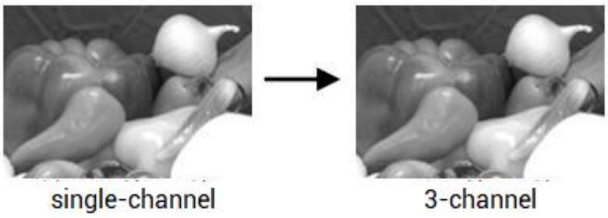
5.7.Preprocessing

5.7.1.Color

5.7.1.1. ZV_RGBTOGRAY – From RGB To Grayscale

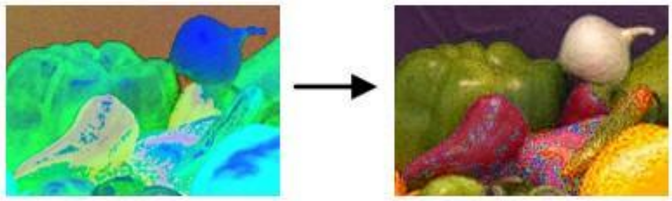
Type	Color
Description	It is used to convert RGB or RGBA form image to grayscale image, 8U or 16U type, gray = r*0.299 + g*0.587 + b*0.114.
Grammar	ZV_RGBTOGRAY(src, dst) src: ZVOBJECT type, images of 3-channel rgb or 4-channel rgba dst: ZVOBJECT type, single-channel gray image
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT rgb, gray ZV_READIMAGE(rgb,"test.jpg",0) 'read image in the original format ZV_RGBTOGRAY(rgb,gray) 'from rgb to grayscale
Related Instruction	ZV_GRAYTORGB

5.7.1.2. ZV_GRAYTORGB – From Grayscale To RGB

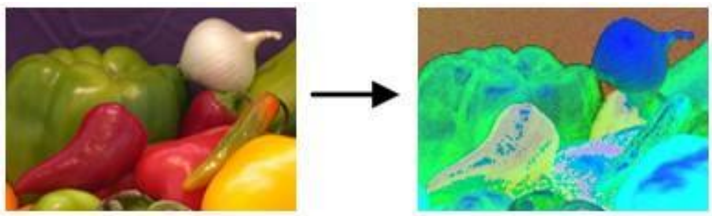
Type	Color
Description	It is used to convert grayscale image to RGB or RGBA, 8U or 16U type.
Grammar	ZV_GRAYTORGB(src, dst) src: ZVOBJECT type, grayscale image dst: ZVOBJECT type, 3-channel rgb image
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(rgb,"test.jpg",0) 'read image in the original format ZV_GRAYTORGB(src, dst) 'from grayscale to rgb
Related Instruction	ZV_RGBTOGRAY

5.7.1.3. ZV_COLORTORGB – From Other Colors To RGB

Type	Color
Description	It is used to convert other colors images to RGB, 8U type.
Grammar	ZV_COLORTORGB(src, dst, colorSpace) src: ZVOBJECT type, source image is the 3-channel image dst: ZVOBJECT type, image colorSpace: other colors space, refer to "from RGB to other colors".
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

<table>
<tr><td>Example</td><td>
ZVOBJECT src, dst
ZV_READIMAGE(rgb,"test.jpg",0) ‘read image in the original format
ZV_COLORTORGB(src, dst, 1) 'from HSV form to RGB</td></tr>
</table>

5.7.1.4. ZV_RGBTOCOLOR – From RGB To Other Colors

<table>
<tr><td>Type</td><td>Color</td></tr>
<tr><td>Description</td><td>It is used to convert RGB images to other colors, 8U type.</td></tr>
<tr><td>Grammar</td><td>ZV_RGBTOCOLOR(src, dst, colorSpace)
src: ZVOBJECT type, RGB source image
dst: ZVOBJECT type, image that needs to be converted
colorSpace: color space
<table>
<tr><th>colorSpace</th><th>Description</th></tr>
<tr><td>0</td><td>YUV</td></tr>
<tr><td>1</td><td>HSV</td></tr>
<tr><td>2</td><td>Lab</td></tr>
<tr><td>3</td><td>HLS</td></tr>
<tr><td>4</td><td>YCrCb</td></tr>
<tr><td>5</td><td>Luv</td></tr>
<tr><td>6</td><td>XYZ</td></tr>
<tr><td>7</td><td>RGBA</td></tr>
</table></td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><td>Example</td><td></td></tr>
</table>

	ZVOBJECT src, dst ZV_READIMAGE(rgb,"test.jpg",0) ‘read image in the original format ZV_RGBTOCOLOR(src, dst, 1) 'convert the RGBA color of src image to dst image

5.7.1.5. ZV_BAYERTORGB – From Bayer To RGB

<table>
<tr><td>Type</td><td>Color</td></tr>
<tr><td>Description</td><td>It is used to convert bayer images to rgb images, 8U or 16U type.</td></tr>
<tr><td>Grammar</td><td>ZV_BAYERTORGB(src, dst, bayerType)
src: ZVOBJECT type, single-channel bayer image
dst: ZVOBJECT type, 3-channel rgb image
bayerType: bayer type
<table><tr><td>bayerType</td><td>Description</td></tr><tr><td>0</td><td>BG</td></tr><tr><td>1</td><td>GB</td></tr><tr><td>2</td><td>RG</td></tr><tr><td>3</td><td>GR</td></tr></table></td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><td>Example</td><td>ZVOBJECT src, dst
ZV_READIMAGE(rgb,"test.jpg",0)
‘read image in the original format
ZV_BAYERTORGB(src, dst, 0) 'convert the image src with bayer type to image dst with RGB</td></tr>
</table>

5.7.2. Geometric Transformation

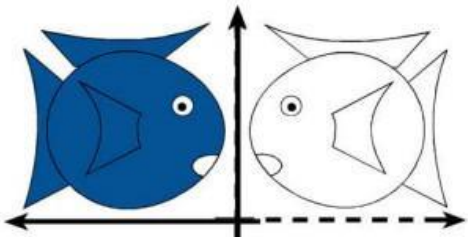
5.7.2.1. ZV_MIRROR – Mirror

Type	Geometric transformation

<table>
<tr>
<td>Description</td>
<td>
Calculate the mirror of the image, image homogeneous coordinate ring transformation formula:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & \mathrm{e} & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Below is one horizontal mirror, flip along the axis y, then:



Flip along the axis x:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Flip along the axis y:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

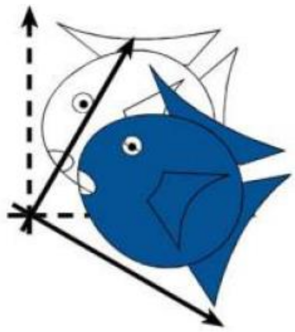
Mirror of origin point:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
</td>
</tr>
</table>

Grammar	ZV_MIRROR(src,dst,type) src: ZVOBJECT type, the source image is a single-channel or three-channel image dst: ZVOBJECT type, image after mirroring type: mirror type: 0- vertical mirroring, the flip axis is the horizontal axis, that is, flip up and down. 1- horizontal mirror, the flip axis is the vertical axis, that is, flip left and right. 2- diagonal mirror, the flip axis is the main diagonal, that is, both the horizontal and vertical axes are used as the flip axis, flipping is performed along both axes
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src, dst ZV_READIMAGE(rgb,"test.jpg",0) 'read image in the original format ZV_MIRROR(src, dst, 1) 'dst is the horizontally mirrored image of the generated source image src

5.7.2.2. ZV_ROTATE – Rotation

Type	Geometric transformation
Description	Rotate the image clockwise by angle around the center point, the unit is angle, and the image homogeneous coordinate ring transformation formula: $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ Below is one rotation example:

<table>
<tr><td></td><td>


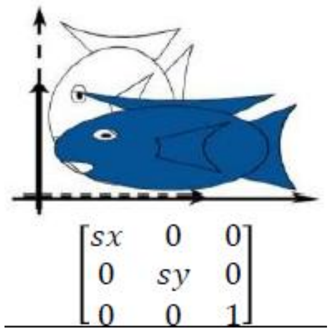
$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
</td></tr>
<tr><td>Grammar</td><td>
ZV_ROTATE(src,dst,angle,interp)
src: ZVOBJECT type, the source image is a single-channel or three-channel image
dst: ZVOBJECT type, the rotated image has the same size and type as the original image
angle: rotation angle, the direction is determined according to the image coordinate system, clockwise is positive
interp: interpolation algorithm
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>nearest neighbor interpolation</td></tr>
<tr><td>1</td><td>bilinear interpolation</td></tr>
<tr><td>2</td><td>bicubic interpolation</td></tr>
<tr><td>3</td><td>LANCZOS</td></tr>
</table>
</td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><td>Example</td><td>
ZVOBJECT src, dst
ZV_READIMAGE(rgb,"test.jpg",0)
‘read image in the original format
ZV_ROTATE(src, dst, 30, 0) 'assign the src image that is rotated 30 degrees to dst image
</td></tr>
</table>

5.7.2.3. ZV_ZOOM – Scale Factor Zooming

<table>
<tr><th>Type</th><td>Geometric transformation</td></tr>
<tr><th>Description</th><td>The image src is scaled according to the scaling factors sx and sy, and the image homogeneous coordinate ring transformation formula is:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Below shows one zooming example:


</td></tr>
<tr><th>Grammar</th><td>ZV_ZOOM(src,dst,sx,sy,interp)
src: ZVOBJECT type, the source image is a single-channel or three-channel image
dst: ZVOBJECT type, the zoomed image
sx: the zoom ratio in the width direction, > 0, the zoomed width is src.width * sx, and the zoomed width is rounded down
sy: the scaling ratio in the height direction, ≥ 0, if it is = 0, take sy = sx, the height after scaling is src.height * sy, and the height after scaling is rounded down
interp: interpolation algorithm
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>nearest neighbor interpolation</td></tr>
<tr><td>1</td><td>bilinear interpolation</td></tr>
<tr><td>2</td><td>bicubic interpolation</td></tr>
<tr><td>3</td><td>LANCZOS</td></tr>
</table>
</td></tr>
<tr><th>Controller</th><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
</table>

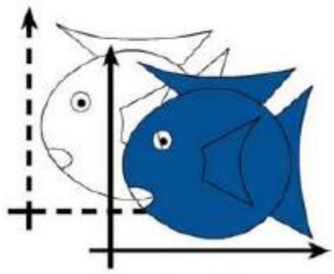
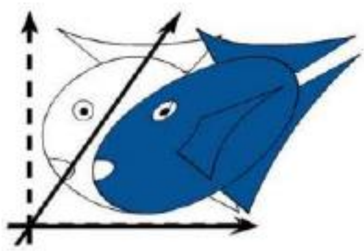
Example	ZVOBJECT src, dst ZV_READIMAGE(rgb,"test.jpg",0) 'read image in the original format ZV_ROTATE(src, dst, 0.5, 0.5, 1) 'both width and height are reduced to half

5.7.2.4. ZV_RESIZE – Target Size Zooming

Type	Geometric transformation
Description	According to the size of target image, zoom in and out the src.
Grammar	ZV_RESIZE(src,dst,dw,dh,interp) src: ZVOBJECT type, the source image is a single-channel or three-channel image dst: ZVOBJECT type, the zoomed image dw: image width after zooming, the range is [1, 32766] dh: image height after zooming, the range is [1, 32766] interp: interpolation algorithm Value / Description: 0 – nearest neighbor interpolation 1 – bilinear interpolation 2 – bicubic interpolation 3 – LANCZOS
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src, dst ZV_READIMAGE(rgb,"test.jpg",0) 'read image in the original format ZV_ROTATE(src, dst, 512, 512, 1) 'the image is scaled to 512x512 size, using bilinear interpolation, and the size of dst is 512x512

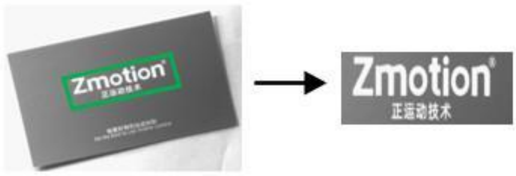
Value	Description
0	nearest neighbor interpolation
1	bilinear interpolation
2	bicubic interpolation
3	LANCZOS

5.7.2.5. ZV_AFFINE – Image Affine Transformation

Type	Geometric transformation
Description	Perform affine transformation on the image, and the image homogeneous coordinate ring transformation formula: $$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$ ● translation transformation: the figure below is a schematic diagram of translation.  $$\begin{bmatrix} 1 & 0 & X \\ 0 & 1 & Y \\ 0 & 0 & 1 \end{bmatrix}$$ ● shear transformation: below is a schematic diagram of shearing along the x-axis.  ➢ shear along axis x: $$\begin{bmatrix} 1 & tan\theta & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ ➢ shear along axis y: $$\begin{bmatrix} 1 & 0 & 0 \\ tan\theta & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

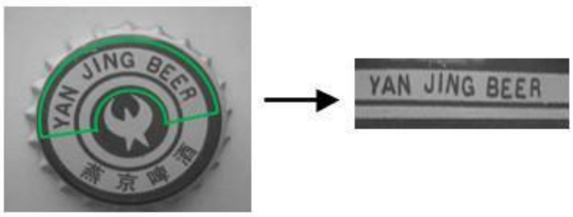
<table>
<tr>
<td>Grammar</td>
<td>ZV_AFFINE(src,mat,dst[,dw = 0,dh = 0,interp = 1,border = "0"])
src: ZVOBJECT type, the source image is a single-channel or three-channel image
mat: ZVOBJECT type, radial transformation matrix, 2 rows and 3 columns or 3 rows and 3 columns
dst: ZVOBJECT type, transformed image
dw: the width of the dst image, the default is 0, which is equal to src, and the range is [1,32766]
dh: the height of the dst image, the default value is 0, equal to src, range [1,32766]
interp: interpolation algorithm, default bilinear interpolation, refer to “ZV_ROTATE”.
border: string type, border processing method, default value "0", fill with 0 beyond the image part, commonly used values are as follows:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>“0”</td><td>Make up 0 to fill in `iiiiii|abcdefgh|iiiiiii`</td></tr>
<tr><td>“mirror1”</td><td>element symmetric `gfedcb|abcdefgh|gfedcba`</td></tr>
<tr><td>“mirror”</td><td>boundary symmetry `fedcba|abcdefgh|hgfedcb`</td></tr>
<tr><td>“continue”</td><td>repeat `aaaaaa|abcdefgh|hhhhhhh`</td></tr>
<tr><td>“wrap”</td><td>surround `cdefgh|abcdefgh|abcdefg`</td></tr>
</table>
</td>
</tr>
<tr>
<td>Controller</td>
<td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td>
</tr>
<tr>
<td>Example</td>
<td>ZVOBJECT mat, src, dst
ZV_READIMAGE(rgb,"test.jpg",0)
‘read image in the original format
ZV_IMGINFO((src,0)
ZV_GETSIMILARITYP(mat,TABLE(0)/2,TABLE(1)/2,30,0.8)
'the image mat is rotated 30° around the center and reduced by 0.8, the size of the image remains the same, and the surrounding padding is 0
ZV_AFFINE(src, mat,dst,0,0,1,”0”)</td>
</tr>
</table>

5.7.2.6. ZV_WRAPRECT2 – Capture Rotated Rectangle Image

Type	Geometric transformation
Description	Extract the area image specified by the rotation rectangle roi from the image, and the roi should not exceed the range of img
Grammar	ZV_WARPRECT2(img,subImg,cx,cy,w,h,angle,interp) img: ZVOBJECT type, the source image is a single-channel or three-channel image subImg: ZVOBJECT type, the captured image cx: the x coordinate of the center of the rotating rectangle cy: the x coordinate of the center of the rotating rectangle w: the width of the rotated rectangle, > 1, range [1,32766] h: height of the rotated rectangle, > 1, range [1,32766] angle: rotation rectangle angle, image coordinate system, clockwise is negative, the unit is degree interp: interpolation algorithm, if it is < 0, it defaults to bilinear interpolation, refer to "ZV_ROTATE".
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original image format ZV_WARPRECT2(src,dst,406,280,400,105,-13,0) 'capture image from src

5.7.2.7. ZV_WRAPRING – Capture Ring Image

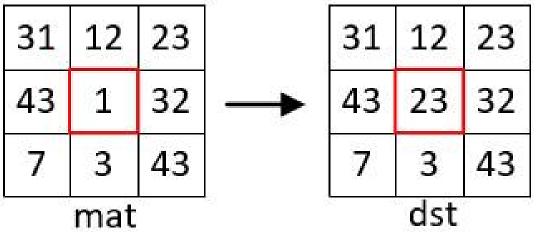
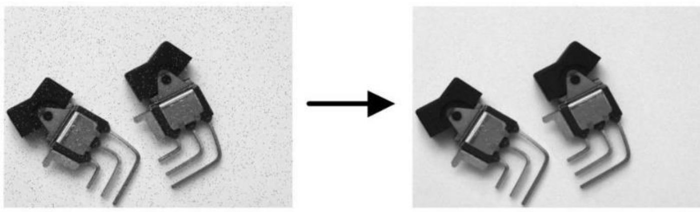
Type	Geometric transformation
Description	Extract the area image specified by the ring roi from the image, and the roi should not exceed the range of img

Grammar	ZV_WARPRING (src,subImg,cx,cy,radius,annR,startA,extentA,interp) img: ZVOBJECT type, the source image is a single-channel or three-channel image subImg: ZVOBJECT type, the captured image cx: the x coordinate of the center of the circle cy: the y coordinate of the center of the circle radius: the radius of the centerline of the ring, > 0 annR: ring width, (0,r) startA: the starting angle of the ring, the image coordinate system, clockwise is positive, the unit is degree extentA: angle range, the range is (0,360], if it is > 360, take 360, the unit is degree interp: interpolation algorithm, if it is < 0, it defaults to bilinear interpolation, refer to "ZV_ROTATE".
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original image format ZV_WARPRING(src, dst,320,240,60,20,0,270,1) 'capture image from src

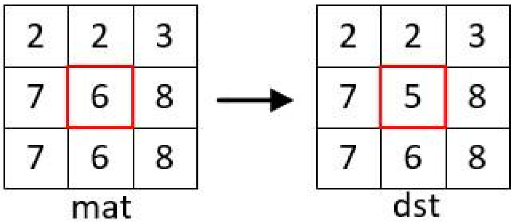
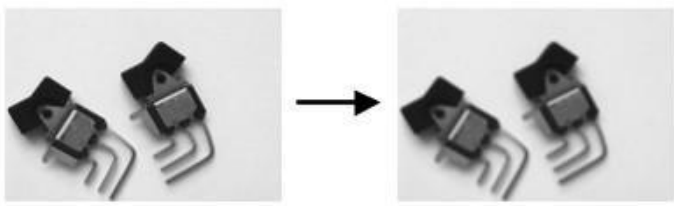
5.7.3. Filtering

Image filtering refers to suppressing the noise of the target image under the condition of preserving image details as much as possible, which is an indispensable operation in image preprocessing.

5.7.3.1. ZV_MEDIANBLUR – Media Filtering

Type	Filtering
Description	Median filtering is a nonlinear smoothing technique that can be used to remove isolated noise points. The median filter can protect the edge of the signal from being blurred while filtering out the noise. These characteristics are not available in the linear filtering method. Principle: it sets the gray value of each pixel as the median value of the gray values of all pixels in a certain neighborhood window of the point. Examples are as follows: 
Grammar	ZV_MEDIANBLUR(src,dst,size) src: ZVOBJECT type, the source image is a single-channel or three-channel image dst: ZVOBJECT type, filtered image size: filter size, range is [1,201], preferably an odd number, if an even number is input, the operator will automatically convert it to the nearest odd number.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original image format ZV_MEDIABLUR(src, dst,3) '3*3 median filter

5.7.3.2. ZV_MEANBLUR – Mean Filtering

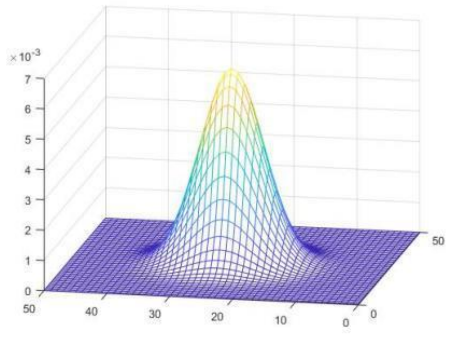
Type	Filtering
Description	Mean filtering is a typical linear filtering algorithm. Principle: use average value to replace each pixel value in the original image. Boundary handling is element-wise symmetric (see Custom Morphology). Calculation formula: $$g(x,y)=\frac{\Sigma f(x,y)}{m}$$ "m" means the total number of pixels in this template that includes current pixel. Example: 
Grammar	ZV_MEANBLUR(src,dst,size) src: ZVOBJECT type, the source image is a single-channel or three-channel image dst: ZVOBJECT type, filtered image size: filter size, range is [1,201].
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original image format ZV_MEANBLUR(src, dst,3) '3*3 mean filter

5.7.3.3. ZV_GAUSSBLUR – Gaussian Filtering

<table>
<tr><th>Type</th><td>Filtering</td></tr>
<tr><th>Description</th><td>Gaussian filtering is a linear smoothing filter, which is suitable for eliminating Gaussian noise and is widely used in the noise reduction process of image processing. Boundary handling is element-wise symmetric (see Custom Morphology).
Two-dimensional Gaussian function:

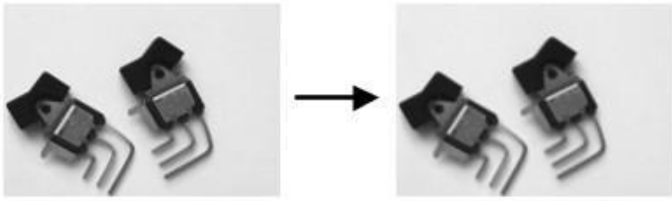
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Image:



Commonly used 3*3 and 5*5 gaussian template: (standard difference = 1.3)

$$\frac{1}{16}\times\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array} \qquad \frac{1}{273}\times\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 7 & 4 & 1 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 7 & 26 & 41 & 26 & 7 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 1 & 4 & 7 & 4 & 1 \\ \hline\end{array}$$
</td></tr>
<tr><th>Grammar</th><td>ZV_GAUSSBLUR (src,dst,size)
src: ZVOBJECT type, the source image is a single-channel or three-channel image
dst: ZVOBJECT type, filtered image
size: filter size, range is [1,201], preferably an odd number, if an even number is input, the operator will automatically convert it to the nearest odd number.</td></tr>
<tr><th>Controller</th><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
</table>

Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original image format ZV_GAUSSBLUR(src, dst,3) '3*3 gaussian filter

5.7.3.4. ZV_BILATERALFLR – Bilateral Filtering

Type	Filtering
Description	Image bilateral filter.
Grammar	ZV_BILATERALFLR (src,dst,sigmaSpace,sigmaRange) src: input image dst: output image sigmaSpace: space filtering parameters, range is [1, 50] sigmaRange: value filtering parameters, range is [1, 50]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original image format ZV_GAUSSBLUR(src, dst,3,3) 'image bilateral filtering

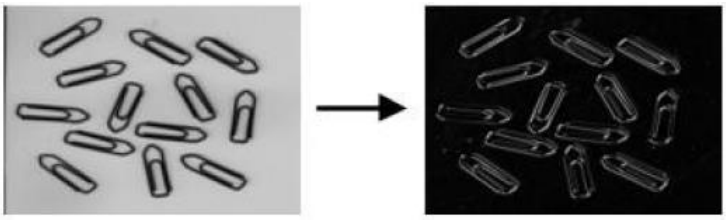
5.7.3.5. ZV_SCHARR – SCHARR Filtering

Type	Filtering
Description	SCHARR filtering.

Grammar	ZV_SCHARR (src,dst,dx,dy) src: input image dst: output image dx: derivative order in the x direction, can only be 0 or 1 dy: derivative order in the y direction, can only be 0 or 1 and dx+dy must be equal to 1
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original image format ZV_SCHARR(src, dst, 1, 0) 'SCHARR filtering

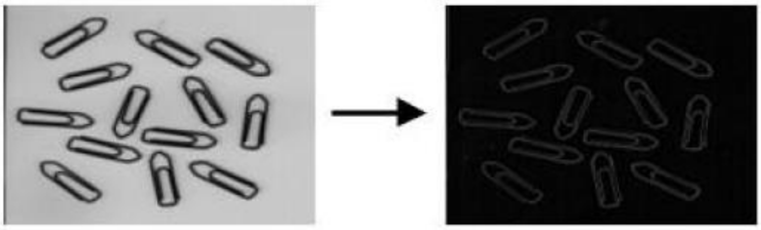
5.7.3.6. ZV_SOBEL – Sobel Edge Detection

Type	Filtering
Description	Sobel operator is a discrete differential operator mainly used for edge detection. It combines Gaussian smoothing and differential derivation to compute approximate gradients for grayscale images. Detect horizontal transformation, the 3*3 kernel is: $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$ Detect horizontal transformation, the 3*3 kernel is: $\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$
Grammar	ZV_SOBEL(src,dst,dx,dy,size) src: ZVOBJECT type, the source image is a single-channel or three-channel image dst: ZVOBJECT type, filtered image, data type 64F dx: derivative order in x direction, range [0, max(size,3)] dy: Derivative order in the y direction, range [0, max(size,3)],

	please note that dx and dy cannot be 0 at the same time. size: filter size, range [1,31], take an odd value, the common value is 3, if the even number is taken, it will be automatically converted to the nearest odd number.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original image format ZV_SOBEL(src, dst, 2, 2, 3) '3*3 filtering size, sobel edge detection

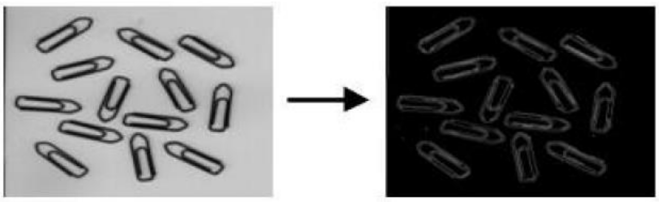
5.7.3.7. ZV_LAPLACE – Laplacian Edge Detection

Type	Filtering
Description	The Laplacian operator correctly locates the step edge points in the image, but is very sensitive to noise, and will lose part of the direction information of the edge, resulting in some discontinuous detection edges. The Laplacian operator is a second-order differential operator in n-dimensional Euclidean space. Suppose the picture is f , the definition of Laplacian operator: $$Laplacian(f) = \frac{\alpha^2 f}{\alpha x^2} + \frac{\alpha^2 f}{\alpha y^2}$$ Laplacian kernel of 3*3 is: $$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

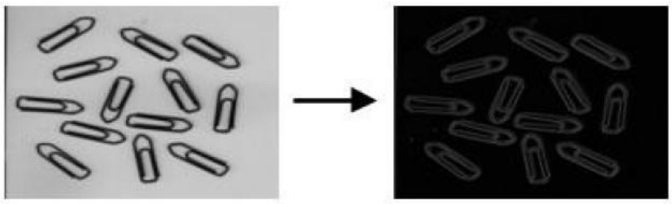
Grammar	ZV_LAPLACE(src,dst,size) src: ZVOBJECT type, the source image is a single-channel or three-channel image dst: ZVOBJECT type, the filtered image, the data type is the same as src size: filter size, range [1,31], odd
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original image format ZV_SOBEL(src, dst, 3) '3*3 Laplace edge detection.

5.7.3.8. ZV_CANNY – CANNY Edge Detection

Type	Filtering
Description	Canny edge detection, the steps are as follows: 1. use Gaussian filtering to eliminate noise, for example, a 3*3 Gaussian kernel: $\frac{1}{16}\times\begin{array}{\|c\|c\|c\|}\hline 1&2&1\\\hline 2&4&2\\\hline 1&2&1\\\hline\end{array}$ 2. calculate gradient magnitude and direction: (1) use sobel operator to obtain image gradient in the directions of x and y: $G_y=\begin{bmatrix}1&2&1\\0&0&0\\-1&-2&-1\end{bmatrix}\quad G_x=\begin{bmatrix}1&0&-1\\2&0&-2\\1&0&-1\end{bmatrix}$ (2) use below formula to calculate gradient magnitude and

	direction $$G = \sqrt{G_x^2 + G_y^2} \qquad \theta = arctan\left(\frac{G_y}{G_x}\right)$$ (The general values of the gradient direction are: 0°, 45°, 90°, 135°) 3. non-maximum suppression: this step excludes non-edge pixels and only retains some thin lines (candidate edges) 4. hysteresis threshold: Canny uses hysteresis thresholds (high and low thresholds): (1) If the magnitude of a certain pixel location exceeds the high threshold, the pixel is retained as an edge pixel. (2) If the magnitude of a certain pixel position is less than the low threshold, the pixel is excluded. (3) If the magnitude of a pixel location is between two thresholds, the pixel is only kept if it is connected to a pixel above the upper threshold.
Grammar	ZV_CANNY(src,dst,thresh1,thresh2,size) src: ZVOBJECT type, the source image is single-channel or three-channel image dst: ZVOBJECT type, edge image thresh1: low threshold thresh2: high threshold, > thresh1 size: filter size, range [3, 7], odd
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original image format ZV_SOBEL(src, dst, 10, 200, 3) '3*3 filtering, Canny edge detection

5.7.3.9. ZV_GRADIENT – Gradient Calculation

Type	Filtering
Description	Calculate the image gradient. The horizontal gradient and vertical gradient use the sobel operator to calculate the gradient in the x-axis and y-axis directions. All gradients are calculated by the following formula. Gradient magnitude: $$mag = \sqrt{g_x^2 + g_y^2}$$ Gradient direction: $$ang = arctan\frac{g_y}{g_x}$$
Grammar	ZV_GRADIENT(src,dst,type) src: ZVOBJECT type, the source image is a single-channel or three-channel image dst: ZVOBJECT type, the gradient image type: type of gradient, 0 – horizontal gradient, 1 – vertical gradient, 2 – all gradients.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original image format ZV_SOBEL(src, dst, 0) 'calculate image's horizontal gradient.

5.7.4.Frequency Domain Processing

5.7.4.1. ZV_DFT -- Fourier Transform

Type	Frequency domain
Description	Fourier transform, from the instant domain to the frequency domain, and the output spectrum size will be larger than the input image size because fast Fourier transform is used.
Grammar	ZV_DFT(src,dst) src: input single channel image dst: output spectrum, the spectrum is a double-channel single-precision floating-point image
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src, dst ZV_DFT(src,dst) 'convert image into spectrum

5.7.4.2. ZV_IDFT – Inverse Fourier Transform

Type	Frequency domain
Description	Fourier transform inversely, from the frequency domain to instant domain, the image obtained after inverse transformation will have black areas on the left and bottom, so a part of the image is intercepted by width and height, and the interception starts from the upper left corner.
Grammar	ZV_IDFT(src,dst,width,height) src: input spectrum, two-channel single-precision image dst: output single image width: the width of the output image, > 0, ≤ the width of src height: the height of the output image, > 0, ≤ the height of src
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT src, dst ZV_IDFT(src,dst,width,height) 'convert the spectrum into an image

5.7.4.3. ZV_MULSPECTRUM – Multiple Spectrum

Type	Filtering
Description	Multiple 2 spectrums.
Grammar	ZV_MULSPECTRUM(src1,src2,dst) src1: input spectrum, dual-channel single-precision floating-point image src2: input spectrum, the same size and type as src1 dst: output spectrum, the same size and type as src1
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src1, src2, dst ZV_MULSPECTRUM(src1,src2,dst) 'multiply two spectra

5.7.4.4. ZV_GENGAUSSFILTER – Gaussian Filter

Type	Filtering
Description	It is used to generate a frequency-domain Gaussian filter.
Grammar	ZV_GENGAUSSFILTER(filter,width,height,sigma1,sigma2) filter: output filter, dual-channel single-precision floating-point image width: filter width, positive integer height: filter height, positive integer sigma1: the filter corresponds to the standard deviation in the horizontal direction in the airspace, non-negative sigma2: the filter corresponds to the standard deviation in the vertical direction in the airspace, non-negative, sigma1 and sigma2 cannot be 0 at the same time
Controller	It is valid in controllers that support ZV function or they belong

	to 5XX series or above.
Example	ZVOBJECT filter ZV_GENGAUSSFILTER(filter,5,5,3,0) 'Generate frequency domain Gaussian filter

5.7.4.5. ZV_GENLPFILTER – Ideal Lowpass Filter

Type	Filtering
Description	It is used to generate a frequency-domain ideal lowpass filter.
Grammar	ZV_GENLPFILTER(filter,width,height,frequency) filter: output filter, dual-channel single-precision floating-point image width: filter width, positive integer height: filter height, positive integer frequency: cut-off frequency, which is a scaling factor of width and height, interval [0,1]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT filter ZV_GENLPFILTER(filter,5,5,0.3) 'Generate an ideal low-pass filter in the frequency domain

5.7.4.6. ZV_GENHPFILTER – Ideal High Pass Filter

Type	Filtering
Description	It is used to generate a frequency-domain ideal high-pass filter.
Grammar	ZV_GENHPFILTER(filter,width,height,frequency) filter: output filter, dual-channel single-precision floating-point image width: filter width, positive integer height: filter height, positive integer frequency: cut-off frequency, which is a scaling factor of width and height, interval [0,1]

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT filter ZV_GENHPFILTER(filter,5,5,0.3) 'Generate an ideal high-pass filter in the frequency domain

5.7.4.7. ZV_LPFILTER – Gaussian Lowpass Filter

Type	Filtering
Description	Frequency-domain Gaussian lowpass filter, is to blur the image and remove details.
Grammar	ZV_LPFILTER(src,dst,sizex,sizey) src: input image, single-channel dst: output image, single-channel sizex: the size in the x direction of the filter space, a positive integer, the larger the size, the blurrier the image sizey: the size of the y direction in the filter space, a positive integer, the larger the size, the blurrier the image
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src,dst ZV_READIMAGE(src,"test.jpg",0) 'read the picture in the original image format ZV_LPFILTER(src,dst,3,3) 'generate Gaussian low-pass filter in frequency domain

5.7.4.8. ZV_HPFILTER – Gaussian High-Pass Filter

Type	Filtering
Description	Frequency-domain Gaussian high-pass filter, is to get details.
Grammar	ZV_HPFILTER(src,dst,sizex,sizey) src: input image, single-channel dst: output image, single-channel

	sizex: the size in the x direction of the filter space, a positive integer, the larger the size, the blurrier the image sizey: the size of the y direction in the filter space, a positive integer, the larger the size, the blurrier the image
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src,dst ZV_READIMAGE(src,"test.jpg",0) 'read the picture in the original image format ZV_hPFILTER(src,dst,3,3) 'generate Gaussian high-pass filter in frequency domain

5.7.5. Morphology

5.7.5.1. ZV_ERODE – Erosion

Type	Morphology
Description	Erosion of kw*kh rectangular structure, for a binary image, assume that the current pixel is white, if one of its neighbors is a black pixel, then turn the current pixel into black. If it is a grayscale image, then take the minimum value of current pixel's neighbor. Boundary processing is element symmetry, which can refer to custom morphology. The corrosion diagram is as follows: 
Grammar	ZV_ERODE(src,dst,kw[,kh = 0]) src: ZVOBJECT type, the source image is a single-channel or three-channel image dst: ZVOBJECT type, the etched image kw: structural element width, range [1,511] kh; structure element height, range [1,511], if it is 0 then kh

	= kw
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the picture in the original image format ZV_ERODE(src,dst,3,3) 'matrix erosion, 3*3 structural elements

5.7.5.2. ZV_DILATE – Expansion

Type	Morphology
Description	Expansion of kw*kh rectangular structure, for a binary image, assume that the current pixel is black, if one of its neighbors is a white pixel, then turn the current pixel into white. If it is a grayscale image, then take the maximum value of current pixel's neighbor. Boundary processing is element symmetry, which can refer to custom morphology. The corrosion diagram is as follows: 
Grammar	ZV_ERODE(src,dst,kw[,kh = 0]) src: ZVOBJECT type, the source image is a single-channel or three-channel image dst: ZVOBJECT type, the expanded image

	kw: structural element width, range [1,511] kh; structure element height, range [1,511], if it is 0 then kh = kw
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the picture in the original image format ZV_DILATE(src,dst,3,3) 'matrix expansion, 3*3 structural elements

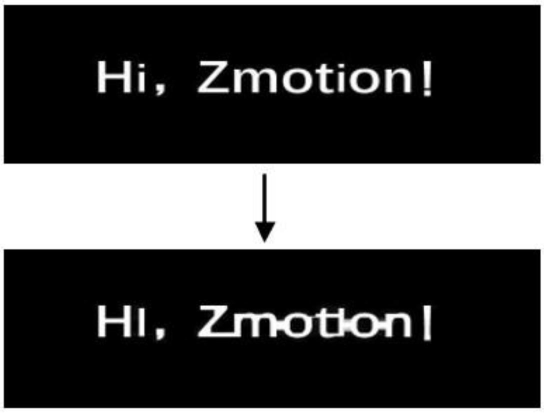
5.7.5.3. ZV_OPENING – Opening Operation

Type	Morphology
Description	The image opening operation of the rectangular structure is equivalent to first erosion and then expansion, which is used to remove isolated small pixels. Boundary processing is element symmetry, which can refer to custom morphology.
Grammar	ZV_OPENING(src,dst,kw[,kh = 0]) src: ZVOBJECT type, the source image is a single-channel or three-channel image dst: ZVOBJECT type, opening operated image kw: structural element width, range [1,511] kh; structure element height, range [1,511], if it is 0 then kh = kw

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the picture in the original image format ZV_OPENING (src,dst,5,5) 'opening operation, 5*5 structural elements

5.7.5.4. ZV_CLOSING – Closing Operation

Type	Morphology
Description	The image closing operation of the rectangular structure is equivalent to first expansion and then erosion, which is used to connect broken pixels together. Boundary processing is element symmetry, which can refer to custom morphology.
Grammar	ZV_OPENING(src,dst,kw[,kh = 0]) src: ZVOBJECT type, the source image is a single-channel or three-channel image dst: ZVOBJECT type, opening operated image kw: structural element width, range [1,511] kh; structure element height, range [1,511], if it is 0 then kh = kw
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the picture in the original image format ZV_CLOSING(src,dst,5,5) 'closing operation, 5*5 structural elements

5.7.5.5. ZV_MORPHSE – Custom Structural Element

Type	Morphology
Description	It is used to generate structuring elements for custom morphology.
Grammar	ZV_MORPHSE(kernel,shape,width,height,anchorX,anchorY) kernel: ZVOBJECT type, the generated structural element shape: the shape of the structural element, range [0,2], 0-rectangle, 1-cross, 2-ellipse, that is, an ellipse that fills the rectangle of the corresponding size width: width of the structural element, range [1,511] height: structural element height, range [1,511] anchorX: the x coordinate of the anchor point in the structural element coordinate system, the range is [0, width), -1 takes the center anchorY: the y coordinate of the anchor point in the structural element coordinate system, the range [0, height), -1 takes the center
Controller	It is valid in controllers that support ZV function or they belong

	to 5XX series or above.
Example	ZVOBJECT k ZV_MORPHSE(k,0,5,5,-1,-1) 'generate a 5x5 rectangular structuring element with an anchor in the center

5.7.5.6. ZV_MORPH – Custom Morphology

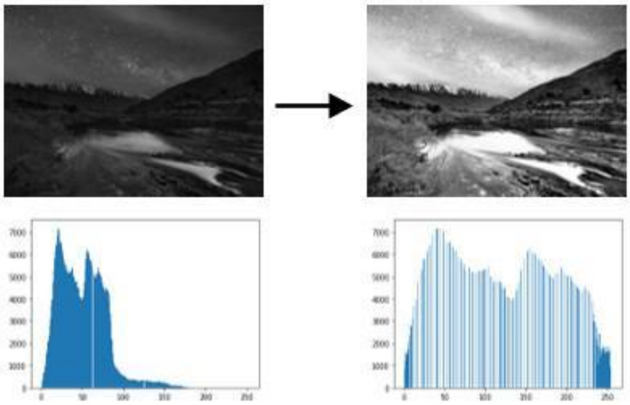
<table>
<tr><td>Type</td><td>Morphology</td></tr>
<tr><td>Description</td><td>Custom morphology operation</td></tr>
<tr><td>Grammar</td><td>ZV_MORPH(src, kernel, dst, op, anchorX, anchorY, iter[, border = "continu e"])
src: ZVOBJECT type, the source image is a single-channel or three-channel image
kernel: ZVOBJECT type, morphological structure element, generated by ZV_MORPHSE command
dst: ZVOBJECT type, image after morphological processing
op: morphological operation type: 0-corrosion, 1-expansion, 2-opening operation, 3-closing operation, 4-morphological gradient, 5-top hat, 6-bottom hat
anchorX: the x coordinate of the anchor point of the structural element, the range is [0, the width of the structural element), if it is -1, the center is taken
anchorY: the y coordinate of the anchor point of the structural element, the range is [0, the height of the structural element), if it is -1, the center is taken
iter: number of executions, range [1,20], common value 1
border: border processing, the values are as follows
<table>
<tr><td>Value</td><td>Constant</td></tr>
<tr><td>“mirror1”</td><td>Element symmetric `gfedcb|abcdefgh|gfedcba`</td></tr>
<tr><td>“mirror”</td><td>Boundary symmetry `fedcba|abcdefgh|hgfedcb`</td></tr>
<tr><td>“continue”</td><td>Repeat `aaaaaa|abcdefgh|hhhhhhh`</td></tr>
</table>
description: the vertical line on the right indicates the image boundary, and the letters indicate the pixel values at different</td></tr>
</table>

	distances from the boundary
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT k, src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original image format ZV_MORPHSE(k,0,5,5,-1,-1) 'generate a 5x5 rectangular structuring element with an anchor in the center ZV_MORPH(src,k,dst,2,-1,-1,2,"continue") 'perform two opening operations on src with a rectangular structure of size 5x5

5.7.6. Image Enhancement

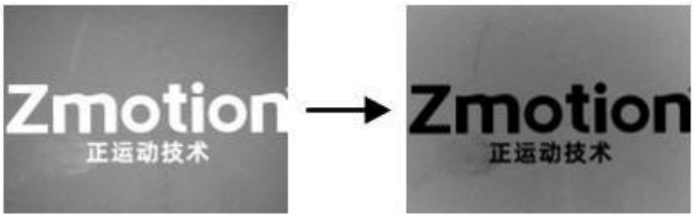
5.7.6.1. ZV_HISTEQ – Histogram Equalization

Type	Image enhancement
Description	Grayscale image histogram equalization is an important application of grayscale transformation. It is a method to enhance image contrast by stretching the pixel intensity distribution range. It is efficient and easy to implement, and is widely used in image enhancement processing. Histogram equalization steps: 1. Calculate image's histogram H 2. Perform histogram normalization 3. Calculate the histogram integral $$H'_{(i)} = \sum_{0 \le j \le i} H(j)$$ 4. Use H' as a lookup table for image transformation: $$dst\,(x, y) = H'(src(x, y))$$ The effect before and after histogram equalization and histogram comparison chart:

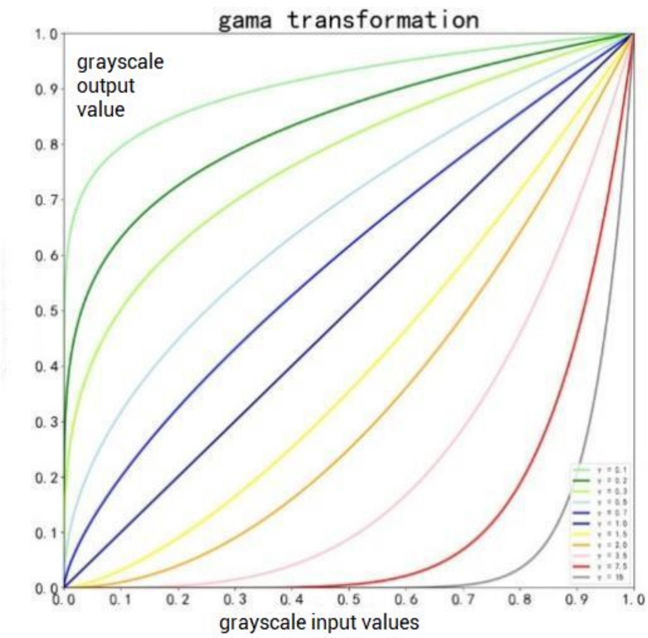
	
Grammar	ZV_HISTEQ(src,dst) src: ZVOBJECT type, source image, single channel 8U image dst: ZVOBJECT type, image after histogram equalization
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original format ZV_HISTEQ(src,dst) 'the source image src histogram equalized image is dst

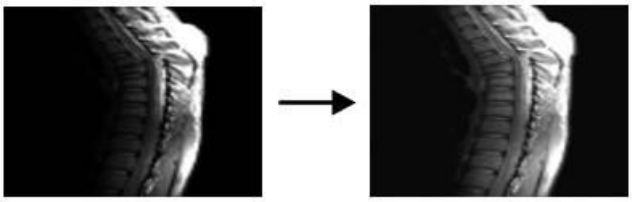
5.7.6.2. ZV_REVERSE – Image Inversion

Type	Image enhancement
Description	Image inversion, white pixels become black pixels, black pixels become white pixels
Grammar	ZV_REVERSE(src,dst) src: ZVOBJECT type, the source image is a single-channel or three-channel image dst: ZVOBJECT type, the output image
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the picture in the original format ZV_REVERSE(src,dst)　　'reverse the color of src

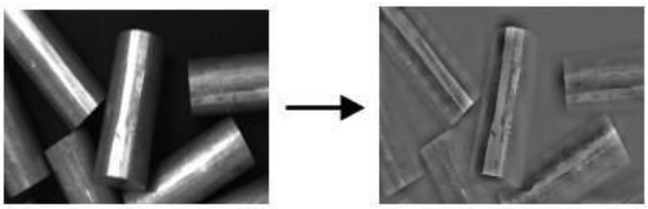
5.7.6.3. ZV_GAMMATRANS – Gamma Transformation

Type	Image enhancement
Description	Gamma transformation is performed on the image, and the gamma transformation is often used to adjust the contrast of the overexposed or underexposed (too dark) grayscale image. The calculation formula is as follows: $$I_{out} = cI_{in}^{\gamma}$$ Among them, c and y are normal numbers, c is the grayscale scaling factor, usually 1. y is the gamma factor size, which controls the scaling degree of the entire transformation. 

Grammar	ZV_GAMMATRANS(src,dst,gama) src: ZVOBJECT type, source image, single-channel image dst: ZVOBJECT type, etched image gama: gamma transformation value, a positive number. when it is < 1, dark pixels are stretched and bright pixels are compressed, and the smaller the gama value is, the more obvious the effect is. when it is = 1, no transformation is performed. when it is > 1, bright pixels are stretched and dark pixels are compressed, the value the bigger, the effect more obvious.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the picture in the original format ZV_GAMMATRANS(src,dst,0.6) 'perform gamma transformation on src

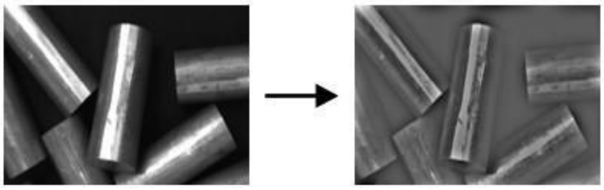
5.7.6.4. ZV_LIGHTCOMPENSATION – Light Compensation

Type	Image enhancement
Description	To perform illumination compensation on images with uneven illumination, the main ideas are as follows: 1. Find the average gray level of the original image I. 2. Divide the original image into N*M blocks, calculate the average value of each block, and obtain the brightness matrix D of the sub-block. 3. Subtract the average gray level of the source image from each element of the matrix D to obtain the brightness

	difference matrix E of the sub-block. 4. Change the matrix E difference into a brightness distribution matrix R with the same size as the source image. 5. Get the rectified image result = I - R.
Grammar	ZV_LIGHTCOMPENSATE(src,dst,blockSize) src: ZVOBJECT type, source image, single-channel image dst: ZVOBJECT type, image after lighting compensation blockSize: the size of the image block processing, a positive number, the smaller the size, the more obvious the light compensation is, but the image information that is lost is more obvious. It is recommended to use 32
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the picture in the original format ZV_LIGHTCOMPENSATION(src,dst,32) 'perform light compensation on image

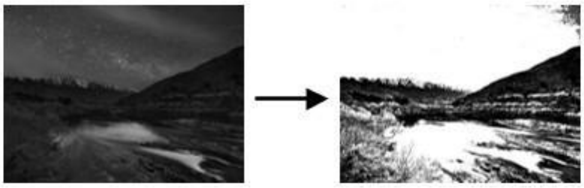
5.7.6.5. ZV_SHADECORRECT -- Shadow Correction

Type	Image enhancement
Description	The main ideas of shadow correction for unevenly illuminated images are as follows: 1. Reduce the original image I by a certain ratio to get a small image I_samll. 2. Filter the reduced image "I_small". 3. Enlarge the reduced image I_small to get a new image I_big. 4. Get the corrected image result = I – I_big.

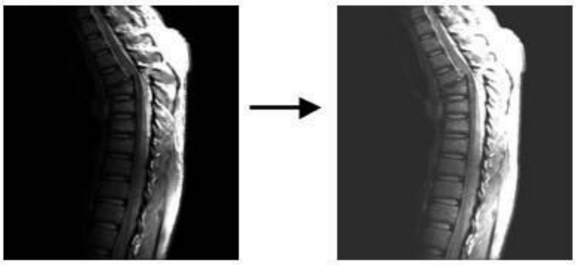
Grammar	ZV_SHADECORRECT(src,dst,filtersize) src: ZVOBJECT type, source image, single-channel image dst: ZVOBJECT type, image after shadow correction filterSize: the size of the image to be filtered, >0, the smaller the size, the more obvious the shadow correction, but the more serious the loss of image information
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the picture in the original format ZV_SHADECORRECT(src,dst,32) 'perform shadow correction on image

5.7.6.6. ZV_GRAYSTRETCH – Grayscale Stretch

Type	Image enhancement
Description	Perform gray scale stretching on the image, and manually stretch the pixels that are smaller than the low threshold to 0, and set the pixels that are larger than the high threshold to 255, that is, stretch the pixels within the high and low thresholds to 0-255
Grammar	ZV_GRAYSTRETCH(src,dst,minVal,maxVal,type) src: ZVOBJECT type, source image, single channel image dst: ZVOBJECT type, image stretched in gray scale minVal: low threshold, range [0,255] maxVal: high threshold, range [0,255], high threshold is > low threshold type: stretching type, 0-manual, 1-automatic, the high and low threshold parameters (minVal & maxVal) will not take effect

	in the automatic type
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the picture in the original format ZV_GRAYSTRETCH(src,dst,50,200,0) 'stretch pixels with gray values in the range [50,200]

5.7.6.7. ZV_NORMALIZE – Image Normalization

Type	Image enhancement
Description	Specify the mean and variance to normalize the image, that is, the normalized image mean and variance are the specified mean and variance.
Grammar	ZV_NORMALIZE(src,dst,mean,var) src: ZVOBJECT type, source image, single-channel image dst: ZVOBJECT type, normalized image mean: mean, range [0,255] var: variance, range [0,255]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0)

	'read the picture in the original format ZV_NORMALIZE(src,dst,120,50) 'specify the image whose mean is 120 and variance is 50 to normalize the image

5.7.6.8. ZV_EMPHASIZE – Emphasize Image

Type	Image enhancement
Description	Image enhancement is also called high-lift filtering, which enhances the edge details of the image. Its steps are mainly: 1. Smooth original image: f → s 2. Subtract the blurred image from the original image, and the resulting difference image is called the template: m = f – s. 3. Add the template to original image, g = f + k * m, and k > 1.
Grammar	ZV_EMPHASIZE(src,dst,kx,ky,factor) src: ZVOBJECT type, source image, single channel image dst: ZVOBJECT type, enhanced image, output kx: filter x direction size, range [1,201] ky: filter y-direction size, range [1,201] factor: enhancement factor, the enhancement ratio of high-frequency edge details, > 0, the larger the value, the stronger the enhancement of edge details
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src, dst ZV_READIMAGE(src,"test.jpg",0) 'read the picture in the original format ZV_EMPHASIZE(src,dst,3,3,2) 'enhance image edge details

5.7.6.9. ZV_DOTSIMAGE – Image Dot Enhanced

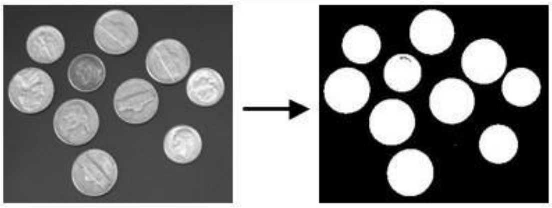
Type	Image enhancement
Description	According to dot's diameter in input image, enhance the

	corresponding dot in image. And this operator is especially suitable for segmentation of dot printing, for example, OCR applications.
Grammar	ZV_DOTSIMAGE(src,dst,diameter,type,shift) src: ZVOBJECT type, source image, single channel image dst: ZVOBJECT type, enhanced image, output diameter: the diameter of the point to be enhanced, optional values: odd numbers from 3 to 23 (including 3 and 23) type: enhance dark points, bright points, all points, corresponding values: -1, 1, 0 shift: transform the response of the filter, enhancing contrast (>0) or suppressing bright spots (-1). Valid values: -1, 0, 1, 2.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT src, dst ZV_DOTSIMAGE(src,dst,10,-1,1)

5.7.7.Binarization

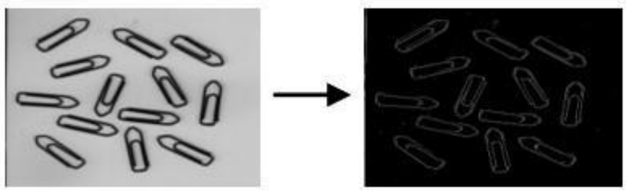
5.7.7.1. ZV_THRESH – Binarization

Type	Segmentation.
Description	Generate a binary image, the pixel whose value is ≥ thresh0 and ≤ thresh1 is 255, otherwise it is 0. There is no limit to the range of thresh0 and thresh1, and the judgment calculation is only performed according to the above range.
Grammar	ZV_THRESH(src,dst,thresh0,thresh1) src: ZVOBJECT type, image, unlimited channel numbers and type dst: ZVOBJECT type, binary image, single channel 8U type thresh0: low threshold thresh1: high threshold, thresh1 is ≥ to thresh0
Controller	It is valid in controllers that support ZV function or they belong

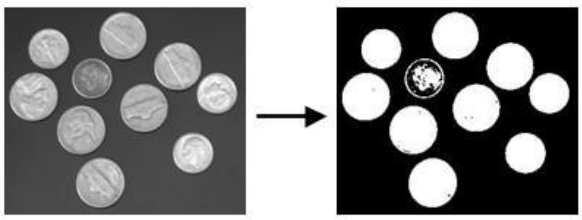
	to 5XX series or above.
Example	 ZVOBJECT src,dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original format ZV_THRESH(src,dst,95,255)'generate a binary image dst, the pixel whose value is between 20-60 is 255, otherwise it is 0

5.7.7.2. ZV_ADPTHRESH – Adaptive Binarization

Type	Segmentation.
Description	Perform adaptive thresholding on an input image, producing a binary image. The effect of adaptive thresholding is similar to high-pass filtering an image - extracting the contours of objects whose size depends on the size of the filter as well as the gradient magnitude of the object contours themselves. The larger the filter size is, the larger the target area can be found. According to experience, the filter size is usually twice of the extracted target contour. What's more, the offset range parameter offset is also very important. It is best not to set offset to 0, which will cause many small areas to be found (usually noise). Values such as 5-40 are more commonly used. The larger the offset, the smaller the extracted area

<table>
<tr>
<td>Grammar</td>
<td>
ZV_ADPTHRESH(src,dst,filterType,filterSize,offset,type)
src: ZVOBJECT type, source image, single channel image
dst: ZVOBJECT type, binary image
filterType: filtering algorithm used: 0-Gaussian filter, 1-mean filter
filterSize: filter size used, range [1,201]
offset: the allowable offset range of the result, range (-255, 255)
type: the result type, ranging from 0-3, the points in the image that meet the type selection type will be used as the target area for extraction
<table>
<tr><th>type</th><th>Description</th></tr>
<tr><td>0</td><td>source image - filtered image is between [-offset, offset], both bright and dark contours are extracted.</td></tr>
<tr><td>1</td><td>source image - filter image > offset or < -offset</td></tr>
<tr><td>2</td><td>source image - filtered image ≥ offset, bright contours are extracted</td></tr>
<tr><td>3</td><td>source image - filtered image ≤ -offset, dark contours are extracted</td></tr>
</table>
</td>
</tr>
<tr>
<td>Controller</td>
<td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td>
</tr>
<tr>
<td>Example</td>
<td>

ZVOBJECT src,dst
ZV_READIMAGE(src,"test.jpg",0)
'read the image in the original format
ZV_ADPTHRESH(src,dst,0,5,10,0)
'select the point where the difference between the image and its own 5x5 Gaussian filter image is between -10 and 10, that is, the smoother area
</td>
</tr>
</table>

5.7.7.3. ZV_AUTOTHRESH – Automatic Binarization

Type	Segmentation.
Description	Use the OTSU algorithm to calculate the optimal threshold and threshold the image, the OTSU algorithm regards the white pixels and black pixels after the threshold as two types, that is, the algorithm is to find the best threshold to maximize the inter-class variance of black and white pixels after thresholding.
Grammar	ZV_AUTOTHRESH(src,dst,tabId) src: ZVOBJECT type, source image, single-channel image dst: ZVOBJECT type, binary image tabId: TABLE index, output parameters, used segmentation threshold
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT src,dst ZV_READIMAGE(src,"test.jpg",0) 'read the image in the original format ZV_AUTOTHRESH(src,dst,0) 'threshold image using OTSU algorithm ? TABLE(0) 'print Threshold

Chapter VI Matching

6.1.Shape Matching

Note: the angle parameters of ZV_MCCREATESHAPE and ZV_MCCCREATESHAPESCALE commands to create templates are the starting angle and angle range, and ZV_SHAPECREATE and ZV_SHAPECREATERE are the starting angle and end angle.

For template matching based on shape contour, the matching process uses a pyramid, and the origin of the template is the center of the image.

6.1.1.ZV_MCCREATESHAPE – Create the Template

Type	Shape template creating
Description	Use the template image “img” and specify the effective area “re” of the template image to create a shape matching template, and this is mainly for when there is a lot of noise in the template image, using the re area to specify that some parts of the template image are valid to create a template instead of using the entire template image. The effective area in the template image is specified by the area re to create a template, and the part with more noise in the template image can be removed by performing some set operations or morphological operations on the re area, so as to obtain a more robust template feature. The created template reference position (template origin) is the center of the template image, that is, ((width-1)/2.0,(height-1)/2.0). According to the setting of the system parameter "ExtensionShape", the extension algorithm is supported, and the reference position is the center of gravity of the area re.
Grammar	ZV_MCCREATESHAPE (img, re, model, angleStart, angleExt, thresh [,ptRedu ce=0, minContLen=0, angleStep=0]) img: ZVOBJECT type, image for making templates, input parameter, 8U single-channel re: ZVOBJECT type, specify the effective area of the

	template image, and the part corresponding to re in the template image will be used to create the template. re is an area based on run-length encoding, and its set operation is more convenient to remove invalid parts and retain valid parts to create templates. Usually, for template images with more noise or inconspicuous contour features, re is used to remove weak features in the template image and retain strong features. If re is empty, the entire template is the valid area by default. model: ZVOBJECT type, created template, output parameter angleStart: starting angle, determined by the image coordinate system, range [-180,180) angleExt: angle range, range [0,360]. The end angle is the start angle plus the angle range. After the template is created, targets within the start and end angle ranges can be matched. thresh: contrast threshold for extracting edge contours--absolute threshold, range [0,255], when it is 0, an appropriate threshold will be selected internally, the greater the contrast, the stronger the strength of the extracted edge contour, this parameter can control the extraction of strong edges or weak edge, the smaller the threshold, the more weak-edges are extracted, and it may bring some noise at the same time. ptReduce: optimize to reduce the number of template points. If you set the greedy degree when searching for templates, it needs to set it lower, 0-no reduction, 1-slight reduction, 2-moderate reduction, 3-large reduction minContLen: minimum contour length, contours smaller than this length will not be extracted, this parameter can control the deletion of some short contours. When it is 0, the appropriate contour is automatically calculated internally. angleStep: angle step size, range [0,12]. The smaller the step size, the better the accuracy but the more time-consuming the matching. The larger the step size, the worse the accuracy but the less time-consuming the matching. It is unreasonable to set the step size too small or as when 0, an appropriate step size will be automatically selected internally, and 0 is recommended.

	Please attention since the angle needs to exceed 0, the angle step obtained by using the command ZV_SHAPEPARAM will be slightly different from angleStep. Notes: When creating a template, a target with a clear outline and unique feature is usually selected as a template, and the features should not be symmetric. The amount of template feature data is usually related to the template size and template parameters, and its data amount is proportional to the size of the template, the complexity of the template outline, the range of rotation angle, and the zoom range. The larger the template, the smaller the angle step used, and the larger the amount of feature data in the same angle range, so the matching time is more time-consuming. Creating a template and matching have a timeout mechanism, the default are 5000ms, when creating a template timeout, it can reduce the amount of template feature data by adjusting the template parameters appropriately (such as reducing the zoom range or using the outline point reduction parameter ptReduce, the default value of this parameter If it is 0, the contour points will not be simplified, and too serious reduction may affect the matching accuracy), or manually set the timeout period, such as ZV_SETSYSDBL ("ShapeCreateTimeout", 5000), ZV_SETSYSDBL ("ShapeFindTimeout", 5000). Creating a template also has a memory protection mechanism. When using an overly large template image to create a template with scaling, the memory occupied reaches the protection threshold. At this time, a memory error is reported and the template creation fails. It can reduce the amount of data by adjusting the template parameters, such as using the ptReduce parameter streamlines some points, or the threshold thresh is set to a larger point to only extract some obvious contour features.

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img 'template image ZVOBJECT model 'template ZVOBJECT re 'specify the effective area of the template image, and re needs to be generated, that is, the part of the template image corresponding to re is used to create the template ZV_READIMAGE(img, "test.png", 0) 'read the image in the original format ZV_REGENRECT(re,0,0,w,h) 'specify an area on the template image for creating a template, w and h are the width and height of the template image respectively ZV_MCCCREATESHAPE(img,re,model,-180,360,0,0,0,0) 'create template
Related Instruction	ZV_MCFINDSHAPE, ZV_MCFINDSHAPESTATE

6.1.2.ZV_MCCREATESHAPESCALE – Create Scaling Template

Type	Shape template creating
Description	Use the template image "img" and specify the effective area "re" of the template image to create a shape matching template, and this is mainly for when there is a lot of noise in the template image, using the re area to specify that some parts of the template image are valid to create a template instead of using the entire template image. The effective area in the template image is specified by the area re to create a template, and the part with more noise in the template image can be removed by performing some set operations or morphological operations on the re area, so as to obtain a more robust template feature. The created template reference position (template origin) is the

	center of the template image, that is, ((width-1)/2.0,(height-1)/2.0). According to the setting of the system parameter "ExtensionShape", the extension algorithm is supported, and the reference position is the center of gravity of the area re
Grammar	ZV_MCCREATESHAPESCALE (img, re, model, angleStart, angleExt, ScaleMin, ScaleMax, thresh [,ptRedu ce=0, minContLen=0, angleStep=0], scaleStep=0], levelNum=0) img: ZVOBJECT type, image for making templates, input parameter, 8U single-channel re: ZVOBJECT type, specify the effective area of the template image, and the part corresponding to re in the template image will be used to create the template. re is an area based on run-length encoding, and its set operation is more convenient to remove invalid parts and retain valid parts to create templates. Usually, for template images with more noise or inconspicuous contour features, re is used to remove weak features in the template image and retain strong features. If re is empty, the entire template is the valid area by default. model: ZVOBJECT type, created template, output parameter angleStart: starting angle, determined by the image coordinate system, range [-180,180) angleExt: angle range, range [0,360]. The end angle is the start angle plus the angle range. After the template is created, targets within the start and end angle ranges can be matched. scaleMin: minimum ratio of matching zoom, range [0.5, 2.0] scaleMax: maximum ratio of matching zoom, range [0.5, 2.0], ≥ scale_min, after creating the template, targets within the minimum and maximum scaling ranges can be matched. thresh: contrast threshold for extracting edge contours--absolute threshold, range [0,255], when it is 0, an appropriate threshold will be selected internally, the greater the contrast, the stronger the strength of the extracted edge contour, this parameter can control the extraction of strong edges or weak edge, the smaller the threshold, the more weak-edges are

	extracted, and it may bring some noise at the same time. ptReduce: optimize to reduce the number of template points. If you set the greedy degree when searching for templates, it needs to set it lower, 0-no reduction, 1-slight reduction, 2-moderate reduction, 3-large reduction minContLen: minimum contour length, contours smaller than this length will not be extracted, this parameter can control the deletion of some short contours. When it is 0, the appropriate contour is automatically calculated internally. angleStep: angle step size, range [0,12]. The smaller the step size, the better the accuracy but the more time-consuming the matching. The larger the step size, the worse the accuracy but the less time-consuming the matching. It is unreasonable to set the step size too small or as when 0, an appropriate step size will be automatically selected internally, and 0 is recommended. Please attention since the angle needs to exceed 0, the angle step obtained by using the command ZV_SHAPEPARAM will be slightly different from angleStep. scaleStep: scaling step size, [0, scaleMax - scaleMin], the smaller the step size, the better the accuracy but the more time-consuming the matching, the larger the step size the worse the accuracy but the less time-consuming the matching, the step size is too small or when it is 0, an appropriate step size will be automatically selected internally, and it is recommended to be 0. Please attention since the zoom needs to exceed 1, the zoom step obtained by using the command ZV_SHAPEDEFPARAM will be slightly different from the scaleStep. levelNum: the number of pyramid layers, range [0, infinity), the smaller the number of layers, the more time-consuming the matching, if it is 0 or the number of layers is too large, it will automatically select the appropriate number of layers, it is recommended to be 0 Notes: When creating a template, a target with a clear outline and

	unique feature is usually selected as a template, and the features should not be symmetric. The amount of template feature data is usually related to the template size and template parameters, and its data amount is proportional to the size of the template, the complexity of the template outline, the range of rotation angle, and the zoom range. The larger the template, the smaller the angle step used, and the larger the amount of feature data in the same angle range, so the matching time is more time-consuming. Creating a template and matching have a timeout mechanism, the default are 5000ms, when creating a template timeout, it can reduce the amount of template feature data by adjusting the template parameters appropriately (such as reducing the zoom range or using the outline point reduction parameter ptReduce, the default value of this parameter If it is 0, the contour points will not be simplified, and too serious reduction may affect the matching accuracy), or manually set the timeout period, such as ZV_SETSYSDBL ("ShapeCreateTimeout", 5000), ZV_SETSYSDBL ("ShapeFindTimeout", 5000). Creating a template also has a memory protection mechanism. When using an overly large template image to create a template with scaling, the memory occupied reaches the protection threshold. At this time, a memory error is reported and the template creation fails. It can reduce the amount of data by adjusting the template parameters, such as using the ptReduce parameter streamlines some points, or the threshold thresh is set to a larger point to only extract some obvious contour features.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img 'template image ZVOBJECT model 'template ZVOBJECT re 'specify the effective area of the template image, and re needs to be generated, that is,

	the part of the template image corresponding to re is used to create the template ZV_READIMAGE(img, "test.png", 0) 'read the image in the original format ZV_REGENRECT(re,0,0,w,h) 'specify an area on the template image for creating a template, w and h are the width and height of the template image respectively ZV_MCCCREATESHAPESCALE(img, re, model, -180, 360, 1, 1, 0, 0, 0, 0, 0, 0) 'create template
Related Instruction	ZV_MCFINDSHAPE, ZV_MCFINDSHAPESTATE

6.1.3.ZV_MCFINDSHAPE – Matching

Type	Shape template creating
Description	Use single-shape template to find and match in image “img”. According to template type and expansion algorithm parameters, it can support expansion algorithm.

Grammar	ZV_MCFINDSHAPE (model, img, matchs, minScore [,nums=0, maxOverlap=0.5, minThresh=-1,accuracy=1,speed=9,polar=0, deform=0, boundary=4]) model: ZVOBJECT type, shape template img: ZVOBJECT type, the search image to be matched, it cannot be 1:1 proportional to the template image, 8U single channel matchs: ZVOBJECT type, matching result, matrix type, n rows and 5 columns, each row has a matching target, and the columns are the matching score “score”, x coordinate, y coordinate, rotation angle “angle”, scaling “scale” minScore: the minimum matching score, (0,100], the higher the score, the more accurate the matching target nums: the maximum number of matches, [0, infinity), when num is > the real target, output all targets with target scores from high to low, when num is < the real target, output num targets with target scores from high to low, when num is 0, output all targets with target scores from high to low. maxOverlap: the maximum overlap rate. When the overlapping part of the matching results exceeds, only the best result will be kept. It is mainly used to remove overlapping target objects. When there is only one target in the search map, the parameter can also be set smaller, which will speed up the matching speed but not obvious. minThresh: the lowest edge threshold of the target contour, when minThresh is < zero, the threshold when creating the template will be used accuracy: matching accuracy, 0-pixel accuracy, 1-interpolation accuracy, 2-least squares fitting accuracy, 3-multiple iterations least squares fitting accuracy. Accuracy 1 can meet most applications, 2 or 3 are used in occasions with higher accuracy requirements, but it will also be more time-consuming. speed: matching speed 0-10, the bigger the speed, the faster, but may lose the target, when it is > 10, take 10

<table>
<tr>
<td></td>
<td>
polar: matching polarity
<table>
<tr><th>polar</th><th>Polarity</th><th>Description</th></tr>
<tr><td>0</td><td>+</td><td>All contour points' light and dark changes of the matching target and the template are consistent.</td></tr>
<tr><td>1</td><td>±</td><td>Both + and - are OK, and all contour points' light and dark changes of the matching target and the template are consistent or inverse.</td></tr>
<tr><td>2</td><td>Any</td><td>All contour points' light and dark changes of the matching target and the template are consistent or inverse.</td></tr>
</table>
deform: deformation size, relative to the template, it allows a slight deformation of the target contour, 0 - does not support deformation. 1 - support slight deformation, but it is more time-consuming, and can be combined with the matching threshold to remove some noise interference to increase the speed.
boundary: boundary mode, the extent to which the target contour exceeds the image boundary, 0-not exceeded, 1-a small amount exceeded, 2-moderate exceeded, 3-a large amount exceeded, 4-completely exceeded, and the matching time increases in sequence. When using the border, it needs to be used with the system parameter "ShapeOnBorder". The border mode will only work when this parameter is set to 1. Please set this value according to the actual situation.
</td>
</tr>
<tr>
<td>Controller</td>
<td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td>
</tr>
<tr>
<td>Example</td>
<td>

template

image to be matched
ZVOBJECT mod,img,re,matchImg,clrImg,rlts,
ZVOBJECT matRigid,modContList,dstContList
</td>
</tr>
</table>

	ZV_READIMAGE(img, "model.jpg", 0) 'read the image in the original format ZV_READIMAGE(matchImg, "1.png", 0) 'read the image in the original format ZV_MCCREATESHAPE(img,mod,re,-180,360,0,0,0,0) 'create template ZV_MCSHAPECONTLIST(mod, modContList) 'get template outline ZV_MCSHAPEFIND(mod,matchImg,rlts,90,1,0.6,-1,3,9,0) 'template matching ZV_MATGETROW(rlts,0,5,0) 'obtain the first row of the matching result matrix, which are: matching score, x coordinate, y coordinate, rotation angle "angle", and scaling "scale" ZV_GETRIGIDVECTOR(matRigid,0,0,0,TABLE(1),TABLE(2),TABLE(3)) 'calculate rigid transformation matrix ZV_CONTAFFINE(modContListt,matRigid,dstContList) 'contour affine transformation ZV_GRAYTORGB(matchImg,clrImg) 'convert grayscale image to RGB image ZV_CONTLIST(clrImg,dstContList,ZV_COLOR(0,255,0),0) 'draw the contour
Related Instruction	ZV_MCCREATESHAPE, ZV_MCCREATESHAPESTATE

6.1.4.ZV_MCFINDSHAPESTATE – Match & Output Contour State

Type	Shape template creating
Description	Use single-shape template to find and match in image "img", also output the matching state of template contour point.

Grammar	ZV_MCFINDSHAPESTATE (model, img, matchs, stats, minScore [,nums=0, maxOverlap=0.5, minThresh=-1, accuracy=1, speed=9, polar=0, deform=0, boundary=4]) model: ZVOBJECT type, shape template img: ZVOBJECT type, the search image to be matched, it cannot be 1:1 proportional to the template image, 8U single channel matchs: ZVOBJECT type, matching result, matrix type, n rows and 5 columns, each row has a matching target, and the columns are the matching score "score", x coordinate, y coordinate, rotation angle "angle", scaling "scale". stats: ZVOBJECT type, the matching status of each point of the template contour point, m x n image type, one template contour per row, and the matching status of each contour point is stored sequentially on the row, that is, for a certain contour point, the matching score is ≥ the set score, it is 1 (matching is successful), when it is < the matching score, it is 0 (matching fails, if it is empty, the contour point matching status will not be output, if it is not empty, the contour point matching status will be output, this output parameter is combined with the drawing template command ZV_DRASHAPEMATCH, then matching success points and failure points can be drawn in different colors. minScore: the minimum matching score, (0,100], the higher the score, the more accurate the matching target. nums: the maximum number of matches, [0, infinity), when num is > the real target, output all targets with target scores from high to low, when num is < the real target, output num targets with target scores from high to low, when num is 0, output all targets with target scores from high to low. maxOverlap: the maximum overlap rate. When the overlapping part of the matching results exceeds, only the best result will be kept. It is mainly used to remove overlapping target objects. When there is only one target in the search map, the parameter can also be set smaller, which will speed up the

matching speed but not obvious.

minThresh: the lowest edge threshold of the target contour, when minThresh is < 0, the threshold when creating the template will be used

accuracy: matching accuracy, 0-pixel accuracy, 1-interpolation accuracy, 2-least squares fitting accuracy, 3-multiple iterations least squares fitting accuracy. Accuracy 1 can meet most applications, 2 or 3 are used in occasions with higher accuracy requirements, but it will also be more time-consuming.

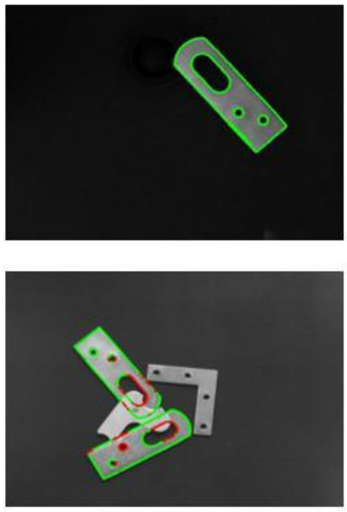
speed: matching speed 0-10, the bigger the speed, the faster, but may lose the target, when it is > 10, take 10

polar: matching polarity

polar	Polarity	Description
0	+	All contour points' light and dark changes of the matching target and the template are consistent.
1	±	Both + and - are OK, and all contour points' light and dark changes of the matching target and the template are consistent or inverse.
2	Any	All contour points' light and dark changes of the matching target and the template are consistent or inverse.

deform: deformation size, relative to the template, it allows a slight deformation of the target contour, 0 - does not support deformation. 1 - support slight deformation, but it is more time-consuming, and can be combined with the matching threshold to remove some noise interference to increase the speed.

boundary: boundary mode, the extent to which the target contour exceeds the image boundary, 0-not exceeded, 1-a small amount exceeded, 2-moderate exceeded, 3-a large amount exceeded, 4-completely exceeded, and the matching time increases in sequence. When using the border, it needs to be used with the system parameter "ShapeOnBorder". The border

	mode will only work when this parameter is set to 1. Please set this value according to the actual situation.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 template  image to be matched ZVOBJECT mod,img,re,matchImg,clrImg,rlts,stats ZV_READIMAGE(img,"model.jpg",0) 'read the image in the original image format ZV_READIMAGE(matchImg,"1.png",0) 'read the image in the original image format ZV_MCCCREATESHAPE(img,re,model,-180,360,0,0,0,0) 'create template ZV_MCFINDSHAPESTATE(model,matchImg,rlts,stats,90,1,0.6,-1,3,9,0) 'template matching ZV_GRAYTORGB(matchImg,clrImg) 'convert grayscale image to RGB image ZV_DRASHAPEMATCH(clrImg,model,rlts,stats,ZV_COLOR(0,255,0),ZV_CO LOR(255,0,0)) 'draw the template on the color image, and the contour points that match successfully are drawn in green, and the contour points that fail to match are drawn red
Related Instruction	ZV_MCCREATESHAPE, ZV_MCCREATESHAPESTATE, ZV_DRASHAPEMATCH

6.1.5.ZV_MCFINDSHAPERE – Match Supported Area

Type	Shape template matching
Description	Use single-shape template to find and match in image "img". According to template type and expansion algorithm parameters, it can support expansion algorithm.
Grammar	ZV_MCFINDSHAPERE (model, img, re, matchs, minScore [,nums=0, maxOverlap=0.5, minThresh=-1, accuracy=1, speed=9, polar=0, deform=0, boundary=4]) model: ZVOBJECT type, shape template img: ZVOBJECT type, the search image to be matched, 8U single channel, size must be bigger than template image re: ZVOBJECT type, valid region of specified matching image matchs: ZVOBJECT type, matching result, matrix type, n rows and 5 columns, each row has a matching target, and the columns are the matching score "score", x coordinate, y coordinate, rotation angle "angle", scaling "scale". minScore: the minimum matching score, (0,100], the higher the score, the more accurate the matching target. nums: the maximum number of matches, [0, infinity), when num is > the real target, output all targets with target scores from high to low, when num is < the real target, output num targets with target scores from high to low, when num is 0, output all targets with target scores from high to low. maxOverlap: the maximum overlap rate. Normal range is [0, 1], 0 – not overlap, that is, objects that are not overlapped are only found, 1 – overlap, that is, objects that are close are found. The overlap ratio indicates the allowed overlap ratio of the target. The overlap rate is calculated by dividing the overlap area of the minimum external moment of the target contour feature by the area of the minimum external moment. When the overlapping part of the matching results exceeds, only the best result will be kept. It is mainly used to remove overlapping target objects. When there is only one target in the search map, the

	parameter can also be set smaller, which will speed up the matching speed but not obvious. minThresh: the minimum edge threshold of the target contour. When minThresh is set to -1, the minimum threshold estimated from the template image will be used. If the difference between the template image and the matching image is large, it may cause the matching to fail. In this case, you should set the threshold yourself, such as setting it to 0. accuracy: matching accuracy, 0-pixel accuracy, 1-interpolation accuracy, 2-least squares fitting accuracy, 3-multiple iterations least squares fitting accuracy. Accuracy 1 can meet most applications, 2 or 3 are used in occasions with higher accuracy requirements, but it will also be more time-consuming. speed: matching speed 0-10, the bigger the speed, the faster, but may lose the target, when it is > 10, take 10 polar: matching polarity

polar	Polarity	Description
0	+	All contour points' light and dark changes of the matching target and the template are consistent.
1	±	Both + and - are OK, and all contour points' light and dark changes of the matching target and the template are consistent or inverse.
2	Any	All contour points' light and dark changes of the matching target and the template are consistent or inverse.

deform: deformation size, relative to the template, it allows a slight deformation of the target contour, 0 - does not support deformation. 1 - support slight deformation, but it is more time-consuming, and can be combined with the matching threshold to remove some noise interference to increase the speed.

boundary: boundary mode, the extent to which the target contour exceeds the image boundary, 0-not exceeded, 1-a small

	amount exceeded, 2-moderate exceeded, 3-a large amount exceeded, 4-completely exceeded, and the matching time increases in sequence. When using the border, it needs to be used with the system parameter "ShapeOnBorder". The border mode will only work when this parameter is set to 1. Please set this value according to the actual situation.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 template  image to be matched ZVOBJECT mod,img,re,matchImg,rlts ZV_READIMAGE(img,"model.jpg",0) 'read the image in the original image format ZV_READIMAGE(matchImg,"1.png",0) 'read the image in the original image format ZV_MCCCREATESHAPE(img,model,re,-180,360,0,0,0,0) 'create template ZV_REGENRECT(re, 200, 0, 400, 400) ZV_MCSHAPEFINDRE(mod, matchImg, re, rlts, 90, 1, 0.6, -1, 3, 9, 0) 'template matching
Related Instruction	ZV_MCCREATESHAPE,

6.1.6.ZV_MCFINDSHAPERESTATE – Match Supported Region & Output Contour State

Type	Shape template matching
Description	Use single-shape template to find and match in image "img", and output matching state of template contour point.

Grammar	ZV_MCFINDSHAPERESTATE (model, img, re, matchs, stat, minScore [,nums=0, maxOverlap=0.5, minThresh=-1, accuracy=1, speed=9, polar=0, deform=0, boundary=4]) model: ZVOBJECT type, shape template img: ZVOBJECT type, the search image to be matched, 8U single channel, size must be bigger than template image re: ZVOBJECT type, valid region of specified matching image matchs: ZVOBJECT type, matching result, matrix type, n rows and 5 columns, each row has a matching target, and the columns are the matching score “score”, x coordinate, y coordinate, rotation angle “angle”, scaling “scale”. stats: ZVOBJECT type, the matching status of each point of the template contour point, m rows n columns image type, one template contour per row, and the matching status of each contour point is stored sequentially on the row, that is, for a certain contour point, the matching score is ≥ the set score, it is 1 (matching is successful), when it is < the matching score, it is 0 (matching fails, if it is empty, the contour point matching status will not be output, if it is not empty, the contour point matching status will be output, this output parameter is combined with the drawing template command ZV_DRASHAPEMATCH, then matching success points and failure points can be drawn in different colors. minScore: the minimum matching score, (0,100], the higher the score, the more accurate the matching target. nums: the maximum number of matches, [0, infinity), when num is > the real target, output all targets with target scores from high to low, when num is < the real target, output num targets with target scores from high to low, when num is 0, output all targets with target scores from high to low. maxOverlap: the maximum overlap rate. Normal range is [0, 1], 0 – not overlap, that is, objects that are not overlapped are only found, 1 – overlap, that is, objects that are close are found. The overlap ratio indicates the allowed overlap ratio of the

target. The overlap rate is calculated by dividing the overlap area of the minimum external moment of the target contour feature by the area of the minimum external moment. When the overlapping part of the matching results exceeds, only the best result will be kept. It is mainly used to remove overlapping target objects. When there is only one target in the search map, the parameter can also be set smaller, which will speed up the matching speed but not obvious.

minThresh: the minimum edge threshold of the target contour. When minThresh is set to -1, the minimum threshold estimated from the template image will be used. If the difference between the template image and the matching image is large, it may cause the matching to fail. In this case, you should set the threshold yourself, such as setting it to 0.

accuracy: matching accuracy, 0-pixel accuracy, 1-interpolation accuracy, 2-least squares fitting accuracy, 3-multiple iterations least squares fitting accuracy. Accuracy 1 can meet most applications, 2 or 3 are used in occasions with higher accuracy requirements, but it will also be more time-consuming.

speed: matching speed 0-10, the bigger the speed, the faster, but may lose the target, when it is > 10, take 10

polar: matching polarity

polar	Polarity	Description
0	+	All contour points' light and dark changes of the matching target and the template are consistent.
1	±	Both + and - are OK, and all contour points' light and dark changes of the matching target and the template are consistent or inverse.
2	Any	All contour points' light and dark changes of the matching target and the template are consistent or inverse.

deform: deformation size, relative to the template, it allows

	a slight deformation of the target contour, 0 - does not support deformation. 1 - support slight deformation, but it is more time-consuming, and can be combined with the matching threshold to remove some noise interference to increase the speed. boundary: boundary mode, the extent to which the target contour exceeds the image boundary, 0-not exceeded, 1-a small amount exceeded, 2-moderate exceeded, 3-a large amount exceeded, 4-completely exceeded, and the matching time increases in sequence. When using the border, it needs to be used with the system parameter "ShapeOnBorder". The border mode will only work when this parameter is set to 1. Please set this value according to the actual situation.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Related Instruction	ZV_MCCREATESHAPE,

6.1.7.ZV_MCFINDSHAPES – Multiple Template Matching

Type	Shape template matching
Description	Use shape template list to match multiple template in image "img".
Grammar	ZV_MCFINDSHAPES (models, param, img, matchs, [minScore=0, nums=0]) models: ZVOBJECT type, shape template list param: ZVOBJECT type, parameter matrix, it can be empty, the number of rows is 1 or equal to the mods list length, the number of columns is less than or equal to 9, each column is the minimum score, quantity, maximum overlap rate, minimum threshold, accuracy, speed, polarity, deformation, boundary. If the number of columns is insufficient, the corresponding column will take the default value. If it is empty or the number of rows is 0, all will take the default value. If the number of rows is 1, all templates will share parameters. Otherwise, the number of rows must be equal to the length of the mods list, and the

	template uses corresponding row parameters. img: ZVOBJECT type, the search image to be matched, 8U single channel, size must be bigger than template image matchs: ZVOBJECT type, matching result, matrix type, n rows and 6 columns, each row has a matching target, and the columns are the matching score “score”, x coordinate, y coordinate, rotation angle “angle”, scaling “scale”. minScore: the minimum matching score, (0,100], the higher the score, the more accurate the matching target. When >0, it is for all matching targets, when = 0, “score” parameter in param is used. nums: the maximum number of matches, [0, infinity), when num is > the real target, output all targets with target scores from high to low, when num is < the real target, output num targets with target scores from high to low, when num is 0, when num is 0, the quantity parameter in param is used. The targets matched by each template are processed according to the corresponding quantity parameter. All retained targets are output in descending order of scores.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mod,img,re,matchImg,rlts,modlist,param ZV_READIMAGE(img, "model1.jpg", 0) 'read the image in the original image format ZV_READIMAGE(matchImg, "1.png", 0) ZV_MCCREATESHAPE(img,mod,re,-180,360,0,0,0,0) 'create template ZV_LISTINSERT(mod,modlist,0) ZV_READIMAGE(img, "model2.jpg", 0) ZV_MCCREATESHAPE(img,mod,re,-180,360,0,0,0,0) 'create template ZV_LISTINSERT(mod,modlist,1) ZV_MATGENCONST(param, 2, 6, 0) 'generate parameter matrix TABLE(0, 60, 0, 0, 1, 1, 9)

	ZV_MATSETROW(param, 0, 6, 0) ZV_MATSETROW(param, 1, 6, 0) ZV_MCSHAPEFINDS(mod,param,matchImg,rlts,90,0) 'template matching
Related Instruction	ZV_MCCREATESHAPE,

6.1.8.ZV_MCFINDSHAPESSTATE – Multi-Template Matching & Contour State Outputting

Type	Shape template matching
Description	Use shape template list to match multiple template in image "img", and output matching state of template contour point.
Grammar	ZV_MCFINDSHAPESSTATE (models, param, img, matchs, stat, [minScore=0, nums=0]) models: ZVOBJECT type, shape template list param: ZVOBJECT type, parameter matrix, it can be empty, the number of rows is 1 or equal to the mods list length, the number of columns is less than or equal to 9, each column is the minimum score, quantity, maximum overlap rate, minimum threshold, accuracy, speed, polarity, deformation, boundary. If the number of columns is insufficient, the corresponding column will take the default value. If it is empty or the number of rows is 0, all will take the default value. If the number of rows is 1, all templates will share parameters. Otherwise, the number of rows must be equal to the length of the mods list, and the template uses corresponding row parameters. img: ZVOBJECT type, the search image to be matched, 8U single channel, size must be bigger than template image matchs: ZVOBJECT type, matching result, matrix type, n rows and 6 columns, each row has a matching target, and the columns are the matching score "score", x coordinate, y coordinate, rotation angle "angle", scaling "scale". stats: ZVOBJECT type, the matching status of each point of the template contour point, m rows n columns image type, m is

	the longest template contour, one template contour per row, and the matching status of each contour point is stored sequentially on the row, that is, for a certain contour point, the matching score is ≥ the set score, it is 1 (matching is successful), when it is < the matching score, it is 0 (matching fails, if it is empty, the contour point matching status will not be output, if it is not empty, the contour point matching status will be output, this output parameter is combined with the drawing template command ZV_DRASHAPEMATCH, then matching success points and failure points can be drawn in different colors. minScore: the minimum matching score, (0,100], the higher the score, the more accurate the matching target. When >0, it is for all matching targets, when = 0, "score" parameter in param is used. nums: the maximum number of matches, [0, infinity), when num is > the real target, output all targets with target scores from high to low, when num is < the real target, output num targets with target scores from high to low, when num is 0, when num is 0, the quantity parameter in param is used. The targets matched by each template are processed according to the corresponding quantity parameter. All retained targets are output in descending order of scores.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Related Instruction	ZV_MCCREATESHAPE,

6.1.9.ZV_MCFINDSHAPESRE – Multiple Templates Match Supported Region

Type	Shape template matching
Description	Use shape template list to match multiple template in image "img".

Grammar	ZV_MCFINDSHAPESER (models, param, img, matchs, [minScore=0, nums=0]) models: ZVOBJECT type, shape template list param: ZVOBJECT type, parameter matrix, it can be empty, the number of rows is 1 or equal to the mods list length, the number of columns is less than or equal to 9, each column is the minimum score, quantity, maximum overlap rate, minimum threshold, accuracy, speed, polarity, deformation, boundary. If the number of columns is insufficient, the corresponding column will take the default value. If it is empty or the number of rows is 0, all will take the default value. If the number of rows is 1, all templates will share parameters. Otherwise, the number of rows must be equal to the length of the mods list, and the template uses corresponding row parameters. img: ZVOBJECT type, the search image to be matched, 8U single channel, size must be bigger than template image re: ZVOBJECT type, valid region of specified matching image matchs: ZVOBJECT type, matching result, matrix type, n rows and 6 columns, each row has a matching target, and the columns are the matching score "score", x coordinate, y coordinate, rotation angle "angle", scaling "scale". minScore: the minimum matching score, (0,100], the higher the score, the more accurate the matching target. When >0, it is for all matching targets, when = 0, "score" parameter in param is used. nums: the maximum number of matches, [0, infinity), when num is > the real target, output all targets with target scores from high to low, when num is < the real target, output num targets with target scores from high to low, when num is 0, when num is 0, the quantity parameter in param is used. The targets matched by each template are processed according to the corresponding quantity parameter. All retained targets are output in descending order of scores.
Controller	It is valid in controllers that support ZV function or they belong

	to 5XX series or above.
Example	ZVOBJECT mod,img,re,matchImg,rlts,modlist,param ZV_READIMAGE(img, "model1.jpg", 0) 'read the image in the original image format ZV_READIMAGE(matchImg, "1.png", 0) ZV_MCCREATESHAPE(img,mod,re,-180,360,0,0,0,0) 'create template ZV_LISTINSERT(mod,modlist,0) ZV_READIMAGE(img, "model2.jpg", 0) ZV_MCCREATESHAPE(img,mod,re,-180,360,0,0,0,0) 'create template ZV_LISTINSERT(mod,modlist,1) ZV_MATGENCONST(param, 2, 6, 0) 'generate parameter matrix TABLE(0, 60, 0, 0, 1, 1, 9) ZV_MATSETROW(param, 0, 6, 0) ZV_MATSETROW(param, 1, 6, 0) ZV_REGENFULLIMG (matchImg, re) ZV_MCSHAPEFINDSRE(mod,param,matchImg,re,rlts,90,0) ‘template matching
Related Instruction	ZV_MCCREATESHAPE,

6.1.10. ZV_MCFINDSHAPESRESTATE – Match Multi-Template Supported Region & Output Contour State

Type	Shape template matching
Description	Use shape template list to match multiple template in image “img”, and output matching state of template contour point.
Grammar	ZV_MCFINDSHAPESRESTATE (models, param, img, re, matchs, stat, [minScore=0, nums=0]) models: ZVOBJECT type, shape template list param: ZVOBJECT type, parameter matrix, it can be empty, the number of rows is 1 or equal to the mods list length, the number of columns is less than or equal to 9, each column is the

	minimum score, quantity, maximum overlap rate, minimum threshold, accuracy, speed, polarity, deformation, boundary. If the number of columns is insufficient, the corresponding column will take the default value. If it is empty or the number of rows is 0, all will take the default value. If the number of rows is 1, all templates will share parameters. Otherwise, the number of rows must be equal to the length of the mods list, and the template uses corresponding row parameters. img: ZVOBJECT type, the search image to be matched, 8U single channel, size must be bigger than template image re: ZVOBJECT type, valid region of specified matching image matchs: ZVOBJECT type, matching result, matrix type, n rows and 6 columns, each row has a matching target, and the columns are the matching score "score", x coordinate, y coordinate, rotation angle "angle", scaling "scale". stats: ZVOBJECT type, the matching status of each point of the template contour point, m rows n columns image type, m is the longest template contour, one template contour per row, and the matching status of each contour point is stored sequentially on the row, that is, for a certain contour point, the matching score is ≥ the set score, it is 1 (matching is successful), when it is < the matching score, it is 0 (matching fails, if it is empty, the contour point matching status will not be output, if it is not empty, the contour point matching status will be output, this output parameter is combined with the drawing template command ZV_DRASHAPEMATCH, then matching success points and failure points can be drawn in different colors. minScore: the minimum matching score, (0,100], the higher the score, the more accurate the matching target. When >0, it is for all matching targets, when = 0, "score" parameter in param is used. nums: the maximum number of matches, [0, infinity), when num is > the real target, output all targets with target scores from high to low, when num is < the real target, output num targets

	with target scores from high to low, when num is 0, when num is 0, the quantity parameter in param is used. The targets matched by each template are processed according to the corresponding quantity parameter. All retained targets are output in descending order of scores.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Related Instruction	ZV_MCCREATESHAPE,

6.1.11. ZV_MCSHAPECONTLIST – Get Template Contour

Type	Shape template creating
Description	It is used to get template contour.
Grammar	ZV_MCSHAPECONTLIST(model,contlist) model: ZVOBJECT type, source shape template contlist: ZVOBJECT type, template contour list
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	VOBJECT img, re, model, contlist ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_MCCCREATESHAPE(img,re,model,-180,360,0,0,0,0) 'create the template ZV_MCSHAPECONTLIST(model,contlist) 'get template contour

6.1.12. ZV_SHAPECREATE – Use Image to Create Template

Type	Shape template creating
Description	Create a shape-matching template through a template image. And the reference position of the created template is the integer part of half the size of the template image, that is, integer

	division (ImageWidth/2, ImageHeight/2)
Grammar	ZV_SHAPECREATE(img,model,angleStart,angleEnd,scaleMin, scaleMax,thresh[,levelNum=0,ptReduce=0,angleStep=0,scaleStep=0,m inContLen=20]) img: ZVOBJECT type, image for making templates, 8U single-channel. model: ZVOBJECT type, created template, output parameter angleStart: starting angle, determined by the image coordinate system, range [-360,360), if it exceeds the range, it will be automatically normalized to this range. angleEnd: angle range, range [-360,360). angleEnd must be ≥ to angleStart, otherwise an error will be reported, and the difference must be ≤ to 360, if it exceeds, angleEnd value will be cut out. After the template is created, targets within the starting and ending angles all can be matched. scaleMin: minimum ratio of matching zoom, range [0.5, 2.0] scaleMax: maximum ratio of matching zoom, range [0.5, 2.0], ≥ scale_min, after creating the template, targets within the minimum and maximum scaling ranges can be matched. thresh: contrast threshold for extracting edge contours--absolute threshold, range [0,255], when it is 0, an appropriate threshold will be selected internally, the greater the contrast, the stronger the strength of the extracted edge contour, this parameter can control the extraction of strong edges or weak edge, the smaller the threshold, the more weak-edges are extracted, and it may bring some noise at the same time. levelNum: the number of pyramid layers, range [0, infinity), the smaller the number of layers, the more time-consuming the matching, if it is 0 or the number of layers is too large, it will automatically select the appropriate number of layers, it is recommended to be 0 ptReduce: optimize to reduce the number of template points. If you set the greedy degree when searching for templates, it needs to set it lower, 0-no reduction, 1-slight reduction, 2-moderate reduction, 3-large reduction

	angleStep: angle step size, range [0,12]. The smaller the step size, the better the accuracy but the more time-consuming the matching. The larger the step size, the worse the accuracy but the less time-consuming the matching. It is unreasonable to set the step size too small or as when 0, an appropriate step size will be automatically selected internally, and 0 is recommended. Please attention since the angle needs to exceed 0, the angle step obtained by using the command ZV_SHAPEPARAM will be slightly different from angleStep. scaleStep: scaling step size, [0, scaleMax - scaleMin], the smaller the step size, the better the accuracy but the more time-consuming the matching, the larger the step size the worse the accuracy but the less time-consuming the matching, the step size is too small or when it is 0, an appropriate step size will be automatically selected internally, and it is recommended to be 0. Please attention since the zoom needs to exceed 1, the zoom step obtained by using the command ZV_SHAPEDEFPARAM will be slightly different from the scaleStep. minContLen: minimum contour length, contours smaller than this length will not be extracted, this parameter can control the deletion of some short contours. Notes: When creating a template, a target with a clear outline and unique feature is usually selected as a template, and the features should not be symmetric. The amount of template feature data is usually related to the template size and template parameters, and its data amount is proportional to the size of the template, the complexity of the template outline, the range of rotation angle, and the zoom range. The larger the template, the smaller the angle step used, and the larger the amount of feature data in the same angle range, so the matching time is more time-consuming. Creating a template and matching have a timeout mechanism, the default are 5000ms, when creating a template

	timeout, it can reduce the amount of template feature data by adjusting the template parameters appropriately (such as reducing the zoom range or using the outline point reduction parameter ptReduce, the default value of this parameter If it is 0, the contour points will not be simplified, and too serious reduction may affect the matching accuracy), or manually set the timeout period, such as ZV_SETSYSDBL ("ShapeCreateTimeout", 5000), ZV_SETSYSDBL ("ShapeFindTimeout", 5000). Creating a template also has a memory protection mechanism. When using an overly large template image to create a template with scaling, the memory occupied reaches the protection threshold. At this time, a memory error is reported and the template creation fails. It can reduce the amount of data by adjusting the template parameters, such as using the ptReduce parameter streamlines some points, or the threshold thresh is set to a larger point to only extract some obvious contour features.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, model ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_SHAPECREATE(img,model,0,360,1,1,50,0,0,0,0) 'create a template
Related Instruction	ZV_SHAPEFIND

6.1.13. ZV_SHAPECREATERE – Use Region to Create Template

Type	Shape template creating
Description	Use the template image "img" and specify the valid area "re" of the template image to create a shape matching template. This is mainly used when there is a lot of noise in the template image,

	creating a template by specifying some parts of the template image to be valid through the re area instead of the entire template image. create. For this method, it can remove the noisier part of the template image by performing some set operations or morphological operations on the re area, thereby obtaining a more robust template feature. The reference position of the created template is the integer part of half the size of the template image, that is, integer division (ImageWidth/2, ImageHeight/2)
Grammar	ZV_SHAPECREATERE (img, re, model, angleStart, angleEnd, scaleMin, scaleMax, thresh[,levelNum=0, ptReduce=0, angleStep=0, scaleStep=0, m inContLen]) img: ZVOBJECT type, image for making templates, 8U single-channel. re: ZVOBJECT type, specify the effective area of the template image, and the part corresponding to re in the template image will be used to create the template. re is an area based on run-length encoding, and its set operation is more convenient to remove invalid parts and retain valid parts to create templates. Usually, for template images with more noise or inconspicuous contour features, re is used to remove weak features in the template image and retain strong features. If re is empty, the entire template is the valid area by default. model: ZVOBJECT type, created template, output parameter angleStart: starting angle, determined by the image coordinate system, range [-360,360), if it exceeds the range, it will be automatically normalized to this range. angleEnd: angle range, range [-360,360). angleEnd must be ≥ to angleStart, otherwise an error will be reported, and the difference must be ≤ to 360, if it exceeds, angleEnd value will be cut out. After the template is created, targets within the starting and ending angles all can be matched. scaleMin: minimum ratio of matching zoom, range [0.5, 2.0] scaleMax: maximum ratio of matching zoom, range [0.5, 2.0], ≥ scale_min, after creating the template, targets within the

	minimum and maximum scaling ranges can be matched. thresh: contrast threshold for extracting edge contours--absolute threshold, range [0,255], when it is 0, an appropriate threshold will be selected internally, the greater the contrast, the stronger the strength of the extracted edge contour, this parameter can control the extraction of strong edges or weak edge, the smaller the threshold, the more weak-edges are extracted, and it may bring some noise at the same time. levelNum: the number of pyramid layers, range [0, infinity), the smaller the number of layers, the more time-consuming the matching, if it is 0 or the number of layers is too large, it will automatically select the appropriate number of layers, it is recommended to be 0 ptReduce: optimize to reduce the number of template points. If you set the greedy degree when searching for templates, it needs to set it lower, 0-no reduction, 1-slight reduction, 2-moderate reduction, 3-large reduction angleStep: angle step size, range [0,12]. The smaller the step size, the better the accuracy but the more time-consuming the matching. The larger the step size, the worse the accuracy but the less time-consuming the matching. It is unreasonable to set the step size too small or as when 0, an appropriate step size will be automatically selected internally, and 0 is recommended. Please attention since the angle needs to exceed 0, the angle step obtained by using the command ZV_SHAPEPARAM will be slightly different from angleStep. scaleStep: scaling step size, [0, scaleMax - scaleMin], the smaller the step size, the better the accuracy but the more time-consuming the matching, the larger the step size the worse the accuracy but the less time-consuming the matching, the step size is too small or when it is 0, an appropriate step size will be automatically selected internally, and it is recommended to be 0. Please attention since the zoom needs to exceed 1, the zoom step obtained by using the command ZV_SHAPEDEFPARAM will be slightly different from the scaleStep.

	minContLen: minimum contour length, contours smaller than this length will not be extracted, this parameter can control the deletion of some short contours. Notes: When creating a template, a target with a clear outline and unique feature is usually selected as a template, and the features should not be symmetric. The amount of template feature data is usually related to the template size and template parameters, and its data amount is proportional to the size of the template, the complexity of the template outline, the range of rotation angle, and the zoom range. The larger the template, the smaller the angle step used, and the larger the amount of feature data in the same angle range, so the matching time is more time-consuming. Creating a template and matching have a timeout mechanism, the default are 5000ms, when creating a template timeout, it can reduce the amount of template feature data by adjusting the template parameters appropriately (such as reducing the zoom range or using the outline point reduction parameter ptReduce, the default value of this parameter If it is 0, the contour points will not be simplified, and too serious reduction may affect the matching accuracy), or manually set the timeout period, such as ZV_SETSYSDBL ("ShapeCreateTimeout", 5000), ZV_SETSYSDBL ("ShapeFindTimeout", 5000). Creating a template also has a memory protection mechanism. When using an overly large template image to create a template with scaling, the memory occupied reaches the protection threshold. At this time, a memory error is reported and the template creation fails. It can reduce the amount of data by adjusting the template parameters, such as using the ptReduce parameter streamlines some points, or the threshold thresh is set to a larger point to only extract some obvious contour features.

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	**Example 1: when the re region is empty** ZVOBJECT img 'template image ZVOBJECT model 'template ZVOBJECT re 'specify the valid area of the template image. When only declaring "re" not to be processed, it is equivalent to re being empty, that is, the entire template image is valid, and the function is the same as ZV_SHAPECREATE ZV_READIMAGE(img, "test.png", 0) 'read the image in the original format ZV_SHAPECREATERE(img, re, model, angleStart, angleEnd, scaleMin, scale Max, thresh, levelNum, ptReduce, angleStep, scaleStep) 'create template **Example 2: when the re region is not empty** ZVOBJECT img 'template image ZVOBJECT model 'template ZVOBJECT re 'specify the valid area of the template image. It is necessary to generate re, that is, the part of the template image corresponding to re is used to create the template. ZV_READIMAGE(img, "test.png", 0) 'read the image in the original format ZV_REGENRECT(re,0,0,w,h) 'specify an area on the template image for creating the template, w and h are the width and height of the template image respectively ZV_SHAPECREATERE(img,re,model,0,360,1,1,120,0,0,0,0) 'create template
Related Instruction	ZV_SHAPEFIND

6.1.14. ZV_SHAPEFIND – Matching

Type	Shape template creating
Description	Use single-shape template to find and match in image “img”.
Grammar	ZV_SHAPEFIND(model, img, matchs, minScore [,nums=0, minDist=0, minThre sh=-1, accuracy=3, speed=9, polar=0]) model: ZVOBJECT type, shape template img: ZVOBJECT type, the search image to be matched, it cannot be 1:1 proportional to the template image, 8U single channel matches: ZVOBJECT type, matching result, matrix type, n rows and 5 columns, each row has a matching target, and the columns are the matching score “score”, x coordinate, y coordinate, rotation angle “angle”, scaling “scale” minScore: the minimum matching score, (0,100], the higher the score, the more accurate the matching target nums: the maximum number of matches, [0, infinity), when num is > the real target, output all targets with target scores from high to low, when num is < the real target, output num targets with target scores from high to low, when num is 0, output all targets with target scores from high to low. minDist: minimum matching distance, indicating the minimum distance allowed between two targets. When minDist is ≤ 0, the appropriate distance will be selected internally. When minDist is ≤ the distance between the two targets, the two targets will be matched. When minDist is > the distance between the two targets, the low-scoring target will be deleted and only the high-scoring target will be left and the value target is matched. minThresh: the lowest edge threshold of the target contour, when minThresh is < zero, the threshold when creating the template will be used accuracy: matching accuracy, 0-pixel accuracy, 1-interpolation accuracy, 2-least squares fitting accuracy, 3-multiple iterations least squares fitting accuracy. Accuracy 1

<table>
<tr>
<td></td>
<td>can meet most applications, 2 or 3 are used in occasions with higher accuracy requirements, but it will also be more time-consuming.
speed: matching speed 0-10, the bigger the speed, the faster, but may lose the target, when it is > 10, take 10
polar: matching polarity
<table>
<tr><td>polar</td><td>Polarity</td><td>Description</td></tr>
<tr><td>0</td><td>+</td><td>All contour points' light and dark changes of the matching target and the template are consistent.</td></tr>
<tr><td>1</td><td>±</td><td>Both + and - are OK, and all contour points' light and dark changes of the matching target and the template are consistent or inverse.</td></tr>
<tr><td>2</td><td>Any</td><td>All contour points' light and dark changes of the matching target and the template are consistent or inverse.</td></tr>
</table>
</td>
</tr>
<tr>
<td>Controller</td>
<td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td>
</tr>
<tr>
<td>Example</td>
<td>

template

image to be matched
ZVOBJECT mod,img,matchImg,clrImg,rlts,
ZVOBJECT matRigid,modContList,dstContList
ZV_READIMAGE(img, "model.jpg", 0)
'read the image in the original image format
ZV_SHAPECREATE(img,mod,0,360,1,1,50,0,0,0,0)
'create template
ZV_SHAPECONTOURS(mod, modContList, 0)
'get template outline
ZV_READIMAGE(matchImg, "1.png", 0)
'read the image in the original format
</td>
</tr>
</table>

	ZV_SHAPEFIND(mod,matchImg,rlts,90,1,0,-1,3,9,0) 'template matching ZV_MATGETROW(rlts,0,5,0) 'obtain the first row of the matching result matrix, which are: matching score, x coordinate, y coordinate, rotation angle "angle", and scaling "scale" ZV_GETRIGIDVECTOR (matRigid, 0, 0, 0, TABLE(1), TABLE(2), TABLE(3)) 'calculate rigid transformation matrix ZV_CONTAFFINE (modContListt, matRigid, dstContList) 'contour affine transformation ZV_GRAYTORGB(matchImg,clrImg) 'convert grayscale image to RGB image ZV_CONTLIST (clrImg, dstContList, ZV_COLOR(0,255,0),0) 'draw the outline
Related Instruction	ZV_SHAPECREATE, ZV_SHAPECREATERE

6.1.15. ZV_SHAPEFINDST – Match & Output Contour State

Type	Shape template creating
Description	Use single-shape template to find and match in image "img", also output the matching state of template contour point.
Grammar	ZV_SHAPEFINDST (model, img, matchs, stats, minScore [,nums=0, minDist=0 , minThresh=-1, accuracy=3, speed=9, polar=0]) model: ZVOBJECT type, shape template img: ZVOBJECT type, the search image to be matched, it cannot be 1:1 proportional to the template image, 8U single channel matches: ZVOBJECT type, matching result, matrix type, n rows and 5 columns, each row has a matching target, and the columns are the matching score "score", x coordinate, y coordinate, rotation angle "angle", scaling "scale". stats: ZVOBJECT type, the matching status of each point of

	the template contour point, m x n image type, one template contour per row, and the matching status of each contour point is stored sequentially on the row, that is, for a certain contour point, the matching score is ≥ the set score, it is 1 (matching is successful), when it is < the matching score, it is 0 (matching fails, if it is empty, the contour point matching status will not be output, if it is not empty, the contour point matching status will be output, this output parameter is combined with the drawing template command ZV_DRASHAPEMATCH, then matching success points and failure points can be drawn in different colors. minScore: the minimum matching score, (0,100], the higher the score, the more accurate the matching target. nums: the maximum number of matches, [0, infinity), when num is > the real target, output all targets with target scores from high to low, when num is < the real target, output num targets with target scores from high to low, when num is 0, output all targets with target scores from high to low. minDist: the minimum distance, when it is 0, the distance is automatically selected, indicating the allowable separation distance of the matching results. When the matching result distance is less than the minimum distance, only the best result will be kept. It is mainly used to remove overlapping target objects and is recommended to be 0 when matching non-overlapping targets. When there is only one target in the search graph, it can also set the minimum distance larger, which will speed up the matching but not obvious. minThresh: the lowest edge threshold of the target contour, when minThresh is < 0, the threshold when creating the template will be used accuracy: matching accuracy, 0-pixel accuracy, 1-interpolation accuracy, 2-least squares fitting accuracy, 3-multiple iterations least squares fitting accuracy. Accuracy 1 can meet most applications, 2 or 3 are used in occasions with higher accuracy requirements, but it will also be more time-

<table>
<tr><td></td><td>consuming.
speed: matching speed 0-10, the bigger the speed, the faster, but may lose the target, when it is > 10, take 10
polar: matching polarity
<table>
<tr><td>polar</td><td>Polarity</td><td>Description</td></tr>
<tr><td>0</td><td>+</td><td>All contour points' light and dark changes of the matching target and the template are consistent.</td></tr>
<tr><td>1</td><td>±</td><td>Both + and - are OK, and all contour points' light and dark changes of the matching target and the template are consistent or inverse.</td></tr>
<tr><td>2</td><td>Any</td><td>All contour points' light and dark changes of the matching target and the template are consistent or inverse.</td></tr>
</table></td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><td>Example</td><td>

template

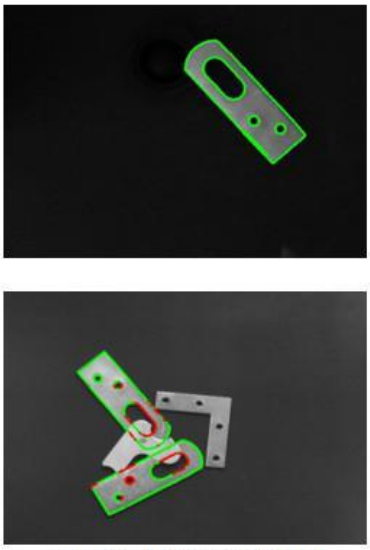


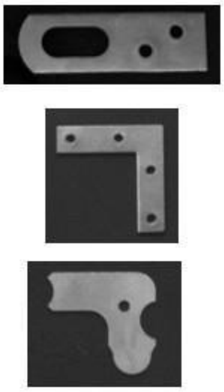
image to be matched

ZVOBJECT mod, img, matchImg, clrImg, rlts, stats
ZV_READIMAGE(img, "model.jpg", 0)
'read the image in the original image format
ZV_SHAPECREATE(img,model,0,360,1,1,50,0,0,0,0)
'create template
ZV_READIMAGE(matchImg, "1.png", 0)</td></tr>
</table>

	'read the image in the original image format ZV_SHAPEFINDST (model, matchImg, rlts, stats, 90, 1, 0, -1, 3, 9, 0) 'template matching ZV_GRAYTORGB (matchImg, clrImg) 'convert grayscale image to RGB image ZV_DRASHAPEMATCH (clrImg, model, rlts, stats, ZV_COLOR(0,255,0), ZV_CO LOR(255,0,0)) 'draw the template on the color image, and the contour points that match successfully are drawn in green, and the contour points that fail to match are drawn red
Related Instruction	ZV_SHAPECREATE, ZV_SHAPECREATERE, ZV_DRASHAPEMATCH

6.1.16. ZV_SHAPEFINDS – Multi-Template Matching

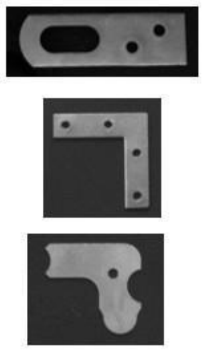
Type	Shape template creating
Description	Use multiple shape templates to find and match in image “img”.
Grammar	ZV_SHAPEFINDS (modelList, param, img, matchs [,numMatchs = 0, minDis = 0]) modelList: ZVOBJECT type, shape template list param: ZVOBJECT type, matching parameter, matrix type, 1 row or n rows, when it is row 1, all templates share the matching parameters of this row, when it is n rows, the number of rows of the matrix must be equal to the length of the template list, indicating that each row corresponds to matching parameter of one template, and the data in each row are minScore, nums, minDist, minThresh, accuracy, speed, and polar. For parameter meanings, please refer to the single template matching parameters. img: ZV_OBJECT type, the search image to be matched, 8U channel matches: ZVOBJECT type, matching result, matrix type, output parameters, matrix with n rows and 6 columns, one matching result per row, and the data in each row are score, x, y,

	angle, scale, modelId in turn. “modelId” represents matched result that corresponds to id template in template list. numMatchs: the total maximum number of matches, range [0, infinity), when numMatchs is 0, use the parameters in param, and output targets in ascending order of template id (if the id is the same, score descending order), when numMatchs is > 0, output in descending order of scores. For relationship between snumMatchs size and real target size, please refer to single-template matching. minDist: minimum matching distance, indicating the minimum distance allowed between two targets. When minDist is ≤ 0, use the parameters in param. For the target corresponding to the template, the meaning refers to single template matching. When minDist is > 0, for all matching targets.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 template list  image to be matched ZVOBJECT mod, mod1, mod2, mod3, modList, img1, img2, img3, rlts ZVOBJECT param, matRigid, modContList, dstContList, matchImg, clrImg ZV_READIMAGE(img1, "1.jpg", 0) 'read the image in the original image format ZV_READIMAGE(img2, "2.jpg", 0) ZV_READIMAGE(img3, "3.jpg", 0) ZV_SHAPECREATE(img1,mod1,0,360,1,1,50,0,0,0,0) 'create template

	ZV_SHAPECREATE(img2,mod2,0,360,1,1,90,0,0,0,0) ZV_SHAPECREATE(img3,mod3,0,360,1,1,77,0,0,0,0) ZV_CLEAR(modList) 'clear model_list ZV_LISTINSERT(mod1, modList, -1) 'insert the template into the template list ZV_LISTINSERT(mod2, modList, -1) ZV_LISTINSERT(mod3, modList, -1) ZV_MATGENCONST(param,3,7,0) 'construct a matrix with 3 rows and 7 columns, and set the matching parameters TABLE(0,90,1,0,-1,3,9,0) ZV_MATSETROW(param,0,7,0) TABLE(0,90,1,0,-1,3,9,0) ZV_MATSETROW(param,1,7,0) TABLE(0,90,1,0,-1,3,9,0) ZV_MATSETROW(param,2,7,0) ZV_READIMAGE(matchImg, "test.png", 0) 'read the image in the original image format ZV_SHAPEFINDS(modList,param,matchImg,rlts,0,0) 'multiple target matching ZV_LISTGET(modList,mod,TABLE(5)) ZV_SHAPECONTOURS(mod, modContList, 0) 'get template contour ZV_GETRIGIDVECTOR(matRigid,0,0,0,TABLE(1),TABLE(2),TABLE(3)) 'calculate rigid transformation matrix ZV_CONTAFFINE(modContListt, matRigid,dstContList) 'contour affine transformation ZV_GRAYTORGB(matchImg,clrImg) 'convert grayscale image to RGB image ZV_CONTLIST(clrImg,dsContList,ZV_COLOR(0,255,0),0) 'draw the contour
Related Instruction	ZV_SHAPECREATE, ZV_SHAPECREATERE

6.1.17. ZV_SHAPEFINDSST – Multi-Template Matching & Contour State Outputting

Type	Shape template creating
Description	Use multiple shape templates to find and match in image "img", also output the matching state of template contour point.
Grammar	ZV_SHAPEFINDSST (modelList, param, img, matchs, stats, [,numMatchs=0,mi nDis=0]) modelList: ZVOBJECT type, shape template list param: ZVOBJECT type, matching parameter, matrix type, 1 row or n rows, when it is row 1, all templates share the matching parameters of this row, when it is n rows, the number of rows of the matrix must be equal to the length of the template list, indicating that each row corresponds to matching parameter of one template, and the data in each row are minScore, nums, minDist, minThresh, accuracy, speed, and polar. For parameter meanings, please refer to the single template matching parameters. img: ZVOBJECT type, the search image to be matched, 8U single channel matches: ZVOBJECT type, matching result, matrix type, n rows and 6 columns, each row has a matching target, and the data in each row are score, x, y, angle, scale, modelId in turn. "modelId" represents matched result that corresponds to id template in template list. stats: ZVOBJECT type, the matching status of each point of the template contour point, m x n image type, one template contour per row, and the matching status of each contour point is stored sequentially on the row, that is, for a certain contour point, the matching score is ≥ the set score, it is 1 (matching is successful), when it is < the matching score, it is 0 (matching fails, if it is empty, the contour point matching status will not be output, if it is not empty, the contour point matching status will be output, this output parameter is combined with the drawing

	template command ZV_DRASHAPEMATCH, then matching success points and failure points can be drawn in different colors. numMatchs: the total maximum number of matches, range [0, infinity), when numMatchs is 0, use the parameters in param, and output targets in ascending order of template id (if the id is the same, score descending order), when numMatchs is > 0, output in descending order of scores. For relationship between snumMatchs size and real target size, please refer to single-template matching. minDist: minimum matching distance, indicating the minimum distance allowed between two targets. When minDist is ≤ 0, use the parameters in param. For the target corresponding to the template, the meaning refers to single template matching. When minDist is > 0, for all matching targets.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 template list  image to be matched DIM rows ZVOBJECT mod, mod1, mod2, mod3, modList, img1, img2, img3, rlts, subRlts ZVOBJECT param, matRigid, modContList, dstContList, matchImg, clrImg ZV_READIMAGE(img1, "1.jpg", 0) 'read the image in the original format ZV_READIMAGE(img2, "2.jpg", 0) ZV_READIMAGE(img3, "3.jpg", 0) ZV_SHAPECREATE(img1,mod1,0,360,1,1,50,0,0,0,0)

	'create template ZV_SHAPECREATE(img2,mod2,0,360,1,1,90,0,0,0,0) ZV_SHAPECREATE(img3,mod3,0,360,1,1,77,0,0,0,0) ZV_CLEAR(modList) 'clear model_list ZV_LISTINSERT(mod1, modList, -1) 'insert the template into the template list ZV_LISTINSERT(mod2, modList, -1) ZV_LISTINSERT(mod3, modList, -1) ZV_MATGENCONST(param,3,7,0)' construct a matrix with 3 rows and 7 columns, and set the matching parameter TABLE(0,90,1,0,-1,3,9,0) ZV_MATSETROW(param,0,7,0) TABLE(0,90,1,0,-1,3,9,0) ZV_MATSETROW(param,1,7,0) TABLE(0,90,1,0,-1,3,9,0) ZV_MATSETROW(param,2,7,0) ZV_READIMAGE(matchImg, "test.png",0) 'read the image in the original image format ZV_SHAPEFINDSST(modList,param,matchImg,rlts,stats,0,0) 'multiple targets matching ZV_MATGETROW(rlts,0,6,0) 'obtain first row of matched result matrix ZV_LISTGET(modList,model,TABLE(5)) ‘get first row result matrix corresponding template rows = ZV_MATROWS(rlts) 'get the number of rows of the result matrix ZV_MATGETSUB(rlts,subRlts,0,0,5,rows) 'intercept result matrix ZV_GRAYTORGB(matchImg,clrImg) 'convert grayscale image to RGB image ZV_DRASHAPEMATCH (clrImg, modList, subRlts, stats, ZV_COLOR (0,255,0), ZV_COLOR(255,0,0)) ‘draw the template list on the color image, and the contour points that match successfully are drawn in

	green, and the contour points that fail to match are drawn in red.
Related Instruction	ZV_SHAPECREATE, ZV_SHAPECREATERE

6.1.18. ZV_SHAPECONTOURS – Get Template Contour

Type	Shape template creating
Description	It is used to get the template outline on the specified layer pyramid.
Grammar	ZV_SHAPECONTOURS(model,contlist,level) alias: ZV_SHAPECONTLIST model: ZVOBJECT type, source shape template contlist: ZVOBJECT type, output parameters, template outline list level: pyramid layer No.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, model, contlist ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_SHAPECREATE(img,model,0,360,1,1,50,0,0,0,0) 'create template ZV_SHAPECONTOURS(model,contlist,0) 'get the template outline on the 0th layer pyramid

6.1.19. ZV_SHAPETEMPL – Get Template Image

Type	Shape template creating
Description	It is used to create shape template image.
Grammar	ZV_SHAPETEMPL(model,img) model: ZVOBJECT type, source shape template img: ZVOBJECT type, output parameter, image used to create shape templates

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, model, dst ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_SHAPECREATE(img,model,0,360,1,1,50,0,0,0,0) 'create template ZV_SHAPECONTOURS(model,dst) 'obtain the image that creates shape template "model" and save it into dst

6.1.20. ZV_SHAPEREGION – Get Template Region

Type	Shape template creating
Description	It is used to obtain valid region when creating the shape template.
Grammar	ZV_SHAPEREGION(model,re) model: ZVOBJECT type, source shape template re: ZVOBJECT type, output parameters, valid region when creating shape template.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img 'template image ZVOBJECT model 'template ZVOBJECT re 'specify the valid region of the template image, and re needs to be generated, that is, the part of the template image corresponding to re is used to create the template ZVOBJECT re_dst ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_REGENRECT(re,0,0,w,h) 'specify an area on the template image for

	creating the template, w and h are the width and height of the template image respectively ZV_SHAPECREATERE(img,re,model,0,360,1,1,120,0,0,0,0) 'create template ZV_SHAPEREGION(model,re_dst) 'Obtain the valid region when creating the shape template model and store it in re

6.1.21. ZV_SHAPETEMPSIZE – Get Template Image Size

Type	Shape template creating
Description	It is used to obtain the image size that is for creating shape template.
Grammar	ZV_SHAPETEMPLSIZE(model, tabId) model: ZVOBJECT type, source shape template tabId: TABLE index, output parameter, the image size used to create the template, in turn width, height
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, model ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_SHAPECREATE(img,model,0,360,1,1,50,0,0,0,0) 'create template ZV_SHAPETEMPLSIZE(model,0) 'get the image size of the created shape template and stores it in TABLE(0)

6.1.22. ZV_SHAPEPARAM – Get Template Parameters

Type	Shape template creating
Description	It is used to obtain parameters when creating shape template.

Grammar	ZV_SHAPEPARAM(model,tabId) model: ZVOBJECT type, source shape template tabId: TABLE index, output parameters, obtained template parameters, in order of angleStart, angleEnd, scaleMin, scaleMax, thresh, levelNum, that is, start angle, end angle, minimum zoom, maximum zoom, edge threshold, number of pyramid layers
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, model ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_SHAPECREATE(img,model,0,360,1,1,50,0,0,0,0) 'create template ZV_SHAPEPARAM(model,0) 'save template parameters into TABLE (0)

6.1.23. ZV_SHAPEDEFPARAM – Get Template Default Parameters

Type	Shape template creating
Description	It is used to obtain default parameters of shape template.
Grammar	ZV_SHAPEDEFPARAM(model,tabId) model: ZVOBJECT type, source shape template tabId: TABLE index, output parameters, obtained default parameters, which are angleStep, scaleStep, minThresh, ptReduce in order, that is, angle step, scaling step, minimum edge threshold, and contour point simplification level
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, model ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_SHAPECREATE(img,model,0,360,1,1,50,0,0,0,0)

	'create template ZV_SHAPEDEFPARAM(model,0) 'save template default parameters into TABLE (0)

6.2.NCC Matching

Based on the matching algorithm of normalized grayscale correlation coefficient, the matching process uses a pyramid system to improve efficiency, and the origin of the template is the center of the image.

6.2.1.ZV_NCCCREATERE – Create

Type	NCC matching.
Description	Use the template image and specify the effective area re of the template image to create a template, mainly for when there are many noises in the template image, using the re area to specify that some parts of the template image are valid to create a template instead of using the entire template image. Through "re", the part with more noise in the template image can be removed by performing some set operations or morphological operations on the re area, so as to obtain a more robust template feature.
Grammar	ZV_NCCCREATERE(img,re,model,angleStart,angleEnd[,levelNum =0,angl eStep=0]) img: ZVOBJECT type, image for making templates, 8U single channel re: ZVOBJECT type, valid region selected by the template image, based on run-length encoding model: ZVOBJECT type, output parameters, NCC template made angleStart: starting value of angle matching, clockwise is + angleEnd: ending value of angle matching, clockwise is + levelNum; the number of pyramid layers, > 0, = 0 means

	automatically select the number of layers angleStep: angle step, > 0, = 0 means automatically select the angle step
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, re, model ZV_READIMAGE(img, "model.png", 0) 'read the image in the original image format ZV_REGENRECT2(re,422,181,290,100,50) 'specify an area on the template image used to create the template ZV_NCCCREATERE(img,re,model,-180,180,5,0) 'create NCC template, angle range -180 to 180, 5-layer pyramid, angle step size is automatically selected by the system

6.2.2.ZV_NCCFIND – Match

Type	NCC matching.
Description	It uses NCC template to find and match in image "img".
Grammar	ZV_NCCFIND(model,img,matchs,minScore[,nums=0,minDist=0, isSubpix=1, polar=0]) model: ZVOBJECT type, NCC template img: ZVOBJECT type, search for matching target image, cannot be 1:1 equal to the template image, 8U single channel matches: ZVOBJECT type, matching result, matrix type, one matching target is in each row, and the 4 columns are score, x coordinate, y coordinate, and rotation angle. minScore: minimum matching score, > 0, (0,100] nums: maximum number of matches, take the first nums results with the highest score, if it is 0, take all results minDist: the minimum distance between two matching results, when it is 0, automatically select distance isSubpix: whether to interpolate with sub-pixel precision, 0-

<table>
<tr>
<td></td>
<td>no, 1-yes
polar: match polarity
<table>
<tr><th>polar</th><th>Polarity</th><th>Description</th></tr>
<tr><td>0</td><td>±</td><td>The light-dark transformation of the template and the target are the same.</td></tr>
<tr><td>1</td><td>Any</td><td>The light-dark transformation of the template and the target are the same or the opposite.</td></tr>
</table>
</td>
</tr>
<tr>
<td>Controller</td>
<td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td>
</tr>
<tr>
<td>Example</td>
<td>
template

image to be matched
ZVOBJECT img, clrImg, re, model, matchImg
ZV_READIMAGE(img, "model.png", 0)
'read the image in the original format
ZV_REGENFULLIMG(img,re)
'generate the area covering the whole image
ZV_NCCCREATERE(img,re,model,-180,180,5,0)
'create ncc template
ZV_READIMAGE(match_img, "test.png", 0)
'read the image in the original image format
ZV_NCCFIND(model, matchImg, results, 80, 10, 20, 1)
'NCC match
ZV_GRAYTORGB(matchImg,clrImg)
'convert grayscale image into RGB image
ZV_MATINFO(results, 0)
FOR i = 0 TO TABLE(0)-1
ZV_MATGETROW(results, i, 4, 10)
ZV_MARKER(clrImg,TABLE(11),TABLE(12),0,20,ZV_COLOR(255,0,0))</td>
</tr>
</table>

	'matching points draw red cross marks NEXT

6.2.3.ZV_NCCTEMPL – Get Template Image

Type	NCC matching.
Description	Obtain the image when creating NCC template.
Grammar	ZV_NCCTEMPL(model,img) model: ZVOBJECT type, NCC template img: ZVOBJECT type, output parameter, obtained image
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, re, model, dst ZV_READIMAGE(img, "model.png", 0) 'read the image in the original image format ZV_REGENFULLIMG(img,re) 'generate an area covering the entire image ZV_NCCCREATERE(img,re,model,-180,180,0,0) 'create ncc template ZV_NCCTEMPL(model,dst) 'get the image when creating the NCC template and store it in img
Related Instruction	ZV_NCCCREATERE

6.2.4.ZV_NCCREGION – Get Template Region

Type	NCC matching.
Description	Obtain valid region when creating NCC template.
Grammar	ZV_NCCREGION(model, re) model: ZVOBJECT type, NCC template re: ZVOBJECT type, output parameter, obtained region of created template
Controller	It is valid in controllers that support ZV function or they belong

	to 5XX series or above.
Example	ZVOBJECT img, re, model, dst_re ZV_READIMAGE(img, "model.png", 0) 'read the image in the original image format ZV_REGENFULLIMG(img,re) 'generate an area covering the entire image ZV_NCCCREATERE(img,re,model,-180,180,0,0) 'create ncc template ZV_NCCTEMPL(model,dst_re) 'get the region when creating the NCC template and store it in re
Related Instruction	ZV_NCCCREATERE

6.2.5.ZV_NCCPARAM – Get Template Parameters

Type	NCC matching.
Description	Obtain parameters when creating NCC template.
Grammar	ZV_NCCREGION(model, tabId) model: ZVOBJECT type, source NCC template tabId: TABLE index, output parameters, obtained NCC template parameters, in order angleStart, angleEnd, angleStep, levelNum, that is, starting angle, ending angle, angle step, and number of pyramid levels
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, re, model ZV_READIMAGE(img, "model.png", 0) 'read the image in the original image format ZV_REGENFULLIMG(img,re) 'generate an area covering the entire image ZV_NCCCREATERE(img,re,model,-180,180,0,0) 'create ncc template ZV_NCCTEMPL(model,0) 'get ncc template parameters and store into TABLE (0)

Related Instruction	ZV_NCCCREATERE

6.3.Grayscale Matching

Based on the matching of image gray value, the origin of the template is the center of the image.

6.3.1.ZV_FASHTEMPL – Fast to Match

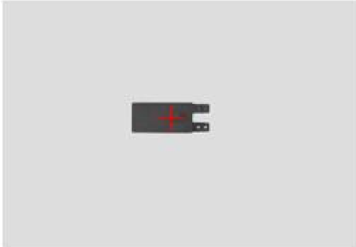
<table>
<tr><th>Type</th><td>NCC matching.</td></tr>
<tr><th>Description</th><td>Get the x, y coordinates of the best matching position, an integer value.</td></tr>
<tr><th>Grammar</th><td>ZV_FASTTEMPL(img,modImg,tabId[,method = 0])
img: ZVOBJECT type, image to be matched
modImg: ZVOBJECT type, template image
tabId: TABLE index, matching result, output parameters, x, y coordinates in order, coordinates are integer values
Metho: matching algorithm
<table>
<tr><td>Method</td><td>Description</td></tr>
<tr><td>0</td><td>normalized correlation coefficient</td></tr>
<tr><td>1</td><td>correlation coefficient</td></tr>
</table></td></tr>
<tr><th>Controller</th><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><th>Example</th><td>
template

image to be matched
ZVOBJECT model, matchImg, clrImg
ZV_READIMAGE(model, "model.png", 0)
'read the image in the original image format</td></tr>
</table>

<table>
<tr><td></td><td>ZV_READIMAGE(matchImg, "test.png", 0)
'read the image in the original image format
ZV_FASTTEMPL(matchImg,model,0,0)
'use the grayscale matching method to obtain the position of the template image in the search image, and store the coordinate position in TABLE (0).
ZV_GRAYTORGB(matchImg,clrImg)
'convert grayscale image to RGB image
ZV_MARKER(clrImg,TABLE(0),TABLE(1),0,50,zv_color(255,0,0))
'draw cross</td></tr>
</table>

6.3.2.ZV_BESTTEMPL – Match Grayscale Template

<table>
<tr><td>Type</td><td>NCC matching.</td></tr>
<tr><td>Description</td><td>Get the best matching position, supporting sub-pixel accuracy.</td></tr>
<tr><td>Grammar</td><td>ZV_BESTTEMPL(img,modImg,minScore,tabId[,isSubpix=0, polar = 0])
img: ZVOBJECT type, the image to be matched, the image is a single-channel image
modImg: ZVOBJECT type, the template image
minScore: minimum matching score
tabId: TABLE index, matching result, output parameters, in order of score, x, y
isSubpix: whether sub-pixel precision interpolation, 0-no, 1-yes
polar: match polarity
<table>
<tr><td>polar</td><td>Polarity</td><td>Description</td></tr>
<tr><td>0</td><td>±</td><td>The light-dark transformation of the template and the target are the same.</td></tr>
<tr><td>1</td><td>Any</td><td>The light-dark transformation of the template and the target are the same or the opposite.</td></tr>
</table></td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
</table>

Example	 template  image to be matched ZVOBJECT model, matchImg, clrImg ZV_READIMAGE(model, "model.png", 0) 'read the image in the original image format ZV_READIMAGE(matchImg, "test.png", 0) 'read the image in the original image format ZV_BESTTEMPL(matchImg,60,0,0,0) 'use the grayscale matching method to obtain the best position and store the coordinate position in TABLE (0). ZV_GRAYTORGB(matchImg,clrImg) 'convert grayscale image to RGB image ? TABLE (0) 'match the score ZV_MARKER(clrImg,TABLE(0),TABLE(1),TABLE(2),0,50,zv_color(255,0,0)) 'draw cross

6.3.3.ZV_MULTITEMPL – Match Grayscale Template

Type	Grayscale matching
Description	Multi-target grayscale matching, search for targets matching the template in the search image, and return the top nums matching results with scores that are greater than minScore.
Grammar	ZV_MULTITEMPL (img, modImg, matRst, minScore, [num=0, minDist = 0, polar = 0]) img: ZVOBJECT type, the image to be matched modImg: ZVOBJECT type, the template image matRst: ZVOBJECT type, matching result, matrix type, N rows and 3 columns, one result in each row, the order of the results is score, x coordinate and y coordinate. minScore: minimum matching score

<table>
<tr><td></td><td>
nums: the maximum number of matches, take the first nums results with the highest score, if it is 0, take all the results
minDist: the minimum distance between two matching results, ≥ 0, if = 0, the distance is automatically selected.
isSubpix: whether sub-pixel precision interpolation
polar: match polarity
<table>
<tr><th>polar</th><th>Polarity</th><th>Description</th></tr>
<tr><td>0</td><td>±</td><td>The light-dark transformation of the template and the target are the same.</td></tr>
<tr><td>1</td><td>Any</td><td>The light-dark transformation of the template and the target are the same or the opposite.</td></tr>
</table>
</td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><td>Example</td><td>

template

image to be matched
ZVOBJECT model, matchImg, clrImg, results
ZV_READIMAGE(model, "model.png", 0)
'read the image in the original image format
ZV_READIMAGE(matchImg, "test.png", 0)
'read the image in the original image format
ZV_MULTITEMPL(matchImg,model,results,90,1,0,0,0)
'grayscale template matching, generating matching matrix results
ZV_MATGETROW(results,0,3,0)
ZV_GRAYTORGB(matchImg,clrImg)
'convert grayscale image to RGB image
? TABLE(0)' match score
ZV_MARKER(clrImg,TABLE(1),TABLE(2),0,50,zv_color(255,0,0))
'draw a cross
</td></tr>
</table>

Chapter VII Measurement

7.1.Measurer Generation

7.1.1.ZV_MRGENRECT – Generate Rectangle Measurer

Type	Measurement region
Description	Generate a rectangular point measurer and a single-area measurer. The measurer cannot exceed the image range, otherwise an error will be reported.
Grammar	ZV_MRGENRECT(mr,x,y,w,h) mr: ZVOBJECT type, rectangular area measurer x: upper left x coordinate of the rectangular area, range [0,32766] y: upper left y coordinate of the rectangular area, range [0,32766] w: width of the rectangular area, range [1,32766] h: height of rectangular area, range [1,32766]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mr ZV_MRGENRECT(mr,20,20,100,100) 'generate a rectangular measurer

7.1.2.ZV_MRGENRECT2 – Generate Rotate Rectangle Measurer

Type	Measurement region
Description	Generate a rotate rectangular point measurer and a single-area measurer. The measurer cannot exceed the image range, otherwise an error will be reported.

Grammar	ZV_MRGENRECT2(mr,cx,cy,w,h,angle,interp) mr: ZVOBJECT type, rotated rectangular area measurer cx: the x coordinate of the center of the rotating rectangle, range [0,32766] cy: the y coordinate of the center of the rotating rectangle, range [0,32766] w: rotated rectangle width, range [1, 32766] h: rotated rectangle height, range [1, 32766] angle: angle of the rotating rectangle, + clockwise, the unit is degree, range (-180, 180], if it exceeds the range, it will be automatically normalized to this range interp: interpolation algorithm, range [0,3], 0-nearest neighbor interpolation, 1-bilinear interpolation, 2-bi-cubic interpolation, 3-LANCZOS, common value 1
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mr ZV_MRGENRECT2(mr,120,120,100,100,60,0) 'generate a rotated rectangular measurer

7.1.3.ZV_MRGENARC – Generate Arc Measurer

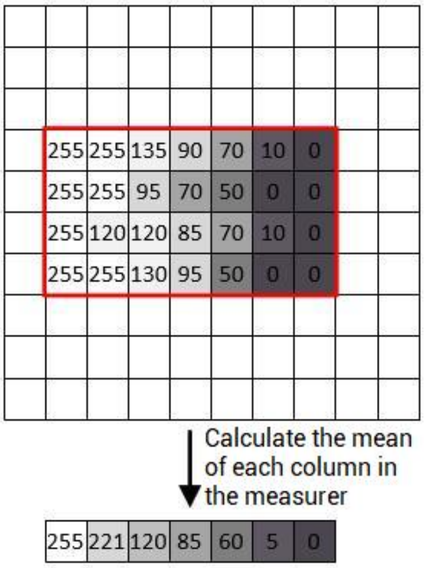
Type	Measurement region
Description	Generate an arc point measurer and a single-area measurer. The measurer cannot exceed the image range, otherwise an error will be reported.
Grammar	ZV_MRGENARC(mr,cx,cy,r,annR[,starAngle=0,extAngle=360,direction=0 , interp=1]) mr: ZVOBJECT type, circular measurer cx: center x coordinate of the arc, range [0,32766] cy: the y coordinate of the center of the arc, range [0,32766] r: the radius of the centerline of the arc, range [1, 16383] annR: half-width of the arc, range (0,r) startAngle: starting angle of the measurement area,

	determined by the image coordinate system, the unit is degree, range (-180,180], if it exceeds the range, it will automatically normalize to this range extAngle: angle range of the measurement area, (0, 360], the unit is degree, if it is > 360, it will automatically convert to 360 internally direction: scan direction, 0-tangential clockwise, 1-radial from outside to inside interp: interpolation algorithm, range [0,3], 0-nearest neighbor interpolation, 1-bilinear interpolation, 2-bi-cubic interpolation, 3-LANCZOS
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mr ZV_MRGENARC(mr,120,120,20,10,60,120,0,1) 'generate an arc measurer

7.2. Single Area Measurement

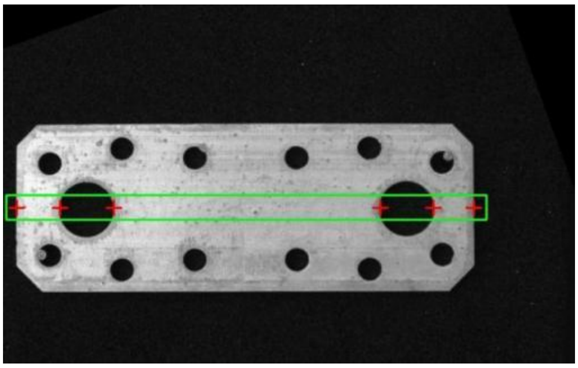
7.2.1. ZV_MRPROJECTION – Grayscale Projection

Type	Single area measurement
Description	The grayscale distribution of the measurement area is along the width direction or along the positive direction of the tangential angle, and the arc is along the specified direction. The circumscribed moment of the measurement area cannot exceed the boundary of the measured image.

	
Grammar	ZV_MRPROJECTION (mr,img,matProj) mr: ZVOBJECT type, single measurement area img: ZVOBJECT type, measured image, single-channel image matProj: ZVOBJECT type, grayscale distribution of the image measurement area, matrix type
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mr,img,result ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_MRGENRECT(mr,10,142,637,262) 'generate a rectangular measurer ZV_MRPROJECTION(mr,img,result) 'grayscale projection of the measurement area

7.2.2.ZV_MRPOS – Detect Point

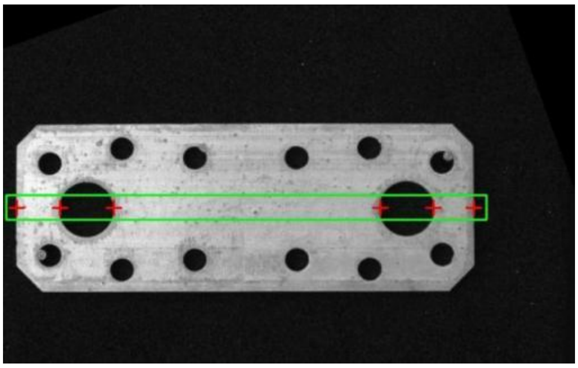
Type	Single area measurement
Description	Use a rectangular, rotating rectangular or arc measurer to measure a point, that is, a point that meets the threshold, the polarity and the position in the grayscale projection, that is, detect a point that meets the conditions in the measurement

	area, and the circumscribed moment of the measurement area cannot exceed the boundary of the measured image.
Grammar	ZV_MRPOS(mr,img matPts,filterSize,thresh,polar,select) mr: ZVOBJECT type, single area measurer img: ZVOBJECT type, measured target image, single-channel image matPts: ZVOBJECT type, matrix type, detected point, n rows and 3 columns, followed by x coordinate, y coordinate and threshold of the point filterSize: filter size, range [1, 201], odd value, common values are 3, 5, 7. If the even number is taken, it will be automatically converted to the nearest odd number internally. thresh: threshold, range [0,255], if it is 0, the default value is 100 polar: edge polarity: 0-white to black, 1-black to white, 2-all selec: edge position: 0-first point, 1-last point, 2-strongest point, 3-all points
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT mr,img,clrImg,ptsMat ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_MRGENRECT(mr,4,255,644,34) 'generate the rectangular measurer ZV_MRPOS(mr,img,ptsMat,3,80,2,3) 'measure the target image and detect the points set by relevant parameters ZV_MATINFO(pts_mat, 0)

	ZV_GRAYTORGB(img,clrImg) 'convert grayscale image to RGB image ZV_RECT(clrImg, 4, 255, 644, 34, ZV_COLOR(0,255,0)) FOR i = 0 TO TABLE(0)-1 ZV_MATGETROW(ptsMat, i, 3, 10) ZV_MARKER(clrImg,TABLE(10),TABLE(11),0,20,ZV_COLOR(255,0,0)) 'draw the cross NEXT

7.2.3.ZV_MRPAIRS – Detect Point-Pair

Type	Judge
Description	Use a rectangular, rotating rectangular or arc measurer to measure the point pair or distance, that is, a point that meets the threshold, the polarity and the position in the grayscale projection, that is, detect a point-pair that meets the conditions in the measurement area, and the circumscribed moment of the measurement area cannot exceed the boundary of the measured image.
Grammar	ZV_MRPAIRS(mr, img, matPts, filterSize, thresh, polar, polar2, select) mr: ZVOBJECT type, the rectangle is the width direction detection, the arc is specified by the parameter, and the single area measurer img: ZVOBJECT type, measured target image, single-channel image matPts: ZVOBJECT type, matrix type, detected point pairs, n rows and 8 columns, followed by point distance, spacing, x1, y1, x2, y2, t1, t2. ➢ point distance -- the distance between point 1 and point 2 ➢ spacing -- the distance between point 1 and the previous point 2 ➢ (x1, y1) -- point 1 coordinates ➢ (x2, y2) -- point 2 coordinates

	➢ t1 -- point 1 threshold, t2--point 2 threshold filterSize: filter size, range [1,201], odd value, commonly used value 3, if the even number is taken, it will be automatically converted to the nearest odd number internally thresh: threshold, range [0,255], if it is 0, the default value is 100 polar1: point 1 edge polarity: 0-white to black, 1-black to white, 2-all polar2: point 2 edge polarity: 0-white to black, 1-black to white, 2-all. The number selected for this parameter is determined based on polar1. The situations are: polar1 = 0, polar2 = 1; polar1 = 1, polar2 = 0; polar1 = 2, polar2 = 2; select: point pair selection: 0-front, 1-last, 2-widest, 3-all
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT mr, img, clrImg, ptsMat ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_MRGENRECT(mr,4,255,644,34) 'generate the rectangular measurer ZV_MRPAIRS(mr,img,ptsMat,3,80,1,2) 'in the image img, use the area measurer, rotated rectangle or arc area measurer mr, according to the parameter settings, generate matrix data mat, in order of point moment, spacing, x1, y1, x2, y2, t1, t2 ZV_MATINFO(pts_mat, 0) ZV_GRAYTORGB(img,clrImg) 'convert grayscale image to RGB image

	ZV_RECT(clrImg, 4, 255, 644, 34, ZV_COLOR(0,255,0)) FOR i = 0 TO TABLE(0)-1 ZV_MATGETROW(ptsMat, i, 8, 10) ZV_MARKER(clrImg,TABLE(12),TABLE(13),0,20,ZV_COLOR(255,0,0)) 'draw the cross ZV_MARKER(clrImg,TABLE(14),TABLE(14),0,20,ZV_COLOR(0,255,0)) 'draw the cross NEXT

7.2.4.ZV_MRPEAK – Detect Peak Point

Type	Single area measurement
Description	Use a rectangular, rotating rectangular or arc measurer to measure the peak point, that is, the leftmost point and the rightmost point of the measurement region, and the circumscribed moment of the measurement area cannot exceed the boundary of the measured image.
Grammar	ZV_MRPEAK(mr,img,tabId,filterSize,thresh,polar,select,scanWidth) mr: ZVOBJECT type, area measurer img: ZVOBJECT type, target image for measurement tabId: TABLE index, the coordinates of the leftmost point and the rightmost point in order, that is, xl, yl, xr, yr filterSize: filter size, range [1,201], take an odd value, the common value is 3, if it is an even number, it will automatically convert to nearest odd number thresh: threshold, range [0,255], if it is 0, the default value is 100 polar: edge polarity: 0-white to black, 1-black to white, 2-all select: edge position: 0-first point, 1-last point, 2-strongest point scanWidth: the scan width of the caliper moment, the common value is 5, if it is > 1 and ≤ 0, 1 is taken.
Controller	It is valid in controllers that support ZV function or they belong

	to 5XX series or above.
Example	ZVOBJECT mr,clrImg,img ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_MRGENRECT(mr,10,142,637,262) 'generate a rectangular measurer ZV_MRPEAK(mr,img,0,3,80,0,0,5) 'measure the target image and detect the points set by related parameters ZV_GRAYTORGB(img,clrImg) 'convert grayscale image to RGB image ZV_MARKER(clrImg,TABLE(0),TABLE(1),0,20,ZV_COLOR(255,0,0)) 'draw a cross ZV_MARKER(clrImg,TABLE(2),TABLE(3),0,20,ZV_COLOR(255,0,0)) 'draw a cross

7.2.5.ZV_MRSIZE –Measurement Area Size Trends

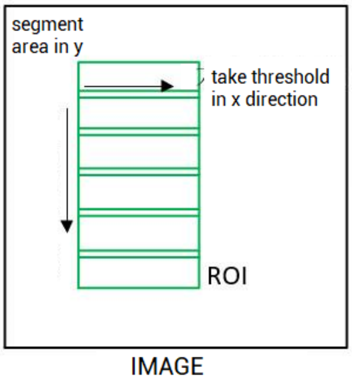
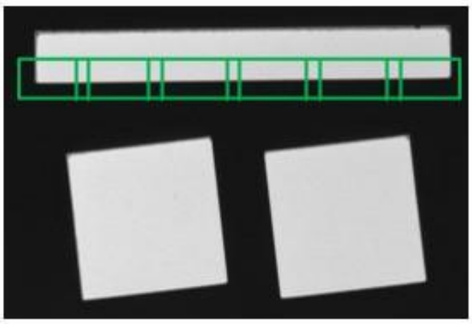
Type	Single area measurement
Description	Use a rectangular, rotating rectangular or arc measurer to measure the minimal and maximum value of point pair size, that is, output the value and the center point position of the corresponding scanning area. The point selection parameters are not set for the measurement, and the widest point pair within the scanning line is used for comparison. Relative threshold mode is used by default.
Grammar	ZV_MRSIZE (mr, img, tabId, filterSize, thresh, polar1, polar2, scanWidth) mr: ZVOBJECT type, area measurer img: ZVOBJECT type, target image for measurement tabId: TABLE index, which is the minimum size, the minimum size corresponding to the x and y coordinates of the center of the scanning area, the maximum size, and the maximum size corresponding to the x and y coordinates of the

	center of the scanning area. filterSize: filter size, range [1,201], take an odd value, the common value is 3, if it is an even number, it will automatically convert to nearest odd number thresh: threshold, range [0,255], if it is 0, the default value is 100 polar1: first edge polarity: 0-white to black, 1-black to white, 2-all polar2: second edge polarity: 0-white to black, 1-black to white, 2-all scanWidth: the scan width of the caliper moment, the common value is 5, if it is > 1 and ≤ 0, 1 is taken.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mr, img ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_MRGENRECT(mr,10,142,637,262) 'generate a rectangular measurer ZV_MRSIZE(mr,img,0,3,80,0,1,5) 'measure size polarity ?table (0), table (3) 'measure minimal value and maximal value in region range

7.3.Segment Region Generation & Measurement

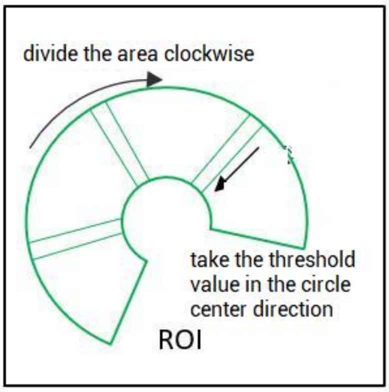
7.3.1.ZV_MRGENLINE – Line Measurement

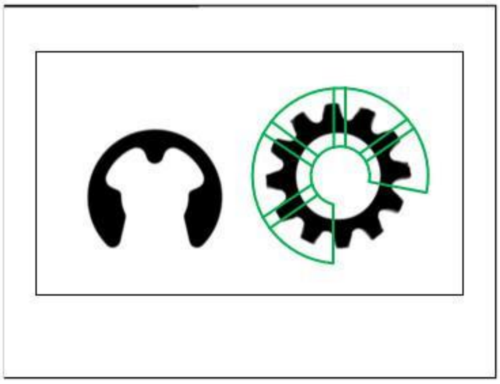
Type	Segment area measurement
Description	Generate a rotated rectangular area for straight-line measurement, slice the area along the y direction, and the subregion takes the threshold along the x direction.

	
Grammar	ZV_MRGENLINE(mr,cx,cy,width,height,angle,interp,subNum,subWidth) mr: ZVOBJECT type, linear measurer cx: the x coordinate of the center of the rotating rectangle, range [0,32766] cy: the y coordinate of the center of the rotating rectangle, range [0,32766] width: width of the rotated rectangle, the unit is pixel, range [1,32766] height: height of rotated rectangle, the unit is pixel, range [1,32766] angle: angle of the rotated rectangle, determined by the image coordinate system, the unit is degree interp: interpolation algorithm, reference rotation subNum: the number of sub-regions, indicating the number of sub-regions that the rotated rectangle is divided into, > 2, otherwise the default value is 8 subWidth: sub-area width, the unit is pixel
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT mr

	ZV_MRGENLINE(mr,,120,100,80,60,90,5,5) 'generate a rotation area for linear measurement

7.3.2.ZV_MRGENCIRCLE – Circle Measurement

Type	Segment area measurement
Description	Generate a circular measurement area, divide the area clockwise along the tangential direction, and calculate the grayscale value of the sub-area along the radial direction toward the center of the circle to select the target point.  IMAGE
Grammar	ZV_MRGENCIRCLE(mr,cx,cy,r,annR[,startAngle=0,extAngle=360, interp=1, subNum=0, subWidth=0]) mr: ZVOBJECT type, circle measurer cx: the center x coordinate of the arc, range [0,32766] cy: the y coordinate of the center of the arc, range [0,32766] r: radius of the center line of the arc, the unit is pixel, range [1,16383] annR: arc half-width, the unit is pixel, range (0,r) startAngle: starting angle of the measurement area, determined by the image coordinate system, that is, clockwise is positive (image coordinate system), the unit is degree extAngle: measurement area angle range, range (0, 360], if it is > 360, it will be automatically converted to 360 internally interp: interpolation algorithm, reference rotation subNum: the number of sub-regions, which represents the

	number of sub-regions (arc is divided), > 3, otherwise the default value is 8 subWidth: sub-area width, the unit is pixel
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT mr ZV_MRGENCIRCLE(mr,120,120,60,40,0,120,1,0,0) 'generate the circle measurer

7.3.3.ZV_MRSETADV – Advanced Parameters Setting of Segment Measurement Region

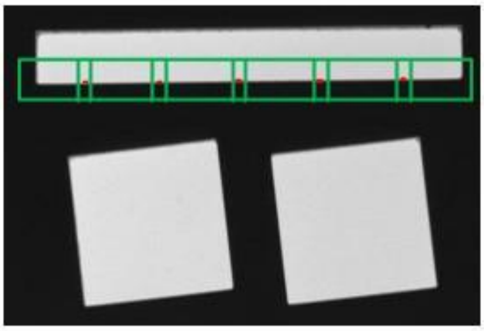
Type	Segment area measurement
Description	Set advanced parameters of segment measurement region.
Grammar	ZV_MRSETADV(mr,filterSize,thresh,polar,select) mr: ZVOBJECT type, measurer filterSize: filter size, range [1,201], default value is 5 thresh: threshold, range [0,255], if it is 0, the default value is 100 polar: edge polarity: 0-white to black, 1-black to white, 2-all, the default value is 2 select: edge position, edge position, based on the scanning direction, 0-first point, 1-last point, 2-strongest point, 3-all

	points
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mr ZV_MRSETADV(mr,5,20,0,1) 'set advanced parameters of segment measurement region.

7.3.4.ZV_MRGETADV – Advanced Parameters Reading of Segment Measurement Region

Type	Segment area measurement
Description	Get advanced parameters of segment measurement region.
Grammar	ZV_MRGETADV(mr,tabId) mr: ZVOBJECT type, measurer tabId: TABLE index, output parameters, obtained measurer parameters, in order filterSize, thresh, polar, select, that is, filter size, threshold, edge polarity, edge position
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mr ZV_MRSETADV(mr,5,20,0,1) 'set advanced parameters for subdivided measurement area ZV_MRGETADV(mr,0) 'output the parameters of the measurer to the TABLE (0) index, and stores the filter size, threshold, edge polarity, and edge position in sequence

7.3.5.ZV_MREDGE – Measure Point of Segment Area

Type	Segment area measurement
Description	Use straight line or circle measurement area to obtain measurement result points 
Grammar	ZV_MREDGE(mr,img,pts) mr: ZVOBJECT type, linear or circular measurement area img: ZVOBJECT type, measured target image, single-channel image pts: ZVOBJECT type, measured result points, n*2 matrix, one point per row, that is, x, y
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mr, img, pts ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_MRGENLINE(mr,48,140,25,259,0,0,10,5) 'generate a rotating area measured by a line ZV_MREDGE(mr,img,pts) 'segment area measurement, use a line or circle measurement area mr in the target image img to measure points and obtain result points and store them in pts

7.3.6.ZV_MRLINE – Line

Type	Segment area measurement
Description	Use the straight-line measurement area to measure straight

	lines, divide the sub-areas along the height, and measure a point in each sub-area, and the scan direction is from left to right.
Grammar	ZV_MRLINE(mr,img,matPts,tabId) mr: ZVOBJECT type, linear measurement area img: ZVOBJECT type, measured target image, single-channel image matPts: ZVOBJECT type, measured result points, n*2 matrix, one point per row tabId: TABLE index, in order x1, y1, x2, y2, that is, the coordinates of the end points of the line
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mr,img,clrImg, pts ZV_READIMAGE(img, "test.png",0) 'read the image in the original image format ZV_MRGENLINE(mr,219,207,80,188,0,0,20,2) 'generate the rotation area of the linear measurement ZV_MRSETADV(mr,3,120,0,1) ZV_MRLINE(mr,img,pts,0) 'store the end point of the target line measured by the rectangular measurement area into the TABLE whose starting index is 0 ZV_GRAYTORGB(img,clrImg) 'convert grayscale image to RGB image ZV_LINE(clrImg, TABLE(0), TABLE(1), TABLE(2), TABLE(3), ZV_COLOR(0,255, 0))

7.3.7.ZV_MRCIRCLE – Circle

Type	Segment area measurement
Description	Use the arc measurement area to measure circle, divide the sub-areas along starting angle to end angle, and measure a point in each sub-area, and the scan direction is from outside to inside.
Grammar	ZV_MRCIRCLE(mr,img,matPts,tabId[,inmr=1])

	mr: ZVOBJECT type, arc measurement area img: ZVOBJECT type, measured target image, single-channel image matPts: ZVOBJECT type, measured result points, n*2 matrix, one point per row tabId: TABLE index, output parameters, in order cx, cy, radius, that is, circle center coordinates and radius inmr: input parameter, indicating whether the measured circle should be included in the arc roi. 0- the circle does not need to be included in the arc roi. 1- the circle is included in the arc roi, but circles beyond the roi will not be detected.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mr,img,clrImg,pts ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_MRGENCIRCLE(mr,302,79,120,15,0,360,1,30,5) ZV_MRCIRCLE(mr,img,pts,0,1) 'store the target circle coordinates and radius measured by the arc measurement area into the TABLE (0) ZV_GRAYTORGB(img,clrImg) 'convert grayscale image to RGB image ZV_CIRCLE(clrImg,TABLE(0),TABLE(1),TABLE(2),ZV_COLOR(0,25,0))

7.4. Measurer ROI

7.4.1. ZV_MRGETROI – Get Measurer ROI & Segment Parameters

Type	Measurer ROI and segment area measurement
Description	Get measurer ROI parameters and segment parameters.
Grammar	ZV_MRGETROI(mr,tabId) mr: ZVOBJECT type, measurer tabId: TABLE starting index, followed by 6 ROI parameters (may not be fully occupied), they are interpolation type (rectangular measurement area is invalid), sub-area number (non-subdivision measurer is invalid), sub-region width (non-subdivision measurer is invalid)
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mr ZV_MRGENLINE(mr,219,207,80,188,0,0,20,2) 'generate a rotation area for linear measurement ZV_MRSETADV(mr,3,120,0,1) ZV_MRGETROI(mr,0) 'get the parameters of the measurer mr into TABLE (0)

7.5. Transformation

7.5.1. ZV_MRCORRECT – Measurement Area Correction

Type	Transformation
Description	According to matrix mat, correct measurement area mar and output to corrMr.
Grammar	ZV_MRCORRECT(mr,mat,corrMr)

	mr: ZVOBJECT type, input measurement area mat: ZVOBJECT type, corrected transformation matrix, 2*2 matrix or 3*3 matrix corrMr: ZVOBJECT type, transformed measurement area
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Related Instruction	Refer to “measurement position correction”.

Chapter VIII Region

8.1.Region Generation

8.1.1.ZV_REGENLINE – Line

Type	Generation
Description	Generate a straight-line area, and the generated area will be automatically cropped to the range where x belongs to [0,32766] and y belongs to [0,32766].
Grammar	ZV_REGENLINE(re,stx,sty,endx,endy) re: ZVOBJECT type, generated area stx: the x coordinate of the starting point of the line, the range is [0,32766], it will be clipped to this range if it exceeds the range sty: the y coordinate of the starting point of the line, the range is [0,32766], it will be clipped to this range if it exceeds the range [0,32766] endx: the x coordinate of the end point of the line, range [0,32766], it will be clipped to this range if it exceeds the range. endy: the y coordinate of the end point of the line, range [0,32766], it will be clipped to this range if it exceeds the range.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT re ZV_REGENLINE(re,50,50,200,200) 'the straight line area with starting point (50,50) and end point (200,200) is put into the area variable "re"

8.1.2.ZV_REGENRECT – Rectangle

Type	Generation
Description	Generate a rectangle area that is parallel to horizontal axis, and

	the generated area will be automatically cropped to the range where x belongs to [0,32766] and y belongs to [0,32766].
Grammar	ZV_REGENRECT(re, x, y, width, height) re: ZVOBJECT type, generated area x: x coordinate of the upper left corner of the rectangle, range (--,32766) y: y coordinate of the upper left corner of the rectangle, range (--,32766) width: width of the rectangle, range [1,32766] height: height of the rectangle, range [1,32766]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT re ZV_REGENRECT(re,0,0,100,100) 'generate a rectangular area with the coordinates of the upper left corner of the rectangle at (0,0) and a width and height of 100 into the area variable re

8.1.3.ZV_REGENRECT2 – Rectangle with Angle

Type	Generation
Description	Generate a rectangle area that is with the angle, and the generated area will be automatically cropped to the range where x belongs to [0,32766] and y belongs to [0,32766].
Grammar	ZV_REGENRECT2(re, cx, cy, width, height, angle) re: ZVOBJECT type, generated area cx: x coordinate of rectangle center cy: y coordinate of rectangle center width: width of the rectangle, range [1,32766] height: height of the rectangle, range [1,32766] angle: rectangle angle, image coordinate system, clockwise is positive, and the unit is degree
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT re ZV_REGENRECT2(re,320,240,120,80,30) 'generate a rectangular area with center point coordinates (320,240), width 120, height 80, angle 30 degrees to the area variable re

8.1.4.ZV_REGENCIRCLE – Circle

Type	Generation
Description	Generate a circle area, and the generated area will be automatically cropped to the range where x belongs to [0,32766] and y belongs to [0,32766].
Grammar	ZV_REGENCIRCLE(re, cx, cy, radius) re: ZVOBJECT type, generated area cx: x coordinate of rectangle center cy: y coordinate of rectangle center radius: circle's radius, range (0, 16383]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_REGENCIRCLE(re,320,240,30) 'generate an area with the circular center coordinates (320,240) and a radius of 30 into the area variable re

8.1.5.ZV_REGENANNULAR – Annular

Type	Generation
Description	Generate an annular area, and the generated area will be automatically cropped to the range where x belongs to [0,32766] and y belongs to [0,32766].
Grammar	ZV_REGENANNULAR(re, cx, cy, radius1, radius2) re: ZVOBJECT type, generated area cx: x coordinate of rectangle center cy: y coordinate of rectangle center

	radius1: inner radius, range (0, 16383] radius2: outer radius, range (radius1, 16383]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT re ZV_REGENANNULAR(re,320,240,30,60) ‘generate an area whose center is (320, 240) coordinates, inner radius is 30 and outer radius is 60 into “re” area variable

8.1.6.ZV_REGENSECTOR – Sector

Type	Generation
Description	Generate a sector area, and the generated area will be automatically cropped to the range where x belongs to [0,32766] and y belongs to [0,32766].
Grammar	ZV_REGENSECTOR(re, cx, cy, radius1, radius2, stAngle, extAngle) re: ZVOBJECT type, generated area cx: x coordinate of rectangle center cy: y coordinate of rectangle center radius1: inner radius radius2: outer radius stAngle: sector starting angle, image coordinate system, clockwise is positive, the unit is degree extAngle: sector angle range, the unit is degree
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT re ZV_REGENSECTOR(re,320,240,30,60,0,120) 'generate a region with the center point coordinates (320,240), an inner circle radius of 30, an outer circle radius of 60, a starting angle of 0, and an angle range of 120 to the area variable re

8.1.7.ZV_REGENPOLYGON – Polygon

Type	Generation
Description	Generate a polygon area, for the polygon area that is composed by point group, it needs 3 points at least, and the generated area will be automatically cropped to the range where x belongs to [0,32766] and y belongs to [0,32766].
Grammar	ZV_REGENPOLYGON(pts,re) pts: ZVOBJECT type, nx2 matrix type, polygon point set re: ZVOBJECT type, generated area
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT pts,re TABLE(0,100,50,200,50,230,100,150,200) ZV_MATGENDATA(pts,4,2,0) 'construct polygon point set ZV_REGENPOLYGON(pts,re) 'generate polygon

8.1.8.ZV_REGENFULLIMG – Full Area

Type	Generation
Description	Generate an area that covers full image.
Grammar	ZV_REGENFULLIMG(pts,re) pts: ZVOBJECT type, input image re: ZVOBJECT type, generated area that covers full image, output parameter
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img,re ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_REGENFULLIMG(img,re) 'generate the area covering the whole image into the re area variable, which is equivalent to ZV_REGENRECT(re,0,0,w,h), and w, h are the width and height of the input image img

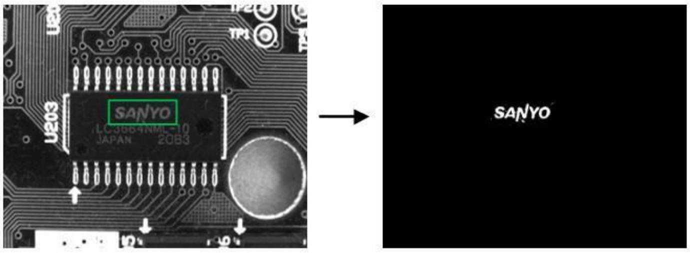
8.2.Region Binarization

8.2.1.ZV_RETHRESH – Region Binarization

Type	Generation
Description	Generate an area that covers full image.
Grammar	ZV_REGENFULLIMG(pts,re) pts: ZVOBJECT type, input image re: ZVOBJECT type, generated area that covers full image, output parameter
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img,re ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_REGENFULLIMG(img,re) 'generate the area covering the whole image into the re area variable, which is equivalent to ZV_REGENRECT(re,0,0,w,h), and w, h are the width and height of the input image img

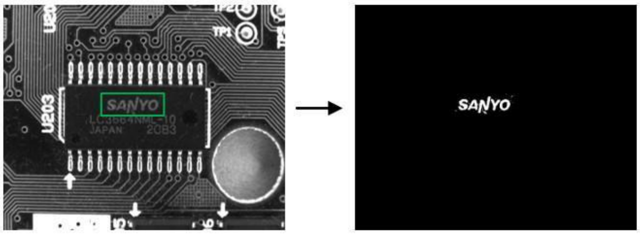
8.2.2.ZV_RETHRESH – Region Binarization

Type	Generation
Description	Image binarization generation area, that is, the image in the mask area specified by “mask” is binarized, and the position where the pixel value is between the thresholds thresh0 and thresh1 is determined as the generation area
Grammar	ZV_RETHRESH(img,mask,re,thresh0,thresh1) img: ZVOBJECT type, input image mask: ZVOBJECT type, mask area re: ZVOBJECT type, area obtained by binarization, output parameter thresh0: low threshold, range [0,255] thresh1: high threshold, range [0,255], thresh1 is greater

	than or equal to thresh0
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img, mask, re, dst ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_IMGINFO(img,0) 'get the basic information of the image ZV_REGENRECT(mask,219,194,151,45) 'generate a rectangular area, generate a mask area ZV_RETHRESH(img,mask,re,130,255) 'binarize the image in the area specified by mask ZV_RETOIMG(re,dst,TABLE(0), TABLE (1)) 'convert region to binarized image

8.2.3.ZV_REAUTOTHRESH – Auto-Binarization

Type	Generation
Description	The image is automatically binarized to generate an area, and the image within the mask area specified by mask is automatically binarized.
Grammar	ZV_REAUTOTHRESH(img,mask,re,tabId) img: ZVOBJECT type, input image mask: ZVOBJECT type, mask area re: ZVOBJECT type, area obtained by automatic binarization, output parameter tabId: TABLE index, output parameter, threshold used for automatic binarization
Controller	It is valid in controllers that support ZV function or they belong

	to 5XX series or above.
Example	 ZVOBJECT img, mask, re, dst ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_IMGINFO(img,0) 'get the basic information of the image ZV_REGENRECT(mask,219,194,151,45) 'generate a rectangular area, generate a mask area ZV_REAUTOTHRESH(img,mask,re,10) 'automatically binarize the image in the area specified by mask, thresh = TABLE(0), thresh is the threshold used for binarization ? TABLE (10) ZV_RETOIMG(re,dst,TABLE(0), TABLE (1)) 'convert region to binarized image

8.2.4.ZV_RETOIMG – Convert Region to Binarization

Type	Conversion
Description	Convert the area to a binary image, and the maximum data size of the generated image cannot exceed 2G, that is, width * width <= 2048*1024*1024
Grammar	ZV_RETOIMG(re,img,width,height) re: ZVOBJECT type, area to be converted img: ZVOBJECT type, converted image width: converted image width, [1,32766] height: converted image height, [1,32766]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT img, mask, re, dst ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_IMGINFO(img,0) 'get the basic information of the image ZV_REGENRECT(mask,227,152,521,527) 'generate rectangular area, generate mask area ZV_REAUTOTHRESH(img,mask,re,10) 'area automatic binarization ZV_RETOIMG(re,dst,TABLE(0), TABLE (1)) 'area to binary image, convert the locale set by re to a binary image with the same size as the img image

8.3.Region Clip

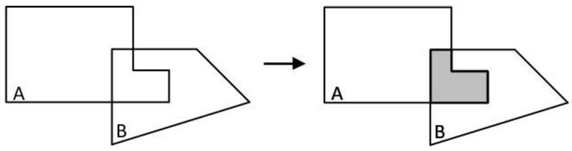
8.3.1.ZV_RECLIP – Clip Region

Type	Conversion
Description	Cropping the input area into a rectangle described by the control parameters, which is equivalent to the intersection of the input area and the rectangular area described by the control parameters, but it is more efficient than using the control parameters to call ZV_REGENRECT to generate a rectangular area and then find out the intersect with the input area obj_region.
Grammar	ZV_RECLIP(re,reCliped,x1,y1,x2,y2) re: ZVOBJECT type, input area reCliped: ZVOBJECT type, clipped area x1: upper left corner of rectangle x coordinate y1: upper left corner of rectangle y coordinate x2: lower right corner of rectangle x coordinate y2: lower right corner of rectangle y coordinate
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

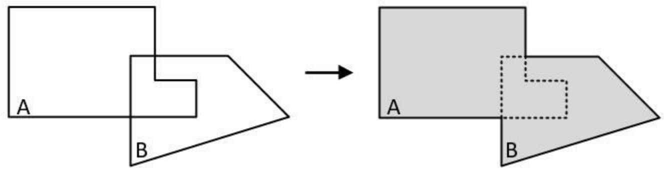
Example	ZVOBJECT img, re, reCliped ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_RECLIP(re,reCliped,0,0,640,480) 'clip the re area to the rectangular area formed by the upper left corner (0,0) coordinates and the lower right corner (640,480) coordinates, and then store it in the variable reCliped

8.4.Region Operation

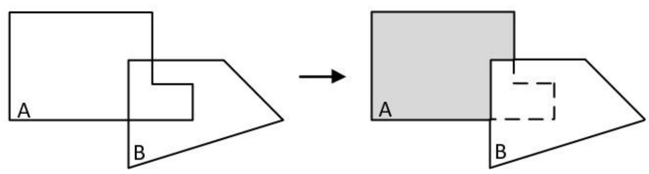
8.4.1.ZV_REITSEC – Intersection

Type	Region operation
Description	Calculate the intersection between re1 and re2. 
Grammar	ZV_REITSEC(re1,re2,re) re1: ZVOBJECT type, region 1 re2: ZVOBJECT type, region 2 re: ZVOBJECT type, calculated intersection of area 1 and area 2
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT re1, re2, re ZV_REITSEC(re1,re2,re) 'calculate the intersection of two areas to the re area

8.4.2.ZV_REUNION – Union

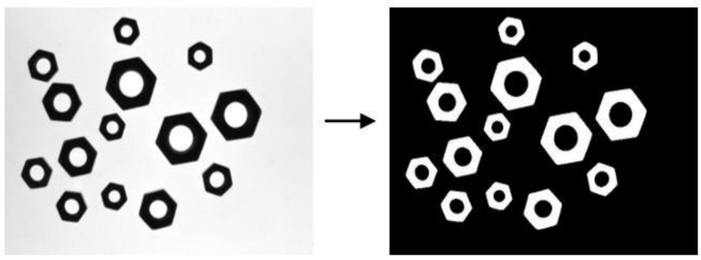
Type	Region operation
Description	Calculate the union between re1 and re2. 
Grammar	ZV_REUNION(re1,re2,re) re1: ZVOBJECT type, region 1 re2: ZVOBJECT type, region 2 re: ZVOBJECT type, calculated union of area 1 and area 2
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT re1, re2, re ZV_REUNION(re1,re2,re) 'calculate the union of two areas to the re area

8.4.3.ZV_REDIFF – Difference Set

Type	Region operation
Description	Calculate the difference set between re1 and re2, that is, re1 minus re2. 
Grammar	ZV_REDIFF(re1,re2,re) re1: ZVOBJECT type, region 1 re2: ZVOBJECT type, region 2 re: ZVOBJECT type, calculated difference set of area 1 and area 2
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT re1, re2, re ZV_REDIFF(re1,re2,re) 'calculate the difference set of two areas to the re area

8.4.4.ZV_RECONNECT – Connection Area

Type	Region operation
Description	Calculate the connected area of the area, decompose the input area into multiple connected areas, one area may be composed of multiple disconnected connected areas, so multiple disconnected connected areas can be obtained by decomposing an area, connected areas --- each trip in the area is connected
Grammar	ZV_RECONNECT(re,reConnect) re: ZVOBJECT type, input area reConnect: ZVOBJECT type, list, output parameters, the area stored in the list is the ZVOBJECT type
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 DIM reCnt ZVOBJECT img, dst, mask, re, reBIn, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_REGENFULLIMG(img,mask) 'generated area that covers the entire image ZV_RETHRESH(img,mask,reBIn,0,150) 'region binarization ZV_RECONNECT(reBIn,reConnect) 'calculate the connected area of the area ZV_IMGCOPY(img,dst) 'copy image

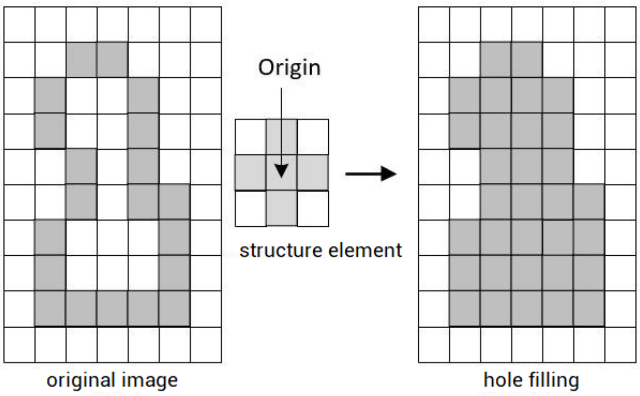
	ZV_IMGSETCONST(dst,0) 'set the pixel value of the image to 0 reCnt = ZV_LISTCOUNT(reConnect) 'get the number of connected areas FOR i = 0 TO reCnt-1 ZV_LISTGET(reConnect,re,i) ZV_REGION(dst,re,0,255) NEXT

8.4.5.ZV_REUNIONLIST – Merge

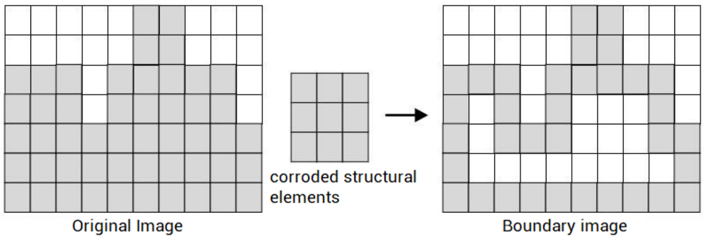
Type	Region operation
Description	Merge all the regions in the list into one region, that is, calculate the union of all the regions in the list
Grammar	ZV_REUNIONLIST(list,reUnion) list: ZVOBJECT type, list reUnion: ZVOBJECT type, merge area, output parameters
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect, reUnion ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_REGENFULLIMG(img,mask) 'generate the area that covers the whole image ZV_RETHRESH(img,mask,reBin,0,150) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_REUNIONLIST(reConnect,reUnion) 'merge the regions in the list into one region

8.4.6.ZV_REFILLUP – Hole Filling

Type	Region operation
Description	Fill the area with holes and output the area after the holes are

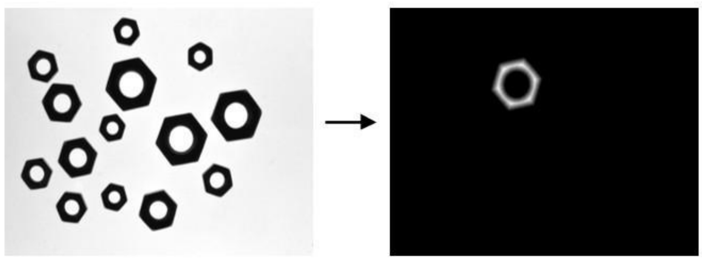
	filled. 
Grammar	ZV_REFILLUP(re,reFill) re: ZVOBJECT type, region reFill: ZVOBJECT type, filled region, output parameter
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img, clrImg, mask, re, reBin, reConnect, refill ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_REGENFULLIMG(img,mask) 'generate the area that covers the whole image ZV_RETHRESH(img,mask,reBin,0,150) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get element number 3 in the list ZV_REFILLUP(re,reFill) 'fill holes in the area ZV_GRAYTORGB(img,clrImg) 'convert grayscale image to RGB image ZV_REGION(clrImg,reFill,0,ZV_COLOR(0,255,0))

8.4.7.ZV_REBOUNDARY – Boundary

Type	Region operation
Description	Calculate the boundary of the area, which is the outline of the area, the outline at the pixel level 
Grammar	ZV_REBOUNDARY(re,reBoundary,type) re: ZVOBJECT type, area reBoundary: ZVOBJECT type, boundary, output parameters, boundary is also represented by area type: border type, 0-outer border, the contour line is located 1 pixel beyond the edge of the area, 1-inner border, the contour line is located at the edge of the area, 2-inner border does not contain holes, the contour line is located in the area edge
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img, dst, mask, re, reBin, reConnect, reBoundary ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_IMGINFO(img,0) 'get the basic information of the image ZV_REGENFULLIMG(img,mask) 'generate an area that covers the entire image ZV_RETHRESH(img,mask,reBin,0,150) 'region binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element with number 3 in the list

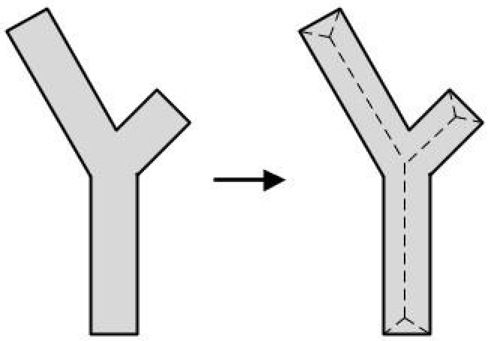
	ZV_REBOUNDARY(re,reBoundary,0) 'calculate the outer boundary of the area ZV_RETOIMG(reVoundary,dst,TABLE(0), TABLE (1)) 'region to binary image

8.4.8.ZV_REDISTTRANS – Region Distance Image

<table>
<tr><td>Type</td><td>Region operation</td></tr>
<tr><td>Description</td><td>Calculate the distance from each point in the area to the area boundary and generate the corresponding distance map.</td></tr>
<tr><td>Grammar</td><td>ZV_REDISTTRANS(re,img,width,height,type)
re: ZVOBJECT type, region
img: ZVOBJECT type, generated single-channel distance map
width: width of the generated image, range [1,32766]
height: height of the generated image, range [1,32766]
type: distance type, as follows:
<table>
<tr><td>Type</td><td>Distance method</td></tr>
<tr><td>0</td><td>max(|x1-x2|,|y1-y2|)</td></tr>
<tr><td>1</td><td>|x1-x2|+|y1-y2|</td></tr>
<tr><td>2</td><td>$\sqrt{(x1-x2^2+(y1-y2)^2}$</td></tr>
</table></td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><td>Example</td><td>
ZVOBJECT img, dst, mask, re, reBin, reConnect
ZV_READIMAGE(img, "test.png", 0)
'read the image in the original image format
ZV_REGENFULLIMG(img,mask)
'generate the area that covers the whole image</td></tr>
</table>

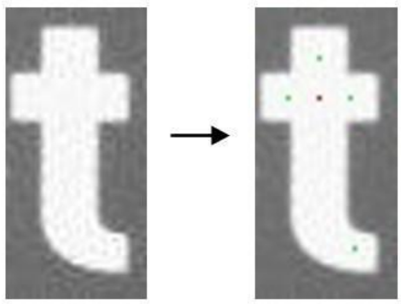
	ZV_RETHRESH(img,mask,reBin,0,150) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element number 3 in the list ZV_REDISTTRANS(re,dst,640,480,2) 'generate the distance map of the area

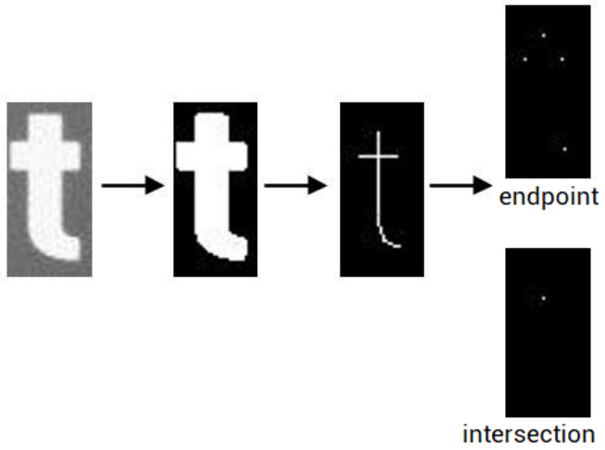
8.4.9.ZV_RESKELETON – Skeletonization

Type	Region operation
Description	Skeletonize regions to generate regions of individual pixels. 
Grammar	ZV_RESKELETON(re,skeRe) re: ZVOBJECT type, area skeRe: ZVOBJECT type, skeleton area
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img, dst, mask, re, reBin, reConnect, skeRe ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_IMGINFO(img,0) 'get the basic information of the image ZV_REGENFULLIMG(img,mask) 'generated area that covers the entire image

	ZV_RETHRESH(img,mask,reBin,0,150) 'area binarization ZV_RECONNECT(reBin,reConnect) ' calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element number 3 in the list ZV_RESKELETON(re,skeRe) 'regional skeletonization ZV_RETOIMG(skeRe,dst,TABLE(0),TABLE(1)) 'convert region to binary image

8.4.10. ZV_RESKELETONJUNCT – Area endpoints and intersections

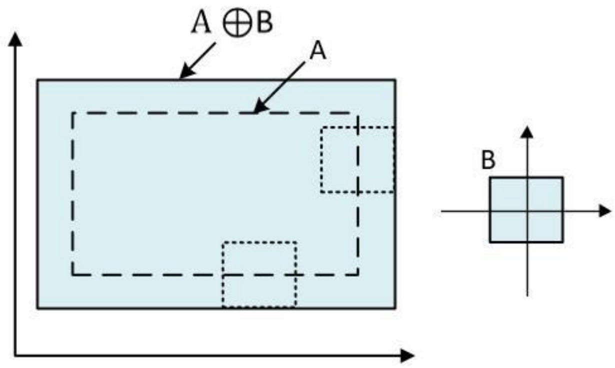
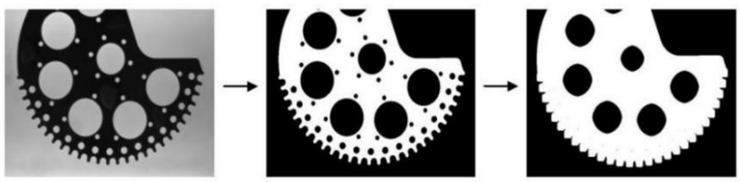
Type	Region operation
Description	Calculate the endpoints and intersections of the region. In order to obtain reliable results, the input region cannot contain lines wider than one pixel, when the skeletonized (zv_re_skeleton) region meets this condition, then the calculated results output the endpoints and intersections in the form of a region.
Grammar	ZV_RESKELETONJUNCT(re,endPtsRe,junPtsRe) re: ZVOBJECT type, area endPtsRe: ZVOBJECT type, area, endpoint junPtsRe: ZVOBJECT type, area, intersection
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	




```
DIM reCnt
ZVOBJECT img, dst1, dst2, mask, re, reBin, reConnect, endPtsRe,
junPtsRe, skeRe
ZV_READIMAGE(img, "test.png", 0)
    'read the picture in the original image format
ZV_REGENFULLIMG(img,mask)
    'generated area that covers the whole image
ZV_RETHRESH(img,mask,reBin,178,255) 'region binarization
ZV_RECONNECT(reBin,reConnect)
    'calculate the connected area of region
ZV_IMGCOPY(img,dst1)
ZV_IMGSETCONST(dst1,0)
ZV_IMGCOPY(img,dst2)
ZV_IMGSETCONST(dst2,0)
reCnt = ZV_LISTCOUNT(reConnect)
FOR i = 0 TO reCnt-1
    ZV_LISTGET(reConnect,re,i)
        'get the element with serial number i in the list
    ZV_RESKELETON(re,skeRe)    'region skeletonization
    ZV_RESKELETONJUNCT(skeRe,endPtsRe,junPtsRe)
    ZV_REGION(dst1,endPtsRe,0,255)
    ZV_REGION(dst2,junPtsRe,0,255)
NEXT
```

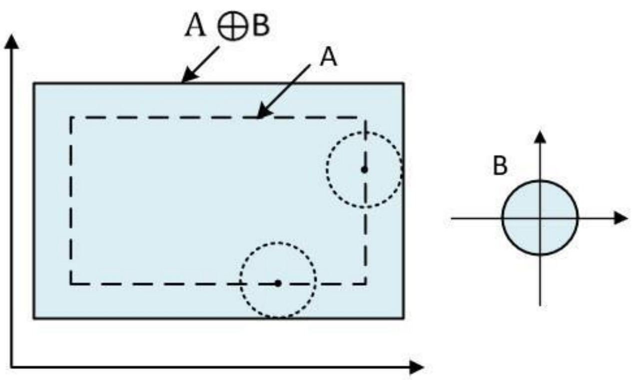
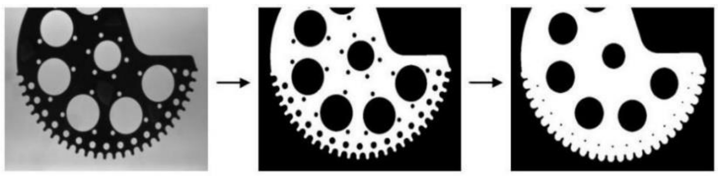
8.5.Morphology

8.5.1.ZV_REDILATE – Rectangle Expansion

Type	Feature
Description	Use rectangular structural elements to expand the area. The expansion will expand the area and fill holes smaller than the structural elements. And the time consumption is proportional to the size of the structural elements. 
Grammar	ZV_REDILATE(re,reDilate,width,height) re: ZVOBJECT type, region reDilate: ZVOBJECT type, expanded area, output parameter width: rectangular structure element width, range [1,511] height: rectangular structure element height, range [1,511]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img, dst, mask, re, reDilate ZV_IMGINFO(img,0) 'get the basic information of the image ZV_REGENFULLIMG(img,mask) 'generated region that covers the entire image ZV_RETHRESH(img,mask,re,0,120) 'area binarization ZV_REDILATE(re,reDilate,3,3) 'dilate the area with a rectangular structure element 3 pixels

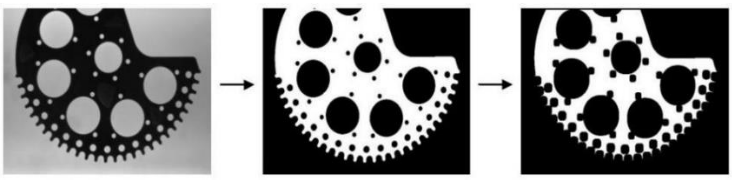
	wide and 3 pixels high ZV_RETOIMG(reDilate,dst,TABLE(0),TABLE (1)) 'region to binary image

8.5.2.ZV_REDILATECIRCLE – Circle Expansion

Type	Feature
Description	Use circular structural elements to expand the area. The expansion will expand the area, smooth the boundary of the area, and fill holes smaller than the structural element. The time consumption is proportional to the size of the structural element. It is recommended that the structural element radius be 0.5, 1.5, 2.5, 3.5, 5.5, etc., mainly to avoid translation of the area, because a circle with an integer radius will have a non-integer center of gravity, and this center of gravity will be rounded to the next integer. 
Grammar	ZV_REDILATECIRCLE(re,reDilate,radius) re: ZVOBJECT type, area reDilate: ZVOBJECT type, expanded area, output parameter radius: circular structure element radius, range [0.5, 255]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	

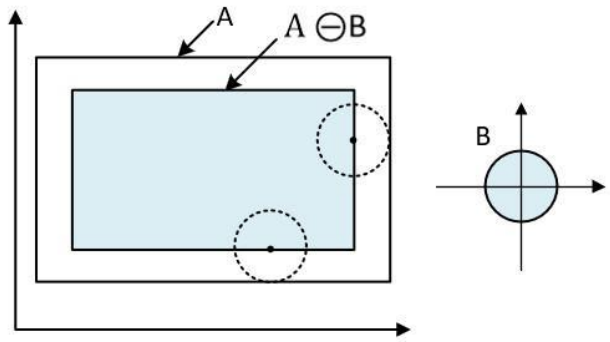
	ZVOBJECT img, dst, mask, re, reDilate ZV_IMGINFO(img,0) 'get the basic information of the image ZV_REGENFULLIMG(img,mask) 'generated region that covers the entire image ZV_RETHRESH(img,mask,re,0,120) 'area binarization ZV_REDILATECIRCLE(re,reDilate,1.5) 'dilate the region with a circular structure element with a radius of 1.5 pixels ZV_RETOIMG(reDilate,dst,TABLE(0),TABLE (1)) 'region to binary image

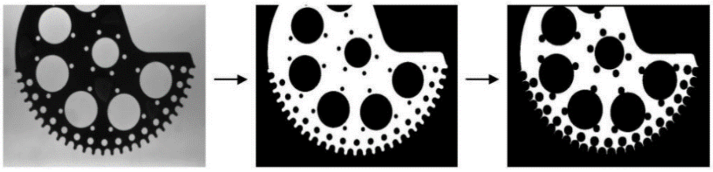
8.5.3.ZV_REERODE – Rectangle Erosion

Type	Feature
Description	Use rectangular structural elements to corrode the area. Erosion will shrink the area and remove areas smaller than the structural elements. The time-consuming is proportional to the size of the structural elements.
Grammar	ZV_REERODE(re,reErode,width,height) re: ZVOBJECT type, area reErode: ZVOBJECT type, corroded area, output parameter width: rectangular structure element width, range [1,511] height: rectangular structure element height, range [1,511]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img, dst, mask, re, reErode ZV_IMGINFO(img,0) 'get the basic information of the image ZV_REGENFULLIMG(img,mask) 'generated area that covers the entire image

	ZV_RETHRESH(img,mask,re,0,120) 'regional binarization ZV_REERODE(re,reErode,3,3) 'use a rectangular structure element with a width of 3 pixels and a height of 3 pixels to erode the area ZV_RETOIMG(reErode,dst,TABLE(0),TABLE (1)) 'region to binary image

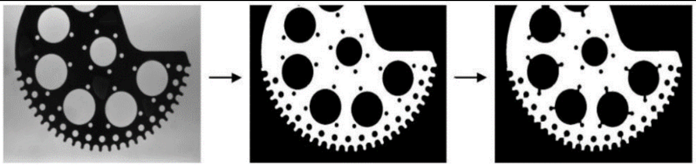
8.5.4.ZV_REERODECIRCLE – Circle Erosion

Type	Feature
Description	Use circular structural elements to corrode the area. Corrosion will shrink the area, smooth the boundaries of the area, and remove areas smaller than the structural element. The time consumption is proportional to the size of the structural element. It is recommended that the structural element radius be 0.5, 1.5, 2.5, 3.5, 5.5, etc., mainly to avoid translation of the area, because a circle with an integer radius will have a non-integer center of gravity, and this center of gravity will be rounded to the next integer. 
Grammar	ZV_REERODECIRCLE(re,reErode,radius) re: ZVOBJECT type, region reErode: ZVOBJECT type, area after corrosion, output parameter radius: radius of circular structure element, range [0.5, 255]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	 ZVOBJECT img, dst, mask, re, reErode ZV_IMGINFO(img,0) 'get the basic information of the image ZV_REGENFULLIMG(img,mask) 'generated area that covers the entire image ZV_RETHRESH(img,mask,re,0,120) 'regional binarization ZV_REERODECIRCLE(re,reErode,1.5) 'use a circular structure element with a radius of 1.5 pixels to erode the area ZV_RETOIMG(reErode,dst,TABLE(0),TABLE (1)) 'region to binary image

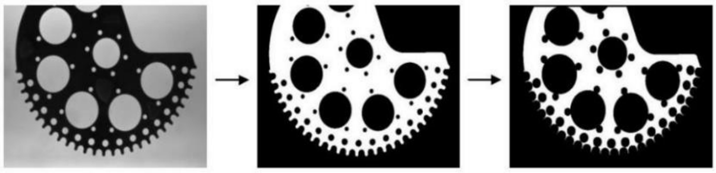
8.5.5.ZV_REOPENING – Rectangle Opening Operation

Type	Feature
Description	Use rectangular structural elements to do opening operation for area, that is, first corrode and then expand. The opening operation will not change the original shape characteristics of the area, but it will remove the isolated area smaller than the structural element or disconnect the connecting line smaller than the structural element. And the time consumption is proportional to the size of the structural element.
Grammar	ZV_REOPENING(re,reOpen,width,height) re: ZVOBJECT type, area reOpen: ZVOBJECT type, area after opening operation, output parameter width: rectangular structure element width, range [1,511] height: rectangular structure element height, range [1,511]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	 ZVOBJECT img, dst, mask, re, reOpen ZV_IMGINFO(img,0) 'get the basic information of the image ZV_REGENFULLIMG(img,mask) 'generated area that covers the entire image ZV_RETHRESH(img,mask,re,0,120) 'regional binarization ZV_REOPENING(re,reOpen,3,3) 'use a rectangular structure with a width of 3 pixels and a height of 3 pixels to open the region ZV_RETOIMG(reOpen,dst,TABLE(0),TABLE (1)) 'region to binary image

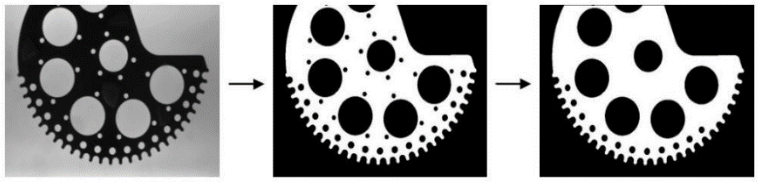
8.5.6.ZV_REOPENCIRCLE – Circle Opening Operation

Type	Feature
Description	Use circular structural elements to do opening operation for area, that is, first corrode and then expand. The opening operation will not change the original shape characteristics of the area, and has the effect of smooth area boundaries, but it will remove isolated areas or disconnections smaller than the structural elements or connecting lines that are smaller than structural elements. Also, the time consumption is proportional to the size of the structural elements. It is recommended the structural element radius are 0.5, 1.5, 2.5, 3.5, 5.5, etc.
Grammar	ZV_REOPENCIRCLE(re,reOpen,radius) re: ZVOBJECT type, region reOpen: ZVOBJECT type, area after opening operation, output parameter radius: radius of circular structure element, range [0.5, 255]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	 ZVOBJECT img, dst, mask, re, reOpen ZV_IMGINFO(img,0) 'get the basic information of the image ZV_REGENFULLIMG(img,mask) 'generated area that covers the entire image ZV_RETHRESH(img,mask,re,0,120) 'regional binarization ZV_REOPENCIRCLE(re,reOpen,1.5) 'use a circular structure element with a radius of 1.5 pixels to open the area. ZV_RETOIMG(reOpen,dst,TABLE(0),TABLE (1)) 'region to binary image

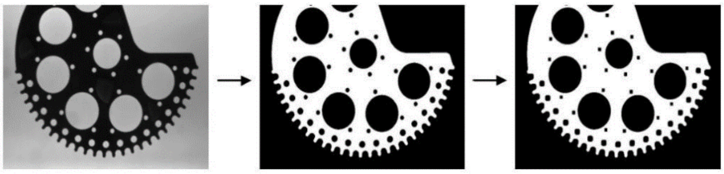
8.5.7.ZV_RECLOSECIRCLE – Circle Closing Operation

Type	Feature
Description	Use circular structural elements to do closing operation for area, that is, first expand and then corrode. The closing operation will not change the original shape characteristics of the area, and has the effect of smooth area boundaries, but it will connect gaps smaller than structural elements or fill holes smaller than structural elements, and the time-consuming is proportional to the size of structural elements.
Grammar	ZV_RECLOSECIRCLE(re,reClose,radius) re: ZVOBJECT type, region reClose: ZVOBJECT type, area after closing operation, output parameter radius: radius of circular structure element, range [0.5, 255]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	 ZVOBJECT img, dst, mask, re, reClose ZV_IMGINFO(img,0) 'get the basic information of the image ZV_REGENFULLIMG(img,mask) 'generated area that covers the entire image ZV_RETHRESH(img,mask,re,0,120) 'regional binarization ZV_REOPENCIRCLE(re,reClose,1.5) 'use a circular structure element with a radius of 1.5 pixels to close the area. ZV_RETOIMG(reClose,dst,TABLE(0),TABLE (1)) 'region to binary image

8.5.8.ZV_RECLOSING – Rectangle Closing Operation

Type	Feature
Description	Use rectangular structural elements to do closing operation for area, that is, first expand and then corrode. The closing operation will not change the original shape characteristics of the area, but it will connect gaps smaller than structural elements or fill holes smaller than structural elements, and the time-consuming is proportional to the size of structural elements.
Grammar	ZV_RECLOSING(re,reClose,width,height) re: ZVOBJECT type, region reClose: ZVOBJECT type, area after closing operation, output parameter width: rectangular structure element width, range [1, 551] height: rectangular structure element height, range [1, 551]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	 ZVOBJECT img, dst, mask, re, reClose ZV_IMGINFO(img,0) 'get the basic information of the image ZV_REGENFULLIMG(img,mask) 'generated area that covers the entire image ZV_RETHRESH(img,mask,re,0,120) 'regional binarization ZV_REOPENCIRCLE(re,reClose,3,3) 'use a rectangular structure element with the width of 3 pixels and the height of 3 pixels to do closing operation ZV_RETOIMG(reClose,dst,TABLE(0),TABLE (1)) 'region to binary image

8.5.9.ZV_REMORPH – Region Morphology

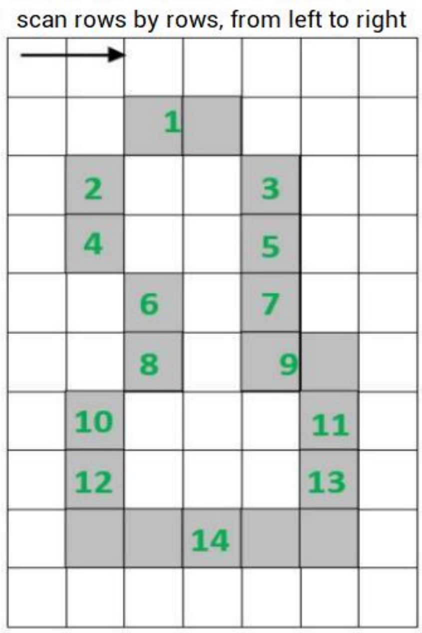
Type	Feature
Description	Use any structure element to perform morphological processing on the region. The structure element uses the region to show, which can be generated by the operator that generates the region, such as ZV_REGENRECT, ZV_REGENRECT2, ZV_REGENCIRCLE, etc. The time consumption is proportional to the size of the structure element and the number of iterations.

<table>
<tr><td>Grammar</td><td>ZV_REMORPH(re,st,reMorph,op,iter)
re: ZVOBJECT type, area
st: ZVOBJECT type, area, if it is empty or the area is invalid, it will take a 3x3 rectangular area
reMorph: ZVOBJECT type, area, output parameters
op: morphological processing type
<table>
<tr><td>0</td><td>Corrosion, shrinking the area</td></tr>
<tr><td>1</td><td>Expand, expand the area</td></tr>
<tr><td>2</td><td>Opening operation to remove isolated areas smaller than the structural elements or connecting lines smaller than the structural elements.</td></tr>
<tr><td>3</td><td>Closing operation to connect gaps smaller than structural elements or fill holes smaller than structural elements</td></tr>
<tr><td>4</td><td>Morphological gradient. The gradient describes the mutation part of the region. The junction from white to black or black to white is the mutation part. That is, the morphological gradient calculates the boundary of the region.</td></tr>
<tr><td>5</td><td>Top hat, which separates isolated regions smaller than structural elements, or separates connecting lines smaller than structural elements</td></tr>
<tr><td>6</td><td>The bottom hat divides gaps that are smaller than the structural element, or holes that are smaller than the structural element.</td></tr>
</table>
iter: the number of iterations, the range [1,20], the common value is 1, which means that the structure element st is used to continuously perform morphological processing on the region, and the time consumption is proportional to the number of times.</td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><td>Example</td><td>ZVOBJECT img, dst, mask, re, st, reMorph
ZV_IMGINFO(img,0)
'get the basic information of the image</td></tr>
</table>

	ZV_REGENFULLIMG(img,mask) 'generated area that covers the whole image ZV_RETHRESH(img,mask,re,0,120) 'region binarization 'ZV_REGENRECT(st,0,0,3,3) 'structural element ZV_REMORPH(re,st,reMorph,0,1) 'use a 3x3 rectangular structure element to erode the region ZV_RETOIMG(reMorph,dst,TABLE(0),TABLE (1)) 'region to binary image

8.6.Feature

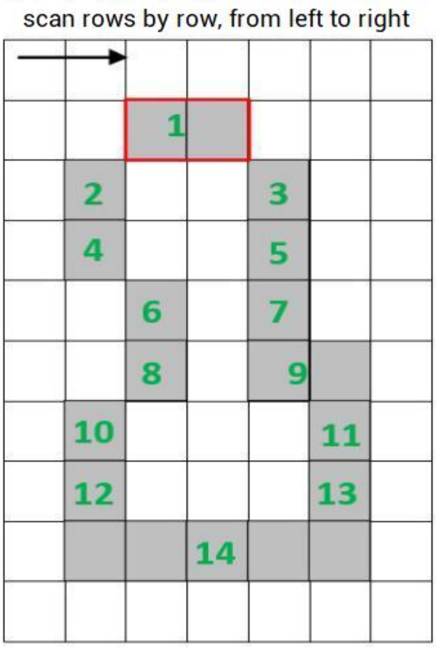
8.6.1.ZV_RERUNSNUM – Travel Numbers

Type	Feature
Description	Get the number of trips in the area. Online command function is supported, using parameters that don't need to pass in TABLE index. 
Grammar	ZV_RERUNSNUM(re,tabId) Or count = ZV_RERUNSNUM(re) re: ZVOBJECT type, region

	tabId: TABLE index, output parameters
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_REGENFULLIMG(img,mask) 'generated area covering the entire image ZV_RETHRESH(img,mask,re,0,150) 'region binarization ZV_RERUNSNUM(re,0) 'get the number of connected domains in the region, and put the number of travels into TABLE(0)

8.6.2.ZV_RERUNS – Get Travel

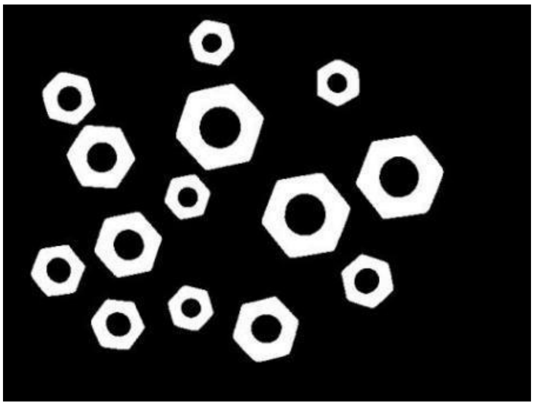
Type	Feature
Description	Obtain the trip of specified No.id in the region.
Grammar	ZV_RERUNS (re, id, tabId) Or count = ZV_RERUNSNUM(re) re: ZVOBJECT type, region id: No. id of specified area, ≥0, < region travel numbers tabId: TABLE index, output parameters, "row" row No. of travel, "cb" starting column and "ce" end column are output in order.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

<table>
<tr><td>Example</td><td>

ZVOBJECT img, mask, re,
ZV_READIMAGE(img, "test.png", 0)
'read the image in the original image format
The area generated by ZV_REGENFULLIMG(img,mask)
‘generated area that covers the whole image
ZV_RETHRESH(img,mask,re,0,150) 'region binarization
ZV_RERUNS(re,1,0)
'get the trip of specified No. 1 in the region, and put the values into the TABLE (0)</td></tr>
</table>

8.6.3.ZV_RECONNECTCNT – The Number of Connected Aeras

Type	Feature
Description	Obtain the number of connected areas in the region.
Grammar	ZV_RECONNECTCNT(re,tabId) re: ZVOBJECT type, area tabId: TABLE index, output parameter
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	 ZVOBJECT img, mask, re, ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format The area generated by ZV_REGENFULLIMG(img,mask) 'generated area that covers the whole image ZV_RETHRESH(img,mask,re,0,150) 'region binarization ZV_RECONNECTCNT(re,0) 'get the number of connected domains in the area and put the number into TABLE(0) ? TABLE(0) 'for the image above, the output value is 14

8.6.4.ZV_REAREA – Area (Square)

Type	Feature
Description	Calculate the square of the area (the number of pixels). Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_REAREA(re,tabId) or area = ZV_REAREA(re) re: ZVOBJECT type, region tabId: TABLE index, output area, output parameters
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect DIM area ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format

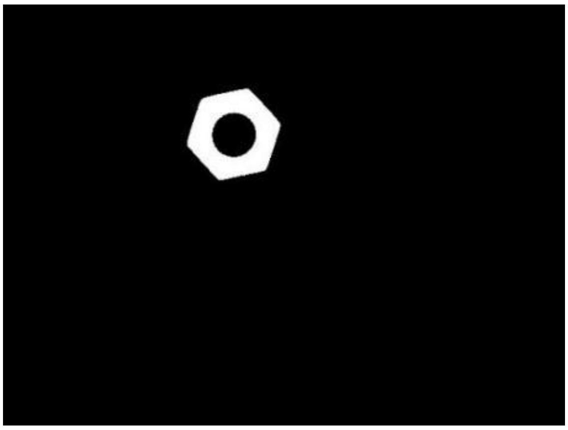
	ZV_REGENFULLIMG(img,mask) 'generated area covering the entire image ZV_RETHRESH(img,mask,reBin,0,150) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element of No.3 in the list area = ZV_REAREA(re) 'calculate the area of the area ? area

8.6.5.ZV_REHOLESCNT – The Number of Holes

Type	Feature
Description	Calculate the number of holes in the region.
Grammar	ZV_REHOLESCNT(re,tabId) re: ZVOBJECT type, region tabId: TABLE index, output parameters
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_REGENFULLIMG(img,mask) ‘generated region that covers the whole image ZV_RETHRESH(img,mask,re,0,100) 'region binarization ZV_REHOLESCNT(re,0) 'calculate the number of holes in the region, put the number in TABLE(0) ? TABLE(0)

8.6.6.ZV_REHOLESAREA – The Aera of Holes

Type	Feature

Description	Calculate the area of holes in the region.
Grammar	ZV_REHOLESAREA(re,tabId) re: ZVOBJECT type, region tabId: TABLE index, output parameters
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_REGENFULLIMG(img,mask) ‘generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) ’region binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reBint,re,3) 'get the element of No.3 in the list ZV_REHOLESAREA(re,0) 'calculate the area of the hole in the region and put the area into TABLE(0)? TABLE(0) 'for the above image, the output value is 1969

8.6.7.ZV_REAREACENTER – Region Area & Position

Type	Feature
Description	Calculate the center of the area.

Grammar	ZV_REAREACENTER(re, tabId) re: ZVOBJECT type, area tabId: TABLE index, output parameters, in order of area, cx, cy, that is, the area of the area and the position of the center.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_REGENFULLIMG(img,mask) 'generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element of No.3 in the list ZV_REAREACENTER(re,0) 'calculate the area and center position of the area, put the position into TABLE, area = TABLE(0), cx = TABLE(1), cy = TABLE(2)

8.6.8.ZV_RECONTLENGTH – Length

Type	Feature
Description	Calculate the perimeter of the region, that is, the contour length of the region. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_RECONTLENGTH(re,tabId) or len = ZV_RECONTLENGTH(re) re: ZVOBJECT type, region tabId: TABLE index, output parameters
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect

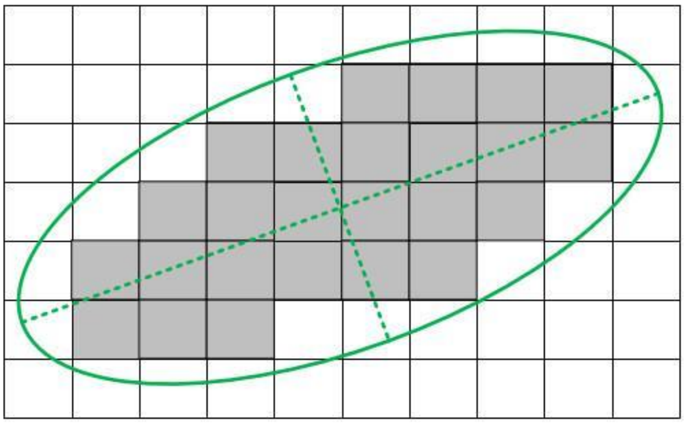
	ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_REGENFULLIMG(img,mask) ‘generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,re_connect) 'calculate the connected area of the area ZV_LISTGET(re_connect, re_src, 3) 'get the element of No.3 in the list ZV_RECONTLENGTH(re_src,0) 'calculate the perimeter of the region, put the perimeter into TABLE(0) ? TABLE (0)

8.6.9.ZV_REORIENT – Angle

Type	Feature
Description	Calculate the angle of the area. This operator is based on the ZV_REELLIPAXIS operator. In addition, it also calculates the point at the maximum distance between the area outline and the center of gravity of the area. If the column coordinate of the point is smaller than the column coordinate of the center of gravity, it will add 180 to the angle calculated by ZV_REELLIPAXIS.
Grammar	ZV_REORIENT(re,tabId) or angle = ZV_REORIENT(re) re: ZVOBJECT type, region tabId: TABLE index, output parameter, angle of the area, clockwise is positive, unit is degrees, range [-180,180)
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_REGENFULLIMG(img,mask)

	'generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'region binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element whose sequence number is 3 in the list ZV_REORIEN(re,0) 'calculate the direction angle of the region, put the angle value into TABLE(0) ? TABLE(0)

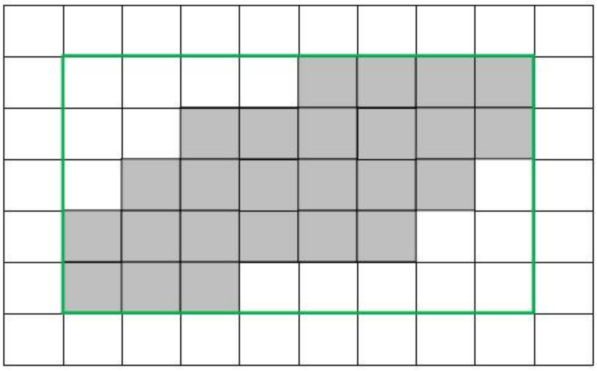
8.6.10. ZV_REELLIPAXIS – Ellipse Axis Parameters

Type	Feature
Description	Calculate the inertia axis (equivalent ellipse) parameters of the area, that is, the axis and direction of the area are the same as the axis and direction of the ellipse, and the direction (angle) is the angle between the major axis (main axis) and the horizontal line. The calculation method is to use the pixel center coordinates to calculate the semi-major axis and semi-minor axis. The returned major axis and minor axis are twice and plus 1 the corresponding semi-axis, that is, semi-axis*2+1. The addition of 1 is because the area of the regional point is not Ideal point, in this way, it can make the ellipse parameters more suitable. 

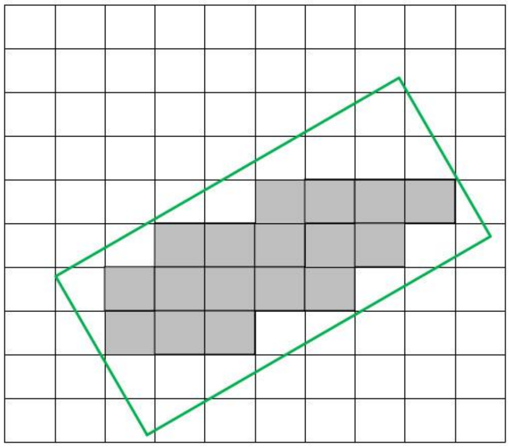
Grammar	ZV_REELLIPAXIS(re,tabId) re: ZVOBJECT type, area tabId: TABLE index, output parameter, the output is the equivalent ellipse major axis, minor axis, angle (unit is degree, clockwise is positive, range [-90,90)
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_REGENFULLIMG(img,mask) 'generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'region binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element of No.3 in the list ZV_REELLIPAXIS(re,0) 'calculate the equivalent ellipse axis parameters of the area, and put the parameters in turn into the TABLE whose index start position is 0

8.6.11. ZV_RERECT – External Rectangle

Type	Feature
Description	Calculate the minimum circumscribed moment of the area parallel to the horizontal axis, that is, the smallest rectangle parallel to the horizontal axis that can enclose the area

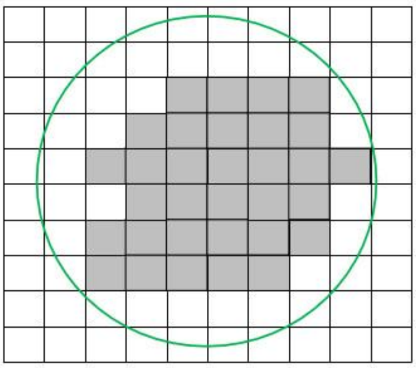
	
Grammar	ZV_RERECT(re, tabId) re: ZVOBJECT type, region tabId: TABLE index, output parameter, the output is in order of x, y, width, height, that is, the x coordinate of the upper left corner of the rectangle, the y coordinate of the upper left corner of the rectangle, the width of the rectangle, and the height of the rectangle
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_REGENFULLIMG(img,mask) 'generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element of No.3 in the list ZV_RERECT(re,0) 'calculate the minimum external moment of the area parallel to the horizontal axis, and put the parameters into the TABLE with the index starting position 0

8.6.12. ZV_RERECT2 – Minimal External Rectangle

Type	Feature
Description	Calculate the minimum circumscribed rectangle, the smallest rectangle can be with an angle, that is, the smallest rectangle with an angle that can enclose the area. The calculation method is to use the pixel center coordinates to calculate the half-width. The returned length and width dimensions are twice the corresponding half-width plus 1, that is, half-width * 2 + 1. Since the area point has an area, the final length and width dimensions are used as the length of the circumscribed rectangle of the area. This is reasonable, but it should be noted that in the case of an angle, the actual area is not completely included in the outer rectangle, because the boundary pixels still have small sharp corners outside the rectangle. For example, at an angle of 45°, the length of the exceeded part is $1 / \sqrt{2}-0.5$ 
Grammar	ZV_RERECT2(re, tabId) re: ZVOBJECT type, region tabId: TABLE index, output parameter, the output is in order of cx, cy, width, height, angle, that is, the cx coordinate of the center the rectangle, the cy coordinate of the center the rectangle, the width of the rectangle, and the height of the rectangle, the angle of the rectangle, angle clockwise is positive, the unit is the degree, longer is the width, shorter is the height, angle range [-90, 90]
Controller	It is valid in controllers that support ZV function or they belong

	to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_REGENFULLIMG(img,mask) 'generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element of No.3 in the list ZV_RERECT2(re,0) 'calculate the minimum external moment of the area, and put the parameters into the TABLE (0)

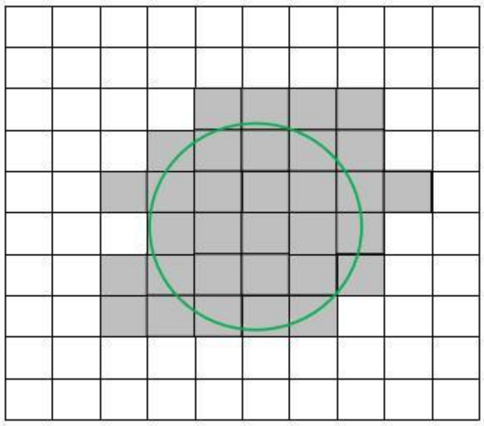
8.6.13. ZV_RECIRCLE – External Circle

Type	Feature
Description	Calculate the minimum circumscribed circle, that is, the smallest circle that can enclose the area. The same as the calculation principle of the minimum circumscribed moment, the calculation is based on the pixel center coordinates. It should also be noted that the actual area is not completely included in the circumscribed circle, and the boundary pixels still have small sharp corners outside the circle, exceeding part of the length is 1 / $\sqrt{2}$-0.5. 

Grammar	ZV_RECIRCLE(re, tabId) re: ZVOBJECT type, region tabId: TABLE index, output parameter, the output is in order of cx, cy, radius, that is, the cx coordinate of the circle center, the cy coordinate of the circle center, circle radius.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_REGENFULLIMG(img,mask) 'generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element of No.3 in the list ZV_RECIRCLE(re,0) 'calculate the minimum external circle of the area, and put the parameters into the TABLE (0)

8.6.14. ZV_REINNERCIRCLE – Inner Circle

Type	Feature
Description	Calculate the maximal inner circle, that is, the largest inner circle that is enclosed by the area. Note: this operator is time-consuming.

	
Grammar	ZV_REINNERRECT(re, tabId) re: ZVOBJECT type, region tabId: TABLE index, output parameter, the output is in order of cx, cy, radius, that is, the cx coordinate of the circle center, the cy coordinate of the circle center, circle radius.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_REGENFULLIMG(img,mask) 'generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element of No.3 in the list ZV_REINNERCIRCLE(re,0) 'calculate the maximal inner circle of the area, and put the parameters into the TABLE (0)

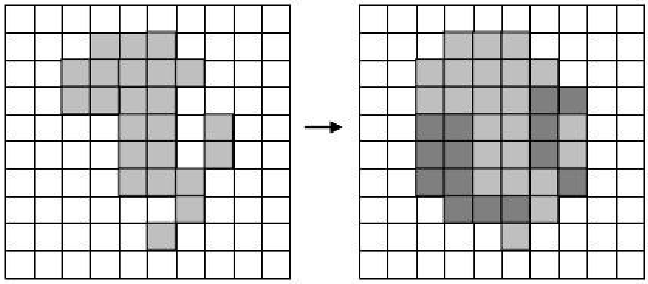
8.6.15. ZV_RECCLTY – Circularity

Type	Feature
Description	Calculate the shape factor of the region – circularity /

	roundness, which indicates how similar the region is to the circle. Assuming F is the square of the area and max is the maximum distance from the center to all contour, circularity C is defined as: $$C' = \frac{F}{(max^2 * \pi)}$$ C = min (1, C')
Grammar	ZV_RECCLTY(re, tabId) re: ZVOBJECT type, region tabId: TABLE index, output parameter, range is [0,1], the larger the value, the more circular it is, the empty area is 0, the strip area is less than 1, and the circle area is 1
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_REGENFULLIMG(img,mask) 'generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element of No.3 in the list ZV_RECCLTY(re,0) 'calculate the circularity of the area, and put the parameters into the TABLE (0)

8.6.16. ZV_RECONVEXITY – Convexity

Type	Feature
Description	Calculate the shape factor of the region – convexity, area's square / the area corresponding to the convex hull. Assuming F_c is the square of the convex hull and F_o is the original square of

	the area, then convexity C is defined as: $C = \frac{F_o}{F_C}$ 
Grammar	ZV_RECONVEXITY(re, tabId) re: ZVOBJECT type, region tabId: TABLE index, output parameter, range is [0,1], the larger the value, the convexity it is, the empty area is 0.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_REGENFULLIMG(img,mask) 'generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element of No.3 in the list ZV_RECONVEXITY(re,0) 'calculate the convexity of the area, and put values into the TABLE (0)

8.6.17. ZV_RECMPTNS – Compactness

Type	Feature
Description	Calculate the shape factor of the region – compactness. Assuming L is the length of the area contour and F is the

	square of the area, then compactness C is defined as: $$C' = \frac{L^2}{4F\pi}$$ C = max (1, C')
Grammar	ZV_RECMPINS(re, tabId) re: ZVOBJECT type, region tabId: TABLE index, output parameter, the empty area is 0.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_REGENFULLIMG(img,mask) 'generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element of No.3 in the list ZV_RECMPINS(re,0) 'calculate the compactness of the area, and put values into the TABLE (0)

8.6.18. ZV_RERECTLTY – Rectangularity

Type	Feature
Description	Calculate the rectangularity of an area, that is, a measure of how close a shape is to being rectangular. The calculation of the rectangularity is ultimately based on the area of the normalized difference between the calculated rectangle and the input region with respect to the area of the rectangle.
Grammar	ZV_RERECTLTY (re, tabId) re: ZVOBJECT type, region tabId: TABLE index, output parameter, the empty area is 0.
Controller	It is valid in controllers that support ZV function or they belong

	to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_REGENFULLIMG(img,mask) ‘generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element of No.3 in the list ZV_RERECTLTY(re,0) 'calculate the rectangularity of the area, and put values into the TABLE (0)

8.6.19. ZV_REECCENTRICITY – Shape Parameter

Type	Feature
Description	Calculate the shape feature parameters of the area, that is, the shape feature that is derived by ellipse parameter. Ra and Rb represent the semi-major and semi-minor axes of the ellipse, and A represents the aera of the region. $Anisometry = \frac{Ra}{Rb}$ $Bulkiness = \frac{\pi \cdot Ra \cdot Rb}{A}$ $StructureFactor = Anisometry \cdot Bulkiness - 1$
Grammar	ZV_REECCENTRICITY (re, tabId) re: ZVOBJECT type, region tabId: TABLE index, output parameter, anisometry, bulkiness, structFactor are output in order, that is, ellipse length axis ratio, fluffy factor, structure factor.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0)

	'read the picture in the original image format ZV_REGENFULLIMG(img,mask) 'generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element of No.3 in the list ZV_REECCENTRICITY(re,0) 'calculate the shape feature parameter of the area, and put values into the TABLE (0)

8.6.20. ZV_REMOM2INVAR – Invariant 2rd Moment

Type	Feature
Description	Calculate the scale-invariant 2rd moment of the region. Z_0 and S_0 are the coordinate of area center R, the area is F, then the definition of moment is: $$M_{ij} = \frac{1}{1=2} \sum_{(z,5)\in R} (Z_0 - Z)^i (S_0 - S)^j$$
Grammar	ZV_REMOM2INVAR (re, tabId) re: ZVOBJECT type, region tabId: TABLE index, output parameter, m11, m20, m02, re11, re12, foreGoundPixNum are output in order.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_REGENFULLIMG(img,mask) 'generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area

	ZV_LISTGET(reConnect,re,3) 'get the element of No.3 in the list ZV_REMOM2INVAR(re,0) 'calculate the scale-invariant 2rd moment of the region, and put values into the TABLE (0)

8.6.21. ZV_REMOM3INVAR – Invariant 3st Moment

Type	Feature
Description	Calculate the scale-invariant 3st moment of the region.
Grammar	ZV_REMOM3INVAR (re, tabId) re: ZVOBJECT type, region tabId: TABLE index, output parameter, m21, m12, m30, m03 are output in order.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_REGENFULLIMG(img,mask) 'generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element of No.3 in the list ZV_REMOM3INVAR(re,0) 'calculate the scale-invariant 3st moment of the region, and put values into the TABLE (0)

8.6.22. ZV_REMOMCENTRA – Center Moment

Type	Feature

Description	Calculate the center moment of the region.
Grammar	ZV_REMOMCENTRA (re, tabId) re: ZVOBJECT type, region tabId: TABLE index, output parameter, center1, center2, center3, center4 are output in order.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_REGENFULLIMG(img,mask) ‘generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET(reConnect,re,3) 'get the element of No.3 in the list ZV_REMOMCENTRA(re,0) 'calculate the center moment of the area, and put values into the TABLE (0)

8.7. Transformation

8.7.1.ZV_RESORT – Sorting

Type	Feature
Description	Sort the region according to Feature.

Grammar	ZV_RESORT (relist, feature, isInc) relist: ZVOBJECT type, area list to be sorted, list type feature: sorted feature type, please see below form isInc: whether is the ascending order, 0 means descending order, otherwise is the ascending order.

0	Area	17	External rectangle height
1	Gravity X	18	External rectangle right bottom x
2	Gravity Y	19	External rectangle right bottom y
3	Angle	20	External rectangle h / w
4	Perimeter	21	Rotary rectangle center X
5	Circularity	22	Rotary rectangle center Y
6	Compactness	23	Rotary rectangle width
7	Rectangularity	24	Rotary rectangle height
8	Convexity	25	Rotary rectangle angle
9	--	26	Rotary rectangle h / w
10	Equivalent elliptic main axis length	27	External circle center x
11	Equivalent elliptic slave axis length	28	External circle center y
12	Equivalent elliptic main axis angle	29	External circle radius r
13	Equivalent elliptic semi-main axis / semi- slave axis	30	--
14	External rectangle x	31	Convex hull area
15	External rectangle y	32	Hole numbers
16	External rectangle width		

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_REGENFULLIMG(img,mask)

	‘generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_RESORT(reconnect,0,0) 'sort the regions in descending order by area feature

8.7.2.ZV_REFILTER – Filtering

Type	Feature
Description	The regions in the region list are filtered by a certain feature, and the regions that meet the feature requirements are reserved.
Grammar	ZV_REFILTER(relist,feature,min,max,isInvert) relist: ZVOBJECT type, which indicates the list of areas to be filtered, list type feature: the feature type of an area, please refer to “ZV_RESORT”. The value can be -1 min: the lower limit of the feature value max: the upper limit of the feature value isInvert: whether to reverse the selection. If the value is 1, the contour that is not in the range is retained.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img, mask, re, reBin, reConnect ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_REGENFULLIMG(img,mask) ‘generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect)

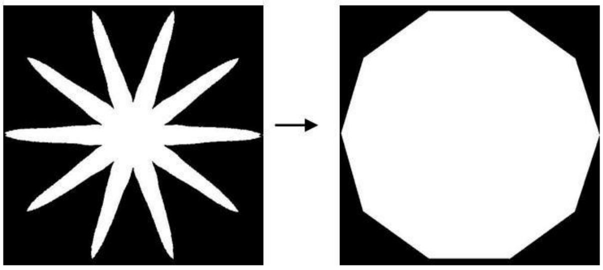
	'calculate the connected area of the area ZV_REFILTER(reConnect,0,1500,3000,0) 'filter regions in area list, and retain regions of 15000-3000, then regions that are not in this range will be filtered.

8.7.3.ZV_REGETPTS – Region Point Set

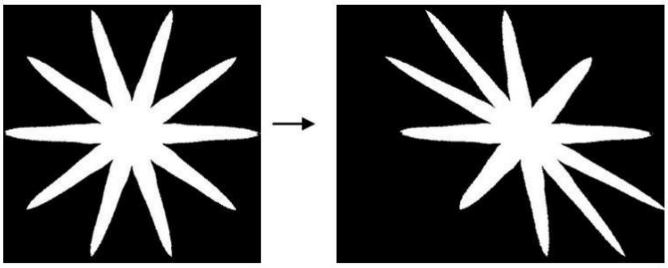
Type	Feature
Description	Convert region into point set, region means each pixel position is converted into one coordinate point.
Grammar	ZV_REGETPTS (re, pts) re: ZVOBJECT type, region pts: ZVOBJECT type, matrix, a matrix of N rows 2 columns, the first row is x coordinate, the second column is the y coordinate.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, re, reBin, reconnect, pts ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_REGENFULLIMG(img,mask) ‘generated region that covers the whole image ZV_RETHRESH(img,mask,reBin,0,100) 'area binarization ZV_RECONNECT(reBin,reConnect) 'calculate the connected area of the area ZV_LISTGET (reconnect, re, 3) ‘get the element of No.3 in the list ZV_REGETPTS(re, pts) 'output region in the form of point set

8.7.4.ZV_RESHAPETRANS – Region Transformation

Type	Feature

<table>
<tr><td>Description</td><td>Convert region into specified type.</td></tr>
<tr><td>Grammar</td><td>ZV_RESHAPETRANS (re, reTrains, type)
re: ZVOBJECT type, region, input parameter
reTrains: ZVOBJECT type, region, output parameter
type: region transformation type
<table>
<tr><td>0</td><td>Convex hull</td></tr>
<tr><td>1</td><td>The external moment of the parallel horizontal axis of the minimum enclosed region</td></tr>
<tr><td>2</td><td>The rotational external moment of the minimum enclosed region</td></tr>
<tr><td>3</td><td>Maximum internal moments surrounded by regions (not supported now)</td></tr>
<tr><td>4</td><td>The circumscribed circle of the smallest enclosed area.</td></tr>
<tr><td>5</td><td>The largest inner circle surrounded by the region.</td></tr>
</table></td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
<tr><td>Example</td><td>
ZVOBJECT img, dst, mask, re, reTrans
ZV_READIMAGE(img, "test.png", 0)
'read the picture in the original image format
ZV_IMGINFO (img, 0) ‘get basic information of image
ZV_REGENFULLIMG(img,mask)
‘generated region that covers the whole image
ZV_RETHRESH(img,mask,re,200,255) 'area binarization
ZV_RESHAPETRANS (re, reTrans, 0)
‘convert region to convex hull region</td></tr>
</table>

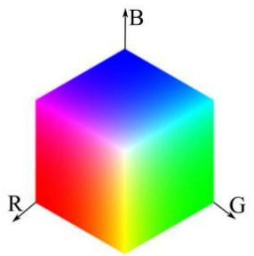
8.7.5.ZV_REAFFINE – Region Affine Transformation

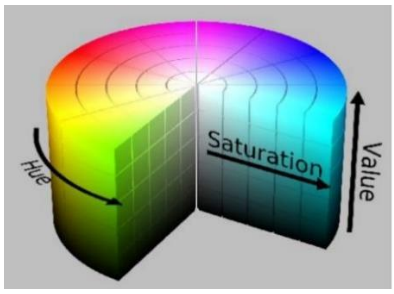
Type	Feature
Description	Affine and transform region to generate new region, like, translate, rotate, zoom in and out region.
Grammar	ZV_REAFFINE (re, mat, reAffine) re: ZVOBJECT type, region mat: ZVOBJECT type, transform matrix, 2 rows 3 columns reAffine: ZVOBJECT type, transformed region
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT mat img, dst, mask, re, reAffine ZV_READIMAGE(img, "test.png", 0) 'read the picture in the original image format ZV_IMGINFO (img, 0) 'get basic information of image ZV_REGENFULLIMG(img,mask) 'generated region that covers the whole image ZV_RETHRESH(img,mask,re,200,255) 'area binarization TABLE (10, 1, 0.5, 0, 0, 1, 0) 'save data into TABLE (10) ZV_MATGENDATA (mat, 2, 3, 10) 'transform matrix, miscut in the x direction ZV_REAFFINE (re, mat, reAffine) 'affine transform the region

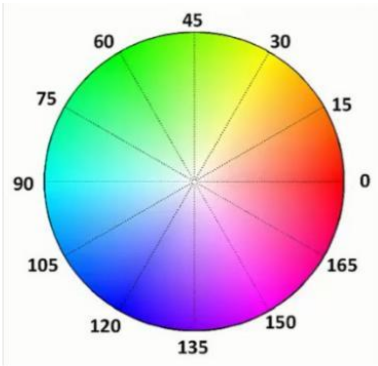
Chapter VIIII Color

9.1.ZV_CLRGENMODEL – Generate Color Model

<table>
<tr><th>Type</th><td>Segment</td></tr>
<tr><th>Description</th><td>Generate color model through threshold range, it supports RGB and HSV color space.







H channel hue circle</td></tr>
<tr><th>Grammar</th><td>ZV_CLRGENMODEL(mod,name,colorType,low1,high1,low2,high2,low3,high3)
mod: ZVOBJECT type, output, generated color model
name: mode name, character string, it can't be empty, that is, “”, or “?”.
colorType: color type, 0 – RGB, 1 – HSV
low1: low threshold, RGB: R channel, range [0, 255]. HSV: H channel, range [0, 180].
high1: high threshold, RGB: R channel, range [0, 255]. HSV: H channel, range [0, 180]. The H channel is a 360° hue ring, and the parameters are expressed as 0-180. Due to the particularity of parameter expression, high1 can be smaller than low1, which means that it spans the interval near 180, that is, [low1,</td></tr>
</table>

	high1+180]. low2: low threshold, RGB: G channel, range [0, 255]. HSV: S channel, range [0, 255]. high2: high threshold, RGB: G channel, range [0, 255]. HSV: S channel, range [0, 255]. low3: low threshold, RGB: B channel, range [0, 255]. HSV: V channel, range [0, 255]. high3: high threshold, RGB: B channel, range [0, 255]. HSV: V channel, range [0, 255].
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT clrMod ZV_CLREGNMODEL (clrMod,"red",0,230,255,0,20,0,20) 'generate red model from RGB

9.2.ZV_CLRGENMODELRE – Generate Color Model

Type	Segment
Description	Generate color model through training some image information.
Grammar	ZV_CLRGENMODELRE(img,mask,mod,name,colorType) img: ZVOBJECT type, 3-channel image mask: ZVOBJECT type, specify valid region "region" in image that is used to generate color model, "region" can be made by "region" command in Chapter VIII. mod: ZVOBJECT type, output, generated color model name: mode name, character string, it can't be empty, that is, "", or "?". colorType: color type, 0 – RGB, 1 – HSV
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, re, clrMod ZV_READIMAGE (img, "test.png", 0) 'read image in the original form ZV_REGENRECT(re,0,0,100,100)

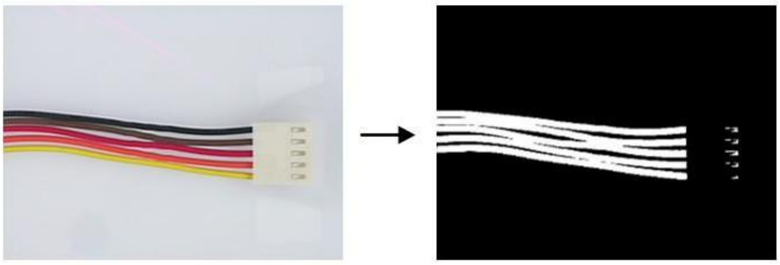
	ZV_CLREGNMODELRE (img, re, color_mod, “color”, 0) ‘generate color model from RGB

9.3.ZV_CLRGETMODELPARAM – Get Color Model Parameters

Type	Segment
Description	Read color model’s parameters.
Grammar	ZV_CLRGETMODELPARAM(mod,maxLen,tabName,tabParam) mod: ZVOBJECT type, color model maxLen: name buffer size, the size should be appropriate, if it is more than or less than model name, it will report an error. tabName: the starting index of the TABLE where the model name information is placed. tabParam: the starting index of the TABLE where the model parameter information is placed, and the TABLE space stores color type and the low and high thresholds of each channel sequentially, such as coloType, low1, high1, low2, high2, low3, high3.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, re, clrMod ZV_READIMAGE (img, “test.png”, 0) ‘read image in the original form ZV_REGENRECT(re,0,0,100,100) ZV_CLREGNMODELRE (img, re, clrMod, “color”, 0) ‘generate color model from RGB ZV_CLRGETMODELPARAM(clrMod,7,0,10) ‘get color parameter information

9.4.ZV_CLRMODELTHRESH – Color Binarization

Type	Segment

Description	Use color model to binarize RGB image and then generate the region.
Grammar	ZV_CLRMODELTHRESH(mod,img,mask,region) mod: ZVOBJECT type, color model or color model list, for performance, the colors in the color list must be the same img: ZVOBJECT type, 3-channel RGB image mask: ZVOBJECT type, needed “region”, assign the region in img to be binarized, when mask is NULL, the whole image is valid. region: ZVOBJECT type, output region
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img, dst, clrMod, mask, reBin, tmp1, tmp2 ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_IMGINFO(img,0) 'get basic information of image ZV_CLRGENMODEL(clrMod,"color",0,143,255,136,255,143,255) 'generate color template ZV_REGENFULLIMG(img,mask) 'generate an area covering the entire image ZV_CLRMODELTHRESH(clrMod,img,mask,reBin) 'color binarization ZV_RETOIMG(reBin,tmp1,TABLE(0), TABLE(1)) 'region to binary image ZV_IMGCOPY(tmp1,tmp2) 'assignment image ZV_IMGSETCONST(tmp2,255) 'constant fill image ZV_ABSDIFF(tmp2,tmp1,dst,1) 'image difference ZV_IMGCOPY(tmp1,tmp2) ZV_IMGSETCONST(tmp2,255) ZV_ABSDIFF(tmp2,tmp1,dst,1)

9.5.ZV_CLRMODELCLASSIFY – Color Classification Recognition

Type	Segment
Description	Use the color model list to identify the colors in the area, that is, one certain color in the region matches with color list the most. If the recognition fails, "?" will be output to indicate that it cannot be recognized.
Grammar	ZV_CLRMODELCLASSIFY (colorList, img, mask, maxLen, tab_name, tabId,score) colorList: ZVOBJECT type, color model list img: ZVOBJECT type, 3-channel RGB image mask: ZVOBJECT type, needed "region", specify the img image to be identified, it cannot be empty. maxLen: the maximum length of the output parameter tab_name TABLE that can be used tab_name: color name, output parameters, TABLE index, the index that stores color, name, and parameters. tabId: color id, output parameter, TABLE index, parameter index for storing color id score: recognition score. If the recognition score is less than the given score, the recognition fails and the output result is "?", otherwise the color name is output normally.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask, clrMod, clrList ZV_READIMAGE(img, "test.png", 0) 'read the image in the original image format ZV_REGENRECT(mask,0,0,100,100) ZV_CLRGENMODEL(clrMod,"red",0,230,255,0,20,0,20) ZV_LISTINSERT(clrMod, clrListt,-1) ZV_CLRMODELCLASSIFY(colorList,img,mask,10,0,20,80) 'color recognition

Chapter X Contour

Contour, as a type of visual variable ZVOBJECT, its type is 4 (which can be viewed by the ZV_TYPE command). It is a data structure that can store a series of point sets, which can be divided into pixel contours and sub-pixel contours.

And there are three properties of contour:

Property 1: segmentation, curve type (not segmented, usually the contours are extracted directly from the image), line type and arc type, indicating whether the contour has been divided by the division operator, please refer to ZV_CONTSEGMENT.

Property 2: multilateral shape, indicating whether the contour is a polygon, that is, whether the contour point set is a continuous dense point set, and the continuous dense point set is non-polygons, then non-continuous sparse point sets are polygons, such as the property of those contours that are processed by polygon approximation "ZV_CONTAPPROXPOLY" is a polygon.

Property 3: closed and non-closed. Therefore, some contour commands can only be used on objects with specific properties.

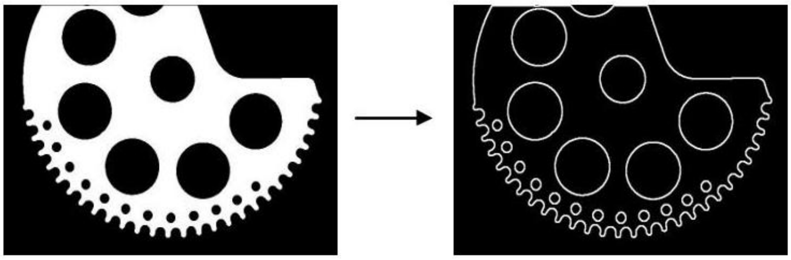
Property 1	
Curve type	Contours extracted from the image or processed by the parallel expansion command.
Line type	The contour is segmented by the segmentation operator, and contour point set can be replaced by line segments under a certain accuracy.
Arc type	The contour is segmented by the segmentation operator, and contour point set can be replaced by arc segments under a certain accuracy.
Property 2	
Polygon	The point set is non-continuous and sparse, with fewer point sets and fast processing speed.
Non-polygon	The point set is continuous and dense, with many point sets and slow processing speed.

Property 3	
Closed	The starting point and end point of the outline are the same or the distance is less than 1 pixel.
Non-closed	It is opposite to “closed”.

10.1. Contour

10.1.1. ZV_CONTGEN – Generate Contour

Type	Contour
Description	Extract the contour of the foreground target in the binary image, that is, the boundary of the white part. The foreground of binary image is the white part of the image, so the extracted contour is a closed contour that surrounds the white part. If the image boundary is also the foreground, the boundary will also be extracted as a contour. Note: If there are more noise or burrs at the edge of the image contour, it can combine the image morphology processing operator to remove them, such as, opening operation or closing operation, then extract the contour.
Grammar	ZV_CONTGEN(img,contlist,mode,appro) img: ZVOBJECT type, single-channel 8U type, binarization image contlist: ZVOBJECT type, list type, output, multiple contours are stored into the list, and the property of the contour depends on parameter “appro”. mode: contour extracting method: 0: external contour, that is, inside contour surrounded by external contour will be filtered, only external contour will be retained, 1: all contours appro: contour expression method: 0: point set, the contour is expressed by a series of point set, 1: concise, the contour is also expressed by a series of point set, but horizontal, vertical and diagonal are simplified as two terminals, it is recommended

	to use 0. When appro is 0, the property contlist is curve type and non-polygon type. When appro is 1, the property contlist is curve type and polygon type.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img, img_bw, dst, contlist, con_src ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,img_bw,200,255) 'image binarization ZV_CONTGEN(img_bw,contlist,1,0) 'save all the found contours into the contour list ZV_IMGCOPY(img,dst) 'copy the image ZV_IMGSETCONST(dst,0) 'constant fill image ZV_CONTLIST(dst,contlist,255,0) 'draw contour

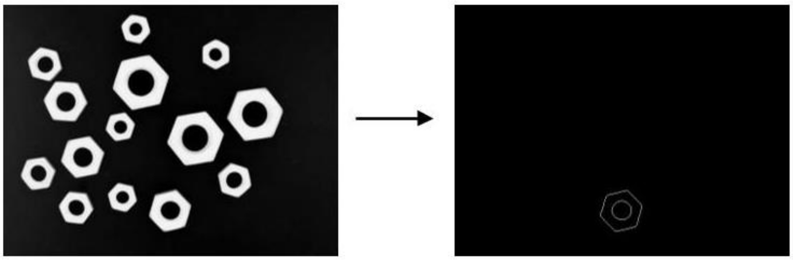
10.1.2. ZV_CONTGENEX – Generate Contour

Type	Contour
Description	Extract the contour of the foreground target in the ROI assigned range of binary image, that is, the boundary of the white part. The foreground of binary image is the white part of the image, so the extracted contour is a closed contour that surrounds the white part. And ROI can't exceed image range, otherwise, it will report an error. Note: If there are more noise or burrs at the edge of the image contour, it can combine the image morphology processing operator to remove them, such as, opening operation or closing operation, then extract the contour.

Grammar	ZV_CONTGENEX (img, contlist, mode, appro, cx, cy, width, height, angle) img: ZVOBJECT type, single-channel 8U type, binarization image contlist: ZVOBJECT type, list type, output, multiple contours are stored into the list, and the property of the contour depends on parameter "appro". mode: contour extracting method: 0: external contour, that is, inside contour surrounded by external contour will be filtered, only external contour will be retained, 1: all contours appro: contour expression method: 0: point set, the contour is expressed by a series of point set, 1: concise, the contour is also expressed by a series of point set, but horizontal, vertical and diagonal are simplified as two terminals, it is recommended to use 0. When appro is 0, the property contlist is curve type and non-polygon type. When appro is 1, the property contlist is curve type and polygon type. cx: ROI center x coordinate cy: ROI center x coordinate width: ROI width height: ROI height angle: ROI angle
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

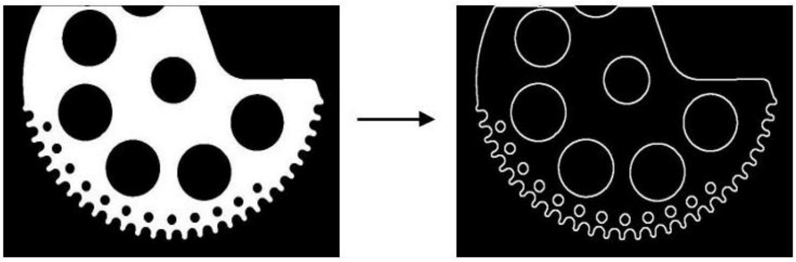
10.1.3. ZV_CONTGENSUBPIX– Sub-Pixel Contour

Type	Contour
Description	Sub-pixel edge contour extraction uses the canny edge detection algorithm with hysteresis threshold to extract the sub-pixel edge contour of the specified region in the image. If the gradient of the edge contour point is greater than the high threshold, it must be a contour, and if it is less than low, it must not be a contour. If it is between low and high and connected to

	the contour edge points are also considered contour points, the extracted contour may not be in the clockwise direction (under the image coordinate system). If you need to check the contour direction, please use the ZV_CONTDIRECT command. Note: If there are more noise or burrs at the edge of the image contour, it can combine the image morphology processing operator to remove them, such as, opening operation or closing operation, then extract the contour.
Grammar	ZV_CONTGENSUBPIX(img, region, contlist, low, high, minLen) img: ZVOBJECT type, source-gray image, single-channel 8U type region: ZVOBJECT type, extract valid region of contour, that is, the image assigned by "region" will be extracted, when "region" is empty, entire image contour is extracted. contlist: ZVOBJECT type, list type, output, multiple contours are stored into the list, the contour property is curve type and non-polygon type. low: low threshold for hysteresis threshold, (0, 255] high: high threshold for hysteresis threshold, (0, 255], > low minLen: the minimal contour length, which means extracted contour is ≥ minContLen
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img, dst, re, contlist ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_REGENRECT(re, 263, 336, 114, 109) 'generate rectangle region ZV_CONTGENSUBPIX(img, re, contlist, 80, 200, 30) 'extract the minimal edge contour of 30 from valid region,

	and save the result into list ZV_IMGCOPY(img,dst) 'copy the image ZV_IMGSETCONST(dst,0) 'constant fill image ZV_CONTLIST(dst,contlist,255,0) 'draw contour

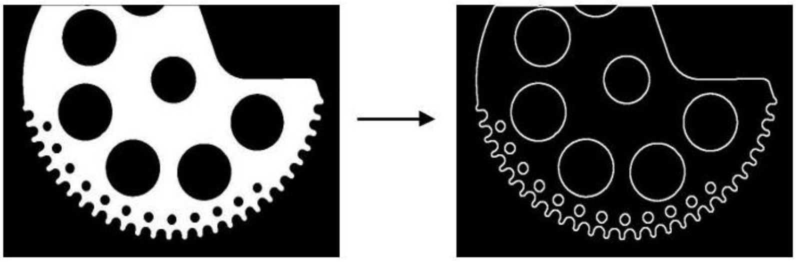
10.1.4. ZV_CONTGAUSSIAN– Contour Gaussian Smoothing

Type	Contour
Description	To do gaussian smoothing for contour, it can smooth obtrusive points in the contour.
Grammar	ZV_CONTGAUSSIAN(src,dst,size) src: ZVOBJECT type, input, it only supports non-polygon property and any contours of other properties. dst: ZVOBJECT type, output, the property is curve type and non-polygon type. size: the size of gaussian, the size is bigger, the smoothing is obvious, and it is not negative, 3, 5 and 7 are recommended, the default value is 3. If 1 is filled, output contour and input contour are the same.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 DIM contCnt ZVOBJECT img, imgBw, dst, contList, contSrc, contDst ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,0,0)

	'store all found outer contours in the contour list contCnt= ZV_LISTCOUNT(contList) 'get the number of contour lists ZV_IMGCOPY(img,dst) 'copy image ZV_IMGSETCONST(dst,0) 'constant fill image FOR i = 0 TO contCnt-1 ZV_LISTGET(contList, contSrc,i) 'get a certain contour ZV_CONTGAUSSIAN(contSrc,contDst,3) 'use a Gaussian filter with a Gaussian kernel size of 3 to smooth the contour and stores the result in con_dst ZV_CONTOUR(dst,contDst,255) 'draw the contour NEXT

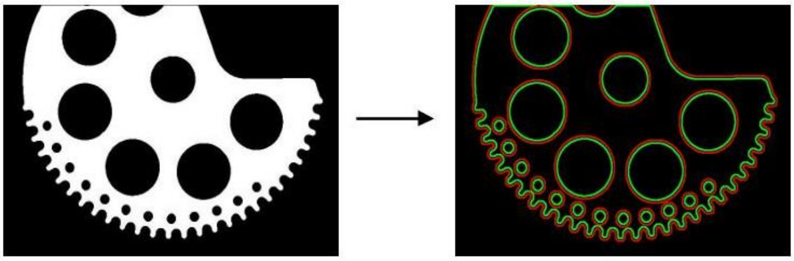
10.1.5. ZV_CONTAPPROXPOLY – Polygon Approximation

Type	Contour
Description	Polygonal approximation of a contour or list of contours, that is, dividing the contour into lines with a certain accuracy, then the polygon formed by these line segments can approximate the contour very well.
Grammar	ZV_CONTAPPROXPOLY(src,dst,eps) src: ZVOBJECT type, contour or contour list, input dst: ZVOBJECT type, contour or contour list, output, properties are curve and polygon, please note that although the point set becomes sparse after approximation, it is still defined as a curve property. eps: the accuracy of contour segmentation. The smaller the segmentation accuracy, the more polygon line segments will be segmented, and the closer the polygon is to the contour, common values are 1, 1.5, and 2, eps is a floating point number greater than or equal to 0, and when the parameter is set to 0, it means that the input is equal to the output.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	 ZVOBJECT img, dst, imgBw, contSrc, contDst ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,150,255) 'image binarization ZV_CONTGEN(imgBw,conSrc,1,0) 'generate the contour ZV_CONTAPPROXPOLY(contSrc, conDst, 1) ‘to do polygon approximation for contour or contour list src to generate dst ZV_IMGCOPY(img,dst) 'copy image ZV_IMGSETCONST(dst,0) 'constant fill image ZV_CONTLIST,contDst,255,0) 'draw the contour

10.1.6. ZV_CONTGENPARALLEL – Generate Parallel Contour

Type	Contour
Description	Generate a new parallel contour that expands or shrinks in a certain distance from the input contour. If the input contour processes polygonal approximation through ZV_CONTAPPROXPOLY, parallel or inward processing will be faster. And the contour point set processed by the command may become non-continuous, but the number of point sets is the same as the input.
Grammar	ZV_CONTGENPARALLEL(src,dst,dist) src: ZVOBJECT type, input, 2 contour points at least dst: ZVOBJECT type, output, property 1 is curve type, other attributes are the same as input dist: expansion or contraction distance, the unit of distance

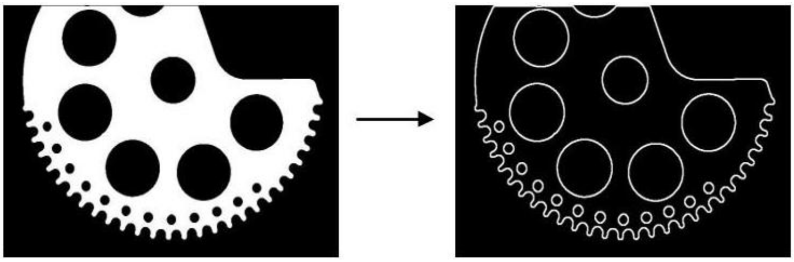
	is consistent with the unit of contour point, positive value is parallel expansion, negative value means parallel contraction. When it is 0, the output contour is the same as the input contour.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 DIM contCnt ZVOBJECT img, dst, imgBw, contList, contSrc, contDt ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,150,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'generate contour contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists ZV_IMGCOPY(img,gray) 'copy image ZV_IMGSETCONST(gray,0) 'constant fill image ZV_GRAYTORGB(gray,dst) 'convert grayscale image to color image FOR i = 0 TO contCnt-1 ZV_LISTGET(contList, conSrc,i) 'get a certain contour ZV_CONTOUR(dst,conSrc,ZV_COLOR(0,255,0)) 'draw the original image contour as green ZV_CONTGENPARALLEL(conSrc,conDst,5) ‘generate a new contour extending parallel to the input contour by a distance of 5 ZV_CONTOUR(dst,conDst,ZV_COLOR(255,0,0)) 'draw parallel contours in red NEXT

10.1.7. ZV_CONTSETMAXRADIUS – Set Max Arc Radius

Type	Contour
Description	Set the maximum radius when the contour is divided into arc primitives. Arcs larger than the maximum radius will not be divided into arc segments, but will be divided into multiple straight-line segments. And use it with the ZV_CONTSEGMENT instruction.
Grammar	ZV_CONTSETMAXRADIUS(radius) radius: max main axis' radius, range is (0, 16383], the default value is 1000
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_CONTSETMAXRADIUS(1000) 'set the max radius is 1000 for the arc

10.1.8. ZV_CONTSEGMENT – Contour Segment

Type	Contour
Description	Divide a continuous curved contour into segments of small primitives such as straight lines and arcs. Primitive element: one small contour segment that can be fitted with a straight line under a given accuracy is called a straight-line primitive, and the same rule for arc and ellipse primitive. A primitive is also a segment of contour, such as the segmentation property of a line primitive is line type. Therefore, contour segmentation produces a series of primitives, and these primitives are stored in the list for outputting. The contour segmentation command can usually be used in conjunction with the Gaussian smoothing command ZV_CONTGAUSSIAN. Gaussian smoothing can smooth contour points with large jitters to a certain extent, thereby segmenting the contours more smoothly.

Grammar	ZV_CONTSEGMENT(cont,list,type,eps1,eps2) cont: ZVOBJECT type, input, it only supports non-polygonal property contour, or when segment type “type” is 0, the contour that is polygon is supported. list: ZVOBJECT type, list type, output, segmented primitives are stored into the list. type: segment type, the type of primitive that is segmented by contour, 0 and 1 are used most. 0: line, 1: line or arc eps1: the accuracy of polygon approximation, that is, the accuracy of dividing a curve outline into small line segments. The smaller the accuracy value, the more accurate it will be for subsequent outline primitives such as arc segmentation. Commonly used values are 1, 1.5, 2, and the recommended value is 1. eps2: the accuracy of merging polygonal contours into primitives. For example, the accuracy of fitting a certain contour into an arc. That is, the accuracy of fitting this contour into an arc is less than or equal to eps2. Then this contour is represented by arc primitives, and the common value is 1, 1.5, 2, eps2 should be greater than or equal to eps1.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img, img_bw, dst, contlist, contlist_seg, con_src DIM con_count ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,img_bw,150,255) 'image binarization ZV_CONTGEN(img_bw,contlist,1,0) 'generate contour ZV_CONTSETMAXRADIUS(1000) 'set the maximum radius of the arc to 1000

	con_count = ZV_LISTCOUNT(contlist)
	    'get the number of contour lists
	ZV_IMGCOPY(img,dst) 'copy image
	ZV_IMGSETCONST(dst,0) 'constant fill image
	FOR i = 0 TO con_count-1
	    ZV_LISTGET(contlist, con_src,i) 'get a certain contour
	    ZV_CONTSEGMENT(con_src,contlist_seg,1,1,1)
	        'split the contour into straight line and arc primitives
	    ZV_CONTLIST(dst,contlist_seg,255,0) 'draw contour
	NEXT

10.1.9. ZV_CONTGETPARAM – Contour Primitive Geometric Parameters

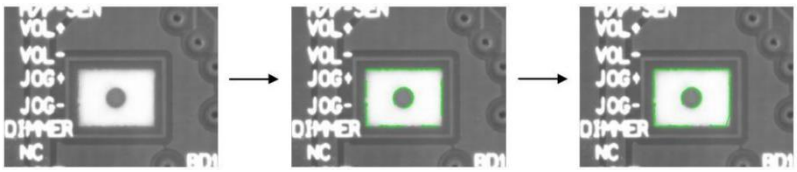
Type	Contour
Description	Obtain the geometric parameters of the contour segmentation primitive. If the straight-line primitive can be replaced by a straight-line segment, then its geometric parameters are the coordinates of the two terminals of the straight-line segment. For the arc primitive, it is the center point, radius, starting point, midpoint, and end point. The direction of the arc primitive from the starting point to the end point is clockwise.

Grammar	ZV_CONTGETPARAM (cont, len, tabId)
	    cont: ZVOBJECT type, input
	    len: store the buffer length of geometric primitive parameters
	    tabId: output geometric primitive parameters, they are type, param1, param2, param3…, that is, segment type and primitive parameter, and primitive parameter is related to contour type, type is shown below:
	<table><tr><td>type</td><td>Primitive parameter</td></tr><tr><td>-1</td><td>Curve: no parameters output, this contour is not segmented</td></tr><tr><td>0</td><td>Line primitive: stx, sty, endx, endy</td></tr></table>

	<table><tr><td>1</td><td>Arc primitive: cx, cy, radius, stx, sty, midx, middy, endx, endy</td></tr></table>
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM contCnt ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i=0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour ZV_CONTGETPARAM(cont,5,0) 'get the contour type and geometric primitive parameters NEXT

10.1.10. ZV_CONTUNIONADJ -- Neighbor Contour Connection

Type	Contour
Description	Connect contours that are close to terminal. If the current contour can be connected to multiple contours, the closest contour will be considered first. If the distance between multiple contours is the same, the longest contour will be considered first. The output contours are all clockwise.
Grammar	ZV_CONTUNIONADJ(src,dst,mode,maxDist) src: ZVOBJECT type, list type, input, contours with polygon property are not supported dst: ZVOBJECT type, list type, output, same as input

	property mode: 1-the first and end points are taken into consideration, that is, the connection will occur only when the distance between the current contour terminal and the terminal of another contour is less than the distance between the start and end points of another contour. 0-do not detect the start and end points, mode 0 is recommended. maxDist: the maximum distance between two contours that satisfies the closest connection.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img, re, dst, contSrc, contDst ZV_READIMAGE(img, "test.bmp",0) 'read the image in the original image format ZV_REGENRECT(re,211,175,247,197) 'generate a rectangular area ZV_CONTGENSUBPIX(img,re,contSrc,40,50,35) 'generate sub-pixel contours ZV_CONTUNIONADJ(contSrc,contDst,1,70) 'neighbor contour connection ZV_GRAYTORGB(img,dst) 'convert grayscale image to color image ZV_CONTLIST(dst,contDst,ZV_COLOR(0,255,0),0) 'draw the contour

10.1.11. ZV_CONTCLOSE – Close Contour

Type	Contour
Description	Forcibly close the contour, even if the first and last points are different or the distance is greater than or equal to 1, the

	property still will be modified as closed.
Grammar	ZV_CONTCLOSE(cont,isClose) cont: ZVOBJECT type, input is also output, the property is closed, other properties remain unchanged isClose: 1-contour closed, 0-contour not closed
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM contCnt ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i=0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour ZV_CONTCLOSE(cont,1) 'set contour closure NEXT

10.1.12. ZV_CONTCLOSEEX – Close Contour

Type	Contour
Description	Closing the contour will increase the number of contour points by one so that the end point is the same as the first point.
Grammar	ZV_CONTCLOSEEX(src, dst) src: ZVOBJECT type, input dst: ZVOBJECT type, output, property is closed, others are the same
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM contCnt ZVOBJECT img, imgBw, contList, contSrc, contDst

	ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i=0 TO contCnt-1 ZV_LISTGET(contList,contSrc,i) 'get a certain contour ZV_CONTCLOSEEX(contSrc, contDst) 'close the contour and get a new closed contour NEXT

10.2. Access

10.2.1. ZV_CONTCOUNT – Contour Numbers

Type	Access
Description	It is used to get the number of contour points. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_CONTCOUNT (count, tabId) / count = ZV_CONTCOUNT (cont) cont: ZVOBJECT type, contour tabId: TABLE index, output parameter, the number of points
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM ptCnt, contCnt ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0)

	'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contList, cont,i) 'get a certain contour ptCnt = ZV_CONTCOUNT(cont) 'get the number of contour points ? ptCnt 'print the number of contour points NEXT

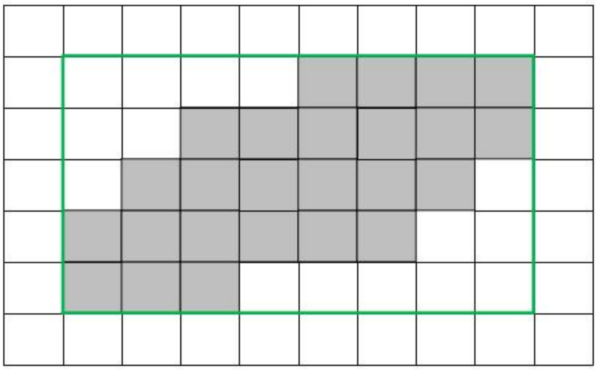
10.2.2. ZV_CONTGETPT – Contour Point Traversal

Type	Access
Description	It is used to get the coordinate of assigned point in contour.
Grammar	ZV_CONTGETPT (count, idx, tabId) cont: ZVOBJECT type, contour idx: index "idx" of assigned point, idx of the first point is 0 tabId: TABLE index, output parameter, obtained point's coordinate x, y
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM ptCnt, contCnt ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contCnt, cont,i) 'get a certain contour ptCnt = ZV_CONTCOUNT(cont) 'get the number of contour points

	FOR j = 0 TO ptCnt -1 ZV_CONTGETPT(cont,j,0) 'put the point coordinates in the contour into TABLE (0) ? *TABLE(0,2) 'print coordinates NEXT NEXT

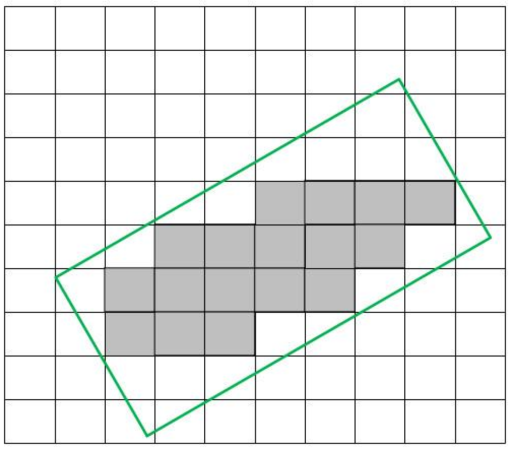
10.3. Geometric Analysis

10.3.1. ZV_CONTRECT – External Rectangle

Type	Geometric analysis
Description	External rectangle of contour that is parallel with coordinate axis. 
Grammar	ZV_CONTRECT(cont, tabId) cont: ZVOBJECT type, contour tabId: TABLE index, output parameter, external rectangle of contour, the output is in order of x, y, width, height, that is, the coordinate of the upper left corner of the rectangle, and width and height.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM contCnt ZVOBJECT img, imgBw, cont, contList

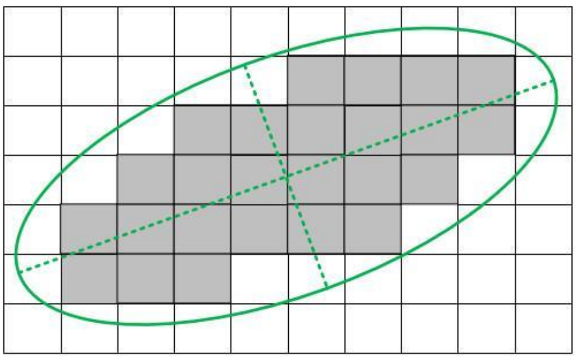
	ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour ZV_CONTRECT(cont,0) 'contour circumscribed rectangle, shape parameters are stored in TABLE(0) in turn ? *TABLE(0,4) 'print parameters NEXT

10.3.2. ZV_CONTRECT2 – Minimal External Rectangle

Type	Geometric analysis
Description	Enclosing matrix of the minimum area of the contour. 
Grammar	ZV_CONTRECT2(cont, tabId) cont: ZVOBJECT type, contour tabId: TABLE index, output parameter, the smallest enclosing rectangle of contour, the output is in order of cx, cy, width, height, angle, that is, the coordinate of the center the rectangle, the width and height of the rectangle, and the rotate

	angle of the rectangle.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM contCnt ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour ZV_CONTRECT2(cont,0) 'contour minimum enclosing rectangle, parameters are stored in TABLE(0) ?*TABLE(0,5) 'print parameters NEXT

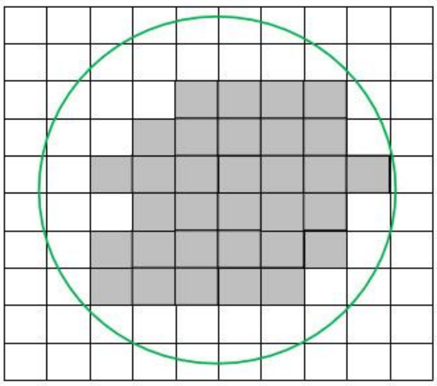
10.3.3. ZV_CONTELLIPAXIS – Feature Ellipse Parameters

Type	Geometric analysis
Description	Calculate feature ellipse parameters of contour. 
Grammar	ZV_CONTELLIPAXIS(cont, tabId) cont: ZVOBJECT type, contour tabId: TABLE index, output parameter, calculated feature

	ellipse parameters, they are majorLen, minorLen, angle in order, that is, the ellipse main axis's length, minor axis length, the angle of main axis and horizontal axis.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	DIM contCnt
	ZVOBJECT img, imgBw, cont, contList
	ZV_READIMAGE(img, "test.jpg",0)
	            'read the image in the original image format
	ZV_THRESH(img,imgBw,200,255) 'image binarization
	ZV_CONTGEN(imgBw,contList,1,0)
	            'save all the found contours into the contour list
	contCnt = ZV_LISTCOUNT(contList)
	            'get the number of contour lists
	FOR i = 0 TO contCnt-1
	      ZV_LISTGET(contList,cont,i) 'get a certain contour
	      ZV_CONTELLIPAXIS(cont,0)
	            'feature ellipse parameters are stored in TABLE(0)
	      ?*TABLE(0,3) 'print parameters
	NEXT

10.3.4. ZV_CONTCIRCLE – External Circle

Type	Geometric analysis
Description	Calculate contour's external circle. 

Grammar	ZV_CONTCIRCLE(cont, tabId) cont: ZVOBJECT type, contour tabId: TABLE index, output parameter, calculate external circle parameters, the output is in order of cx, cy, radius, that is, the coordinate x, y of the circle center, circle radius.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM contCnt ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour ZV_CONTCIRCLE(cont,0) 'external circle parameters are stored in TABLE(0) ?*TABLE(0,3) 'print parameters NEXT

10.4. Feature

10.4.1. ZV_CONTAREA – Area (Spare)

Type	Feature
Description	Calculate contour area. Online command function is supported, using parameters that don't need to pass in TABLE index.

Grammar	ZV_CONTAREA (cont, isOrient, tabId) / area = ZV_CONTAREA (cont, isOrient) cont: ZVOBJECT type, contour isOrient: whether to set the direction, the storage direction of contour point set that is viewed surface is the storage order. If the parameter is set to 0, the absolute value of the area will be output, and if it is 1, the signed area will be output. The area divided into positive and negative indicates the storage order of the contour point set. A positive area means that the contour point set is stored in a clockwise direction, and a negative area means that the contour point set is stored in a counterclockwise direction. tabId: TABLE index, output parameter, area of contour
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM area, contCnt ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour area = ZV_CONTAREA(cont,0) 'calculate contour's area ?*area 'print parameters NEXT

10.4.2. ZV_CONTLENGTH – Perimeter

Type	Feature
Description	Calculate contour length.

	Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_CONTLENGTH (cont, tabId) / len = ZV_CONTLENGTH (cont) cont: ZVOBJECT type, contour tabId: TABLE index, output parameter, length of contour
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM len, contCnt ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour len = ZV_CONTLENGTH (cont,0) 'calculate contour's perimeter ?*len 'print parameters NEXT

10.4.3. ZV_CONTCENTER – Center of Gravity

Type	Feature
Description	Calculate the center of gravity.
Grammar	ZV_CONTLCENTER (cont, tabId) cont: ZVOBJECT type, contour tabId: TABLE index, output parameter, calculate the coordinate x and y of center of gravity.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM contCnt

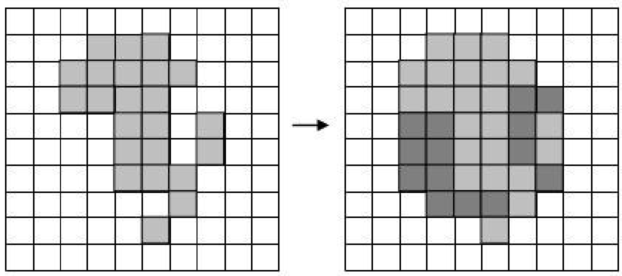
	ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour ZV_CONTCENTER (cont,0) 'calculate contour's center coordinates and save it into TABLE (0) ?TABLE(0) 'print parameters NEXT

10.4.4. ZV_CONTISCONVEX – Convex

Type	Feature
Description	Judge whether the contour is convex. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_CONTISCONVEX (cont, tabId) / convex = ZV_CONTISCONVEX (cont) cont: ZVOBJECT type, contour tabId: TABLE index, output parameter, whether the contour is convex.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM contCnt, convex ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization

	ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour convex = ZV_CONTISCONVEX (cont) 'calculate contour's convex ? convex 'print parameters NEXT

10.4.5. ZV_CONTCONVEXITY – Convexity

Type	Feature
Description	Calculate contour's convexity. Online command function is supported, using parameters that don't need to pass in TABLE index. The contour' area / the area of convex related to region That is: if Fc is the area of convex, Fo is the original area of the region, then, the convexity C will be: C = Fo / Fc. 
Grammar	ZV_CONTCONVEXITY (cont, tabId) / convex = ZV_CONTCONVEXITY (cont) cont: ZVOBJECT type, contour tabId: TABLE index, output parameter, calculate contour's convexity, [0,1], the bigger value, the convex the contour
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM contCnt, convex ZVOBJECT img, imgBw, cont, contList

	ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour convex = ZV_CONTCONVEXITY (cont) 'calculate contour's convexity ? convex 'print parameters NEXT

10.4.6. ZV_CONTCCLTY – Circularity

Type	Feature
Description	Calculate contour's circularity. Online command function is supported, using parameters that don't need to pass in TABLE index. The circularity indicates how similar is between contour and circle. Assume that F is the area of the region, max is the maximum distance from center to all contour pixels, then the circularity C is defined as: $C' = \frac{F}{(max^2 * \pi)}$
Grammar	ZV_CONTCCLITY (cont, tabId) / circular = ZV_CONTCCLTY (cont) cont: ZVOBJECT type, contour tabId: TABLE index, output parameter, calculate contour's circularity, the bigger value, the circular the contour
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM circular, contCnt ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0)

	'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour circular = ZV_CONTCCLTY (cont) 'calculate contour's circularity ? circular 'print parameters NEXT

10.4.7. ZV_CONTCMPTNS – Compactness

Type	Feature
Description	Calculate contour's compactness, C*C/(4*PI*S), S means contour area, C means contour length. Online command function is supported, using parameters that don't need to pass in TABLE index. Assume that L is the length of the contour, F is the area of the region, then the compactness C is defined as: $C' = \frac{L^2}{4F\pi}$
Grammar	ZV_CONTCMPTNS (cont, tabId) / compact = ZV_CONTCMPTNS (cont) cont: ZVOBJECT type, contour tabId: TABLE index, output parameter, calculate contour's compactness, [0,1], the bigger value, the compact the contour
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM compact, contCnt ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format

	ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour compact = ZV_CONTCMPTNS (cont) 'calculate contour's compactness ? compact 'print parameters NEXT

10.4.8. ZV_CONTRECTLY – Rectangularity

Type	Feature
Description	Calculate contour's rectangularity. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_CONTRECTLY (cont, tabId) / rectlity = ZV_CONTRECTLITY (cont) cont: ZVOBJECT type, contour tabId: TABLE index, output parameter, the rectangularity of the contour, [0,1], the bigger value, the rectangular the contour
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM rectlity, contCnt ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists

	FOR i = 0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour rectlity = ZV_CONTCMPTNS (cont) 'calculate contour's rectangularity ? rectlity 'print parameters NEXT

10.4.9. ZV_CONTHULLAREA – Hull Area

Type	Feature
Description	Calculate contour's hull area. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_CONTHULLAREA (cont, tabId) / area = ZV_CONTHULLAREA (cont) cont: ZVOBJECT type, contour tabId: TABLE index, output parameter, calculate the hull area.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM area, contCnt ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour area = ZV_CONTHULLAREA (cont) 'calculate contour's hull area ? area 'print parameters

	NEXT

10.4.10. ZV_CONTDIRECT – Contour Direction

Type	Feature
Description	Judge contour's direction, that is, from starting point to end point.
Grammar	ZV_CONTDIRECT (cont, tabId) cont: ZVOBJECT type, contour tabId: TABLE index, output parameter, contour's direction, -1: clockwise, 0: shared or uncountable, 1: anticlockwise, under image coordinate system.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM rectlity, contCnt ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour ZV_CONTDIRECT (cont,0) 'calculate contour's direction ? TABLE 'print parameters NEXT

10.4.11. ZV_CONTORIENT – Contour Orientation

Type	Feature

Description	Judge contour's orientation.
Grammar	ZV_CONTORIENT (cont, tabId) cont: ZVOBJECT type, contour tabId: TABLE index, output parameter, main axis's angle, clockwise is positive, [-180°, 180°)
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM angle, contCnt ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour ZV_CONTORIENT (cont,0) 'calculate contour's orientation ? TABLE 'print parameters NEXT

10.5. Transformation

10.5.1. ZV_CONTREVERSE – Contour Reverse

Type	Transformation
Description	Reverse contour's direction, that is, covert it from clockwise into anticlockwise.

Grammar	ZV_CONREVERSE (src, dst) src: ZVOBJECT type, input contour dis: ZVOBJECT type, output contour
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM contCnt ZVOBJECT img, imgBw, contList, contSrc, contDst ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list contCnt = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO contCnt-1 ZV_LISTGET(contList,cont,i) 'get a certain contour ZV_CONTREVERSE (conSrc, conDst) 'change contour's direction NEXT

10.5.2. ZV_CONTSORT – Sorting

Type	Feature
Description	Sorting contour list according to "feature".
Grammar	ZV_CONTSORT (list, feature, isInc) list: ZVOBJECT type, contour list to be sorted, list type feature: sorted feature type, please see below form isInc: whether is the ascending order, 0 means descending order, otherwise is the ascending order.

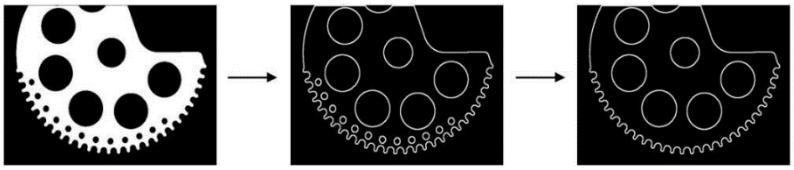
0	Area	17	External rectangle height
1	Gravity X	18	External rectangle x + w
2	Gravity Y	19	External rectangle y + h
3	Angle	20	External rectangle h / w
4	Perimeter, length	21	Min rectangle center X

5	Circularity	22	Min rectangle center Y
6	Compactness	23	Min rectangle width
7	Rectangularity	24	Min rectangle height
8	Convexity	25	Min rectangle angle
9	--	26	Min rectangle h / w
10	Equivalent elliptic main axis length	27	-
11	Equivalent elliptic slave axis length	28	-
12	Equivalent elliptic main axis angle	29	-
13	Equivalent elliptic main axis / slave axis	30	Convex
14	External rectangle x	31	Convex hull area
15	External rectangle y	32	Area with sign
16	External rectangle width		

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list ZV_CONTSORT (contList, 0, 0) 'sorting for contour's area feature ZV_LISTGET(contList,cont,i) 'get the first contour from contour list and save it into contour

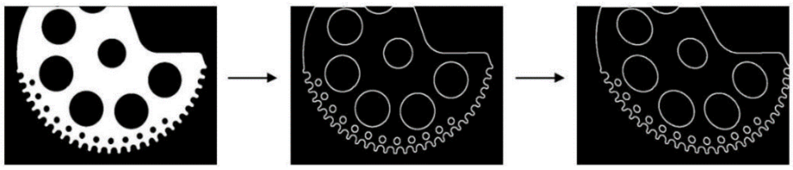
10.5.3. ZV_CONTFILTER -- Filter

Type	Feature
Description	Filter contours in contour list at one certain feature, and remain contours that meet feature.

Grammar	ZV_CONTFILTER (list, feature, min, max, isInvert) list: ZVOBJECT type, contour list that is to be filtered, list type feature: contour feature type. Refer to “sorting”. min: lower limit of feature value max: higher limit of feature value isInvert: whether to be inverse, if it is 1, contours that are not in the range will be retained, the default value is 0.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img, dst, imgBw, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list ZV_CONTFILTER (contList, 0, 500, 300000, 0) ‘filter contours in contour list, remain contours that are with area of 500-300000, that is, others are filtered. ZV_IMGCOPY ‘copy image ZV_IMGSETCONST (dst, 0) ‘constant fills in image ZV_CONTLIST (dst, contList, 255, 0) ‘draw the contour

10.5.4. ZV_CONTAFFINE – Contour / Contour List Affine Transformation

Type	Transformation
Description	Affine transform all points of contour of contour list.

Grammar	ZV_CONTAFFINE (src, matrix, dst) src: ZVOBJECT type, contour or contour list before transformation matrix: ZVOBJECT type, transform matrix, 2 rows and 3 columns or 3 rows and 3 columns dst: ZVOBJECT type, contour or contour list after transformation
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 DIM conCnt ZVOBJECT img, imgBw, dst, matAffine, contListSrc, contListDst ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list TABLE (0, 1, 0.2, 0, 0, 1, 0) 'save data into TABLE (10) ZV_MATGENDATA (matAffine, 2, 3, 0) 'transform matrix, in x direction ZV_CONTAFFINE (contListSrc, matAffine, contListDst) 'affine transform for contour or contour list ZV_IMGCOPY 'copy image ZV_IMGSETCONST (dst, 0) 'constant fills in image ZV_CONTLIST (dst, contList, 255, 0) 'draw the contour

Chapter XI Recognition

11.1. Data Code

Data code includes one dimensional code (bar code) and two-dimensional code.

11.1.1. ZV_CODEMASKBAR – Mask of Manufacture Bar Code Type

Type	Data code
Description	Generate the mask of bar type that is to be recognized, that is, bar type in mask format to be recognized.
Grammar	ZV_CODEMASKBAR(ean8,ean13,code39,code128,upca,upce,tabId) ean8: positive integer, non-zero means EAN-8 barcode type can be recognized ean13: positive integer, non-zero means EAN-13 barcode type can be recognized code39: positive integer, non-zero means CODE-39 barcode type can be recognized code128: positive integer, non-zero means CODE-128 barcode type can be recognized upca: positive integer, non-zero means UPC-A barcode type can be recognized upce: positive integer, non-zero means UPC-E barcode type can be recognized tabId: save generated mask
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_CODEMASKBAR (0, 1, 0, 0, 0, 0, 0) 'generate the mask that recognizes ean13 barcode type and save the mask into TABLE (0).

11.1.2. ZV_CODEREAD – Read Data Code

Type	Data code
Description	Read data code, including barcode and OR code recognition.
Grammar	ZV_CODEREAD(img, codeRst, type, step) Img: ZVOBJECT type, input single-channel image codeRst: ZVOBJECT type, list type, save all recognized data codes into list, and use ZV_LISTGET command to get one certain data code result. type: the type of reading data code, EAN-8, EAN-13, CODE-39, UPC-A, UPC-B, QR, DM, etc.: -- 0: automatic type, all types of barcodes except QR and DM codes can be recognized 1: EAN-8 type. EAN-8 barcode is a shortened version of EAN-13. It is mainly used on smaller products such as pens. The character set is [0-9], consisting of 8 digits. Digits 1-3 are country code (for example, China is 690-699), 4-7 digits are the product code, and the 8th digit is the check code to verify whether the decoding is correct and for anti-counterfeiting. 6 9 0 1 2 3 4 1 country code product code check code EAN-8 2: EAN-13 type, EAN-13 barcode is a globally accepted commodity barcode type. The character set is [0-9], consisting of 13 digits. 1-3 digits are country codes (for example, China is 690-699), 4- the 7th digit is the manufacturer code assigned by the country, the 8th to 12th digit is the product code assigned by the manufacturer, and the 13th digit is the check code to verify whether the decoding is correct and for anti-counterfeiting. EAN-13 is longer than EAN-8, so EAN-13 is more applicable. EAN-13 is compatible with UPC-A.

6 901234 567892

contry manufacture product check

EAN-13

3: CODE-39 type, CODE-39 code is also called Kudba 39 code. It is a variable-length barcode type that can encode data of any length. Its limitation is the product length and the recognition range of the barcode reader. Its character set consists of 44 characters, character set [0-9,cA-Z,-, empty cell, $, /, +, %, *, ;]. Among them, black is the bar and white is the space. One character of the CODE-39 code is composed of 9 units (5 bars and 4 spaces). There are 3 wide units and the rest are narrow units, so it is called CODE-39 code. CODE-39 codes are mainly used in business management, logistics tracking, postal services, medical and health care, industrial production lines, library and information and other fields.

123456

CODE-39

4: CODE-128 type, CODE-128 code is similar to CODE-39, it is also a variable-length barcode type that can encode data of any length. Its limitation also lies in the product length and the recognition range of the barcode reader, but it is a high-density encoding, within the same length area, CODE-128 encodes more data than CODE-39 and has richer data content. Its character set consists of 128 ASCII codes, so it is called CODE-128 code. Like CODE-39, CODE-128 is also mainly used in business management, logistics tracking, postal services, medical and health care, industrial production lines, library and information and other fields.

12345678

CODE-128

5: UPC-A type, UPC-A barcode is similar to EAN-13 and is also a general commodity barcode type, but it is mainly used in the United States and Canada. The character set is [0-9], consisting of 12 digits, and the first digit is the system code, 2-6 digits are the manufacturer code, 7-11 digits are the product code, and the 12th digit is the check code to verify whether the decoding is correct and for anti-counterfeiting. UPC-A is longer than UPC-E, so UPC -A is more applicable, UPC-A is compatible with EAN-13.

0 01234 56789 5

system manufacture product check

UPC-A

6: UPC-E type. UPC-E is a shortened version of UPC-A. It is mainly used for small products with a smaller area. The character set is [0-9] and consists of 8 digits. The first digit is the system code, and the 2-7 digit are the product code, the 8th digit is the check code to verify whether the decoding is correct and for anti-counterfeiting.

0 0 1 2 3 4 5 7

system product check

UPC-E

20: QR type, the "QR" of QR code is the abbreviation of Quick Response, which means that this QR code can be read quickly.

	It can store rich information including text, URL address and other types of data. It is usually used on product packaging. http://www.zmotion.com.cn/ 21: DM type, the "DM" of DM code is the abbreviation of Data Matrix, which is usually used on product packaging. http://www.zmotion.com.cn/ -- step: scanning step size, a positive integer. The larger the step size, the faster it will be but it will affect the recognition accuracy. It is usually 4.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	0 123456 789012 DIM rstStr(64) ZVOBJECT img, codeList, codeRst ZV_READIMAGE(img,"test.png",0) 'read the image in the original image format ZV_CODEREAD(img,codeList,2,4) 'recognize the EAN-13 barcode and store the result in the list

	ZV_LISTGET(codeList,codeRst,0) 'get the first result in the list ZV_CODESTR(codeRst,64,0) 'get the result and store it in TABLE(0) DMCPY rstStr(0),TABLE(0),64 'copy the array of TABLE to rstStr ? rstStr 'print recognition result: 0123456789012

11.1.3. ZV_CODESTR – Get Data Code Result

Type	Data code
Description	Get data code result, and output in character string method.
Grammar	ZV_CODESTR (code, maxLen, tabId) code: data code read by ZV_CODEREAD, ZVOBJECT type maxLen: the maximum acceptable length of the data code string result tabId: TABLE index, output parameter, starting position of obtained data code result in character string
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM rstStr(64) ZVOBJECT img, codeList, codeRst ZV_READIMAGE(img,"test.png",0) 'read the image in the original image format ZV_CODEREAD(img,codeList,2,4) 'recognize the EAN-13 barcode and store the result in the list ZV_LISTGET(codeList,codeRst,0) 'get the first result in the list ZV_CODESTR(codeRst,64,0) 'get the result and store it in TABLE(0) DMCPY rstStr(0),TABLE(0),64 'copy the array of TABLE to rstStr ? rstStr 'print recognition result: 0123456789012

11.1.4. ZV_CODESTR – Get Data Code Type

Type	Data code

Description	Get data code type, and output in value method, such as, EAN-13 barcode type, output the value 2.
Grammar	ZV_CODETYPE(code, tabId) code: data code read by ZV_CODEREAD, ZVOBJECT type tabId: TABLE index, output parameter
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, codeList, codeRst ZV_READIMAGE(img,"test.png",0) 'read the image in the original image format ZV_CODEREAD(img,codeList,2,4) 'recognize the EAN-13 barcode and store the result in the list ZV_LISTGET(codeList,codeRst,0) 'get the first result in the list ZV_CODETYPE(codeRst, 0) 'get the result and store it in TABLE(0) ?TABLE (0) 'print data code type

11.1.5. ZV_CODETYPESTR – Get Data Code Type

Type	Data code
Description	Get data code type, output in character string, such as, EAN-13 barcode type, output "EAN-13" in character string.
Grammar	ZV_CODETYPESTR(code, maxLen, tabId) code: data code read by ZV_CODEREAD, ZVOBJECT type maxLen: the maximum acceptable length of the data code string result tabId: TABLE index, starting position of obtained data code result in character string
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM rstStr(64) ZVOBJECT img, codeList, codeRst ZV_READIMAGE(img,"test.png",0) 'read the image in the original image format

	ZV_CODEREAD(img,codeList,2,4) 'recognize the EAN-13 barcode and store the result in the list ZV_LISTGET(codeList,codeRst,0) 'get the first result in the list ZV_CODETYPESTR(codeRst, 64, 0) 'get the result and store it in TABLE(0) DMCPY rstStr(0),TABLE(0),64 'copy the array of TABLE to rstStr ? rstStr 'print data code type

11.1.6. ZV_CODEPOS – Get Data Code Position

Type	Data code
Description	Get data code position.
Grammar	ZV_CODEPOS(code, tabId) code: data code read by ZV_CODEREAD, ZVOBJECT type, please note this command only can read QR position. tabId: TABLE starting index that saves data code position, they are the coordinates of the upper left, upper right, lower right, and lower left vertices of the rectangle are in order.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT img, codeList, codeRst
	ZV_READIMAGE(img,"test.png",0)
	            'read the image in the original image format
	ZV_CODEREAD(img,codeList,2,4)
	'recognize the EAN-13 barcode and store the result in the list
	ZV_LISTGET(codeList,codeRst,0) 'get the first result in the list
	ZV_CODETYPESTR(codeRst, 0)
	            'get the result and store it in TABLE(0)
	? *TABLE (0, 4) 'print data code type

11.2. OCR

OCR (Optical Character Recognition) is used to recognize characters. First, it needs to segment a single character from the image, and then trains it to learn the features. Finally, the newly segmented character is used to find the character with the highest similarity in the training library based on its features as the recognition object.

11.2.1. ZV_OCRSEGSETPARAM – Set Segment Parameters

<table>
<tr><td>Type</td><td>OCR</td></tr>
<tr><td>Description</td><td>Set character segment parameters.</td></tr>
<tr><td>Grammar</td><td>ZV_OCRSEGSETPARAM (param, threshMode, thresh, minArea, maxArea, minWidth, maxWidth, minHeight, maxHeight, polor, morpType, stWidth, stHeight, minSpace)
param: segment parameter, ZVOBJECT type, output
threshMode: threshold mode, 0 – manual threshold, 1 – automatic threshold, 2 – adaptive threshold
thresh: threshold, this relates to threshold mode:
<table>
<tr><td>threshMode</td><td>thresh</td></tr>
<tr><td>0</td><td>Low threshold segmented by image binarization, at this time, high threshold defaults to 255.</td></tr>
<tr><td>1</td><td>Parameters are invalid.</td></tr>
<tr><td>2</td><td>The size of block of adaptive threshold, positive odd number, that is, in pixel area, calculate area size of threshold.</td></tr>
</table>
minArea: minimum area of characters, non-negative
maxArea: maximum area of characters, non-negative
minWidth: minimum character width, non-negative number
maxWidth: maximum character width, non-negative
minHeight: minimum height of characters, non-negative
maxHeight: maximum height of characters, non-negative
polar: character polarity, 0-black text on white background,</td></tr>
</table>

	1-white text on black background morphType: morphological type, 0-open operation, 1-closed operation stWidth: structure element width, non-negative number stHeight: Structure element height, non-negative number minSpace: the minimum spacing between characters. Two characters smaller than this spacing are considered to be the part of the same character. A negative value is invalid, that is, the spacing parameter does not work when dividing characters.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT param ZV_OCRSEGSETPARAM (param, 0, 120, 20, 30000, 3, 500, 3,500, 0, 0, 3, 3, -1) 'set character's segment parameters

11.2.2. ZV_OCRSEGCHAR – Character Segment

Type	OCR
Description	To do character segmentation for image in ROI, and save all segmented characters into character sample.
Grammar	ZV_OCRSEGCHAR(img, param, sample, cx, cy, width, height, angle) img: single-channel image param: ZVOBJECT type, segment parameters, generate by ZV_OCRSEGSETPARAM sample: ZVOBJECT type, obtained segmented character sample library, there are many samples in sample library, that is, character information, such as, character image, character name, etc., and segmented character name are "?". cx: roi center x coordinate cy: roi center y coordinate width: roi width height: roi height angle: roi angle

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, param, sample ZV_READIMAGE (img, “test.png”, 0) 'read image in the original format ZV_OCRSEGSETPARAM (param, 0, 120, 20, 30000, 3, 500, 3,500, 0, 0, 3, 3, -1) 'set character's segment parameters ZV_OCRSEGCHAR (img, param, sample, 320, 340, 120, 80, 0) 'character segmentation

11.2.3. ZV_OCRSAMPLEAPP – Generate Training Sample

Type	OCR
Description	Use sample library and sample name set to generate sample library that is used to train features.
Grammar	ZV_OCRSAMPLEAPP (sample, trainSample, sampleName) sample: sample library, ZVOBJECT type trainSample: train sample library, ZVOJECT type, output parameter, if trainSample has been generated, then add new sample sampleName: character string, input parameter, each sample name is separated by space, and the number of sample name must be consistent with the number of input samples. When sample names are single-character, space can be omitted, but at this time, character string length should be equal to sample numbers, otherwise fail to train samples.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, param, sample, trainSample ZV_READIMAGE (img, “train.png”, 0) 'read image in the original format ZV_OCRSEGSETPARAM (param, 0, 120, 20, 30000, 3, 500, 3,500, 0, 0, 3, 3, -1) 'set character's segment parameters ZV_OCRSEGCHAR (img, param, sample, 320, 340, 120, 80, 0)

	'character segmentation ZV_OCRSAMPLEAPP (sample, trainSample, "A B C D E") 'generate training sample, there are 5 samples in "sample".

11.2.4. ZV_OCRCREATESVM – Create SVM Classifier

Type	OCR
Description	Create OCR classifier that supports vector machine (SVM).
Grammar	ZV_OCRCREATESVM (ocr) ocr: ocr classifier, ZVOBJECT type, output parameter, used to classify characters.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT ocr ZV_OCRCREATESVM (ocr) 'create OCR classification

11.2.5. ZV_OCRTRAINSVM – Train SVM Classifier

Type	OCR
Description	Use sample library to train SVM classifier, and the sample library must be generated by ZV_OCRSAMPLEAPP command.
Grammar	ZV_OCRTRAINSVM (sample, ocr [, eps = 0.001]) sample: sample library, ZVOBJECT type, input parameter ocr: ocr classifier, ZVOBJECT type, output parameter, used to classify characters eps: training precision, when precision is reached, training ends. When it is more than 0, the default value is 0.001, when it is 0, the value 0.001 is used. The smaller the precision, the longer the training.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, param, sample, trainSample, ocr

	ZV_READIMAGE (img, “train.png”, 0) ‘read image in the original format ZV_OCRSEGSETPARAM (param, 0, 120, 20, 30000, 3, 500, 3,500, 0, 0, 3, 3, -1) ‘set character's segment parameters ZV_OCRSEGCHAR (img, param, sample, 320, 340, 120, 80, 0) ‘character segmentation ZV_OCRSAMPLEAPP (sample, trainSample, “A B C D E”) ‘generate training sample, there are 5 samples in “sample”. ZV_OCRCREATESVM (ocr) ‘create OCR classification ZV_OCRTRAINSVM (trainSample, ocr, 0) ‘train classifier that supports SVM

11.2.6. ZV_OCRCLASSIFYSVM – SVM Classification Recognition

Type	OCR
Description	Use SVM classifier to classify and recognize characters in sample library, and then output results. Each sample's names as the recognized result are saved into charlist.
Grammar	ZV_OCRCLASSIFYSVM(ocr,sample,maxLen,tabId) ocr: classifier, ZVOBJECT type, input sample: character sample library, ZVOBJECT type, input maxLen: the maximum TABLE space length that can be used to store the recognition result tabId tabId: TABLE starting index where recognition results are stored, output parameters
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, param, sample, trainSample, ocr ZV_READIMAGE (img, “sample.png”, 0) ‘read image in the original format ZV_OCRSEGSETPARAM (param, 0, 120, 20, 30000, 3, 500, 3,500, 0, 0, 3, 3, -1) ‘set character's segment parameters

	ZV_OCRSEGCHAR (test_img, param, sample, 320, 340, 120, 80, 0) 'character segmentation ZV_OCRCREATESVM (ocr) 'create OCR classification ZV_OCRCLASSIFYSVM (ocr, sample, 32, 0) 'recognize characters in sample library and save them into TABLE (0)

11.2.7. ZV_OCRCREATEMLP – Create MLP Classifier

Type	OCR
Description	Create Neural Network (MLP) OCR classifier.
Grammar	ZV_OCRCREATEMLP (ocr, neuNum) ocr: ocr classifier, ZVOBJECT type, output parameters, used to classify characters neuNum: the number of hidden layer neurons, >=3. In most applications, a small setting will provide better classification results. If the setting is too large, the MLP classifier may overfit the training data and then may be with poor generalization ability. For example, the classification effect on data that has been trained is very good, but the classification effect on unknown data is relatively poor.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT ocr ZV_OCRCREATEMLP (ocr, 3) 'create a MLP classifier that has 3 neurons

11.2.8. ZV_OCRTRAINMLP – Train MLP Classifier

Type	OCR
Description	Use sample library to train MLP classifier, and the sample library must be generated by ZV_OCRSAMPLEAPP command.

Grammar	ZV_OCRTRAINMLP (sample, ocr [, eps = 0.001]) sample: sample library, ZVOBJECT type, input parameter ocr: ocr classifier, ZVOBJECT type, output parameter, used to classify characters eps: training precision, when precision is reached, training ends. When it is more than 0, the default value is 0.001, when it is 0, the value 0.001 is used. The smaller the precision, the longer the training.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, param, sample, trainSample, ocr ZV_READIMAGE (img, “train.png”, 0) ‘read image in the original format ZV_OCRSEGSETPARAM (param, 0, 120, 20, 30000, 3, 500, 3,500, 0, 0, 3, 3, -1) ‘set character’s segment parameters ZV_OCRSEGCHAR (img, param, sample, 320, 340, 120, 80, 0) ‘character segmentation ZV_OCRSAMPLEAPP (sample, trainSample, “A B C D E”) ‘generate training sample, there are 5 samples in “sample”. ZV_OCRCREATEMLP (ocr, 3) ‘create the MLP classifier that has 3 neurons ZV_OCRTRAINMLP (trainSample, ocr, 0) ‘train MLP classifier

11.2.9. ZV_OCRCLASSIFYMLP – MLP Classification Recognition

Type	OCR
Description	Use MLP classifier to classify and recognize characters in sample library, and then output results. Each sample’s names as the recognized result are saved into charlist.

Grammar	ZV_OCRCLASSIFYMLP (ocr, sample, score, maxLen, tabId) ocr: classifier, ZVOBJECT type, input sample: character sample library, ZVOBJECT type, input score: recognize score, if the score is not met, "?" will be output, the range is [0, 100]. maxLen: the maximum TABLE space length that can be used to store the recognition result tabId tabId: TABLE starting index where recognition results are stored, output parameters
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, param, sample, trainSample, ocr ZV_READIMAGE (img, "sample.png", 0) 'read image in the original format ZV_OCRSEGSETPARAM (param, 0, 120, 20, 30000, 3, 500, 3,500, 0, 0, 3, 3, -1) 'set character's segment parameters ZV_OCRSEGCHAR (img, param, sample, 320, 340, 120, 80, 0) 'character segmentation ZV_OCRCREATEMLP (ocr, 3) 'create the MLP classifier that has 3 neurons ZV_OCRCLASSIFYMLP (ocr, sample, 90, 32, 0) 'recognize characters in sample library and save them into TABLE (0)

11.2.10. ZV_OCRSAMPLEDEL – Delete Sample

Type	OCR
Description	Delete one certain sample from sample library according to sample name.
Grammar	ZV_OCRSAMPLEDEL (sample, sampleName) sample: sample library, ZVOBJECT type sampleName: character string, sample name
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT img, param, sample, trainSample ZV_READIMAGE (img, “train.png”, 0) ‘read image in the original format ZV_OCRSEGSETPARAM (param, 0, 120, 20, 30000, 3, 500, 3,500, 0, 0, 3, 3, -1) ‘set character’s segment parameters ZV_OCRSEGCHAR (img, param, sample, 320, 340, 120, 80, 0) ‘character segmentation ZV_OCRSAMPLEAPP (sample, trainSample, “A B C D E”) ‘generate training sample, there are 5 samples in “sample”. ZV_OCRSAMPLEDEL (sample, “A”) ‘delete sample’s name A from sample library

11.2.11. ZV_OCRSAMPLECOUNT – Get Sample Numbers

Type	OCR
Description	Get the number of samples in sample library.
Grammar	ZV_OCRSAMPLECOUNT (sample, tab_num) sample: sample library, ZVOBJECT type sample_num: TABLE index that saves the number of samples
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, param, sample, trainSample ZV_READIMAGE (img, “train.png”, 0) ‘read image in the original format ZV_OCRSEGSETPARAM (param, 0, 120, 20, 30000, 3, 500, 3,500, 0, 0, 3, 3, -1) ‘set character’s segment parameters ZV_OCRSEGCHAR (img, param, sample, 320, 340, 120, 80, 0) ‘character segmentation ZV_OCRSAMPLEAPP (sample, trainSample, “A B C D E”) ‘generate training sample, there are 5 samples in “sample”. ZV_OCRSAMPLECOUNT (trainSample, 0)

	'get the number of samples and save it into TABLE (0)

11.2.12. ZV_OCRSAMPLEIMG – Get Sample Image

Type	OCR
Description	Get sample image of specified position in sample library.
Grammar	ZV_OCRSAMPLEIMG (sample, img, id) sample: sample library, ZVOBJECT type img: sample image, ZVOBJECT type, output id: id No., used to specify sample at assigned position in sample library
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, dst, sample, sample ZV_READIMAGE (img, "train.png", 0) 'read image in the original format ZV_OCRSEGSETPARAM (param, 0, 120, 20, 30000, 3, 500, 3,500, 0, 0, 3, 3, -1) 'set character's segment parameters ZV_OCRSEGCHAR (img, param, sample, 320, 340, 120, 80, 0) 'character segmentation ZV_OCRSAMPLEIMG (sample, dst, 0) 'get the sample image that is in the first position in sample library

11.2.13. ZV_OCRSAMPLENAME – Get Sample Name

Type	OCR
Description	Get sample name of specified position in sample library.
Grammar	ZV_OCRSAMPLENAME (sample, id, maxLen, tabId) sample: sample library, ZVOBJECT type id: id No., used to specify sample at assigned position in sample library maxLen: the maximum TABLE space length that can be

	used to store the recognition result tabId tabId: TABLE starting index where recognition results are stored, output parameters
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, param, sample, trainSample ZV_READIMAGE (img, "train.png", 0) 'read image in the original format ZV_OCRSEGSETPARAM (param, 0, 120, 20, 30000, 3, 500, 3,500, 0, 0, 3, 3, -1) 'set character's segment parameters ZV_OCRSEGCHAR (img, param, sample, 320, 340, 120, 80, 0) 'character segmentation ZV_OCRSAMPLEAPP (sample, trainSample, "A B C D E") 'generate training sample, there are 5 samples in "sample". ZV_OCRSAMPLENAME (trainSample, 0, 10, 0) 'get the sample image that is in the first position in sample library, and save its name into TABLE (0)

11.2.14. ZV_OCRCLASSCOUNT – Get Classification Numbers

Type	OCR
Description	Get the number of classification.
Grammar	ZV_OCRCLASSCOUNT (ocr, tabId) ocr: ocr classifier, ZVOBJECT type tabId: TABLE index that saves classification numbers
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, param, sample, ocr ZV_READIMAGE (img, "sample.png", 0) 'read image in the original format ZV_OCRSEGSETPARAM (param, 0, 120, 20, 30000, 3, 500, 3,500,

	0, 0, 3, 3, -1) 'set character's segment parameters ZV_OCRSEGCHAR (img, param, sample, 320, 340, 120, 80, 0) 'character segmentation ZV_SCRCREATESVM (ocr) 'create OCR classification ZV_OCRCLASSIFYSVM (ocr, sample, 32, 0) 'recognize characters in sample library, and save results into TABLE (0) ZV_OCRCLASSCOUNT (ocr, 0) 'get the number of classified types and save it into TABLE (0)

11.2.15. ZV_OCRCLASSTONAME – Get Class Name of Specified No.

Type	OCR
Description	Get the name of classified type in assigned id No. of classifier.
Grammar	ZV_OCRCLASSTONAME (ocr, id, maxLen, tabId) ocr: ocr classifier, ZVOBJECT type id: classification id No. in classifier, > 0, < the number of total classified types maxLen: the maximum TABLE space length that can be used to store the recognition result tabId tabId: TABLE starting index where recognition results are stored, output parameters
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, param, sample, ocr ZV_READIMAGE (img, "sample.png", 0) 'read image in the original format ZV_OCRSEGSETPARAM (param, 0, 120, 20, 30000, 3, 500, 3,500, 0, 0, 3, 3, -1) 'set character's segment parameters ZV_OCRSEGCHAR (img, param, sample, 320, 340, 120, 80, 0) 'character segmentation ZV_SCRCREATESVM (ocr) 'create OCR classification

	ZV_OCRCLASSIFYSVM (ocr, sample, 32, 0) ‘recognize characters in sample library, and save results into TABLE (0) ZV_OCRCLASSTONAME (ocr, 0, 10, 0) ‘get the first one classified type’s name in ocr classifier, and save it into TABLE (0)

11.2.16. ZV_OCRCLASSTOID – Get No. of Classified Name

Type	OCR
Description	Get the id No. of classified type's name in ocr classifier.
Grammar	ZV_OCRCLASSTOID (ocr, name, tabId) ocr: ocr classifier, ZVOBJECT type, input parameter name: classification name, character string tabId: TABLE index that saves id, output parameter
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, param, sample, ocr ZV_READIMAGE (img, “sample.png”, 0) ‘read image in the original format ZV_OCRSEGSETPARAM (param, 0, 120, 20, 30000, 3, 500, 3,500, 0, 0, 3, 3, -1) ‘set character’s segment parameters ZV_OCRSEGCHAR (img, param, sample, 320, 340, 120, 80, 0) ‘character segmentation ZV_SCRCREATESVM (ocr) ‘create OCR classification ZV_OCRCLASSIFYSVM (ocr, sample, 32, 0) ‘recognize characters in sample library, and save results into TABLE (0) ZV_OCRCLASSTOID (ocr, “A”, 0) ‘get the sequence number of the category named A in the ocr recognizer and store it in TABLE(0)

11.2.17. ZV_OCRSAMPLERECT2 – Get Sample Rectangle

Type	OCR
Description	Get the bounding rectangle of sample with specified id No. in sample library.
Grammar	ZV_OCRSAMPLERECT2 (sample, id, tabId) sample: sample library, ZVOBJECT type id: the sample ID serial number in the sample library, > 0, < the total number of samples tabId: TABLE starting index that stores sample bounding moment parameters, output parameters, the output order is cx, cy, width, height, angle
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_OCRSAMPLERECT2 (sample, 0, 0) 'get the 0th sample surrounding moment in the sample library and store it in the TABLE (0)

Chapter XII List

List, as a type of visual variable ZVOBJECT, its type is 5 (it can be viewed by the ZV_TYPE command). It is a data structure that can store other visual variables.

Lists are divided into special lists and ordinary lists, and the use of each list has certain restrictions.

- **Special List:**

Special lists include contour lists, area lists, etc., and support operations such as sorting, filtering, size, and element acquisition of the list, but do not support editing of the list (i.e., modifying the size of the list itself, including inserting and deleting the list, etc.). Elements obtained from special list (ZV_LISTGET) cannot be inserted into the general list. If an insertion operation is required, copying or other operations that can create new variables are required, otherwise an error will be reported.

- **General list:**

General lists support operations such as insertion, deletion, list size, and element acquisition. They cannot perform sorting, filtering, and resetting (deprecated) operations for special lists. Elements can be inserted directly into variables. If the original variable is not a list, it will be released and reconstructed into an ordinary list and the insertion operation will be performed. If the original variable is an ordinary list, it will be inserted directly.

The elements obtained by general lists and special lists are references to elements in the list, so modifying the obtained elements will also modify the elements in the list synchronously, such as: ZV_LISTGET(lost, elem, 0), ZV_CLEAR(elem), then the data of variable elem will be cleared and the zero element of the list will also be cleared. However, when it is reconstructed as an output variable, the operation will not be passed to the list. For example, if ZV_IMGCOPY (img, elem) is executed above, the variable elem will be changed to a copy of the image img, but the list elements will not be changed.

Note: inserting an element into a special list will also cause the list to be freed and restructured into a general list, with the number of elements after insertion being 1.

12.1. Access

12.1.1. ZV_LISTCOUNT – Element Numbers

Type	Access
Description	Get the number of elements in list. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_LISTCOUNT (list, tabId) / count = ZV_LISTCOUNT (list) list: ZVOBJECT type, list tabId: TABLE index, output parameter, the number of elements
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM count ZVOBJECT list count = ZV_LISTCOUNT(list) 'get the number of elements in list

12.1.2. ZV_LISTCOUNT – Element Numbers

Type	Access
Description	Get the element of specified id No. in list, the element belongs to zvobject type. Note: what is obtained is the reference of the element. Modifying the obtained element will also modify the elements in the list, but the variable of the obtained element is used as an output parameter to dereference it.
Grammar	ZV_LISTGET (list, obj, id) list: ZVOBJECT type, list obj: ZVOJECT type, obtained element object id: id of specified element, starting from 0
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZVOBJECT list, obj ZV_LISTGET (list, obj, 0) 'get element of No.0 in list

12.2. Insert & Delete

12.2.1. ZV_LISTINSERT – Insert Element

Type	Insert & delete
Description	Insert the element into list.
Grammar	ZV_LISTINSERT (list, obj, pos) list: ZVOBJECT type, list obj: ZVOJECT type, element to be inserted pos: position where element-inserting is, default value is -1, which means inserting element at the end of list
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT list, obj ZV_LISTINSERT (obj, list, -1) 'insert one element at the end of the list

12.2.2. ZV_LISTDELETE – Delete Element

Type	Insert & delete
Description	Delete elements at specified position in list.
Grammar	ZV_LISTDELETE (list, pos) list: ZVOBJECT type, list pos: position where the element is deleted, default value is -1, which means that deleting element at the end of list
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT list

	ZV_LISTDELETE (list, -1) ‘delete one element at the end of the list

12.2.3. ZV_LISTEXTEND – Extend Element

Type	Insert & delete
Description	Expand and combine lists list1 and list2 into a list and output the combined list as list2. The extended combination method is to put the elements in list1 into list2 in sequence, and reset list2.
Grammar	ZV_LISTEXTEND (list1,list2) list1: ZVOBJECT type, list, input parameters list2: ZVOBJECT type, list, both input and output parameters
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT list1, list2 ZV_LISTEXTEND(list1,list2) 'expand and combine lists list1 and list2, and output the combined list as list2

12.2.4. ZV_LISTREPLACE – Replace Element

Type	Insert & delete
Description	Replace element at specified position in list.
Grammar	ZV_LISTREPLACE (elem, list[,idx=-1]) elem: ZVOBJECT type, element to be replaced list: ZVOBJECT type, list, both input and output parameters idx: the position of the element to be replaced, the default -1 means replacing the element at the end of list
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT elem, list ZV_LISTREPLACE(elem,list,-1) 'replace the tail element of the list with elem

12.2.5. ZV_LISTSLICE – Slice Element

Type	Insert & delete
Description	Get sub list in list.
Grammar	ZV_LISTSLICE(list,seq,stidx[,num=-1]) list: ZVOBJECT type, list, input list seq: ZVOBJECT type, list, sub-list stidx: the starting position of the sub-list in the list num: the number of sub-lists, default -1, indicating the number from the starting position to the end position
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT list,seq ZV_LISTSLICE(list,seq,5,-1) 'get a sub-lis t seq from position 5 to the end of the list

Chapter XIII Tool

13.1. Geometry

13.1.1. ZV_DISTPP – Distance of Point and Point

Type	Geometry
Description	Calculate distance between two points.
Grammar	dist = ZV_DISTPP (x1, y1, x2, y2) x1: coordinate x of the first one point y1: coordinate y of the first one point x2: coordinate x of the second one point y2: coordinate y of the second one point returned value: dist: distance between two points
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM dist dist = ZV_DISTPP (100, 100, 200, 200) 'returned value is the distance between two points

13.1.2. ZV_DISTPL – Distance of Point and Line

Type	Geometry
Description	Calculate distance between point and line.
Grammar	dist = ZV_DISTPL (px, py, 1x1, 1y1, 1x2, 1y2) px: coordinate x of the point py: coordinate y of the point 1x1: coordinate x of the first one point of the line 1y1: coordinate y of the first one point of the line 1x2: coordinate x of the second one point of the line 1y2: coordinate y of the second one point of the line returned value:

	dist: distance from the point to line
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM dist dist = ZV_DISTPL (10, 10, 100, 100, 200, 200) 'returned value is the distance from point to line

13.1.3. ZV_DISTPS – Distance of Point and Segment

Type	Geometry
Description	Calculate distance between point and segment.
Grammar	dist = ZV_DISTPS (px, py, 1x1, 1y1, 1x2, 1y2) px: coordinate x of the point py: coordinate y of the point 1x1: coordinate x of the first one point of the segment 1y1: coordinate y of the first one point of the segment 1x2: coordinate x of the second one point of the segment 1y2: coordinate y of the second one point of the segment returned value: dist: distance from the point to segment
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM dist dist = ZV_DISTPS (10, 10, 100, 100, 200, 200) 'returned value is the distance from point to segment

13.1.4. ZV_DISTSL – Distance of Segment and Line

Type	Geometry
Description	Calculate minimal and maximal distance between segment and line.
Grammar	ZV_DISTSL(lsx1,lsy1,lsx2,lsy2,lx1,ly1,lx2,ly2,tab_dist) lsx1: coordinate x of segment point 1

	lsy1: coordinate y of segment point 1 lsx2: coordinate x of segment point 2 lsy2: coordinate y of segment point 2 lx1: coordinate x of line point 1 ly1: coordinate y of line point 1 lx2: coordinate x of line point 2 ly2: coordinate y of line point 2 tab_dist: TABLE index, output parameters, minimal distance and maximal distance in order
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_DISTSL(10, 10, 20, 20, 0, 0, 30, 0, 0) 'calculate minimal and maximal distance from segment to line and save them into TABLE (0).

13.1.5. ZV_DISTSS – Distance of Segment and Segment

Type	Geometry
Description	Calculate minimal and maximal distance between two segments.
Grammar	ZV_DISTSS(ls1x1,ls1y1,ls1x2,ls1y2,ls2x1,ls2y1,ls2x2,ls2y2,tab_dist) ls1x1: coordinate x of segment 1 point 1 ls1y1: coordinate y of segment 1 point 1 ls1x2: coordinate x of segment 1 point 2 ls1y2: coordinate y of segment 1 point 2 ls2x1: coordinate x of segment 2 point 1 ls2y1: coordinate y of segment 2 point 1 ls2x2: coordinate x of segment 2 point 2 ls2y2: coordinate y of segment 2 point 2 tab_dist: TABLE index, output parameters, minimal distance and maximal distance in order
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	ZV_DISTSS(10, 10, 20, 20, 0, 0, 30, 0, 0) 'calculate minimal and maximal distance from segment 1 to segment 2 and save them into TABLE (0).

13.1.6. ZV_DISTCONTP – Min Distance of Point and Contour

Type	Geometry
Description	Calculate the minimal distance from point to contour, that is, the distance of point and the closest point of contour. Online command function is supported, using parameters that don't need to pass in TABLE index.
Grammar	ZV_DISTCONTP (cont, px, py, tabId) / number = ZV_DISTCONTP (cont, px, py) cont: ZVOBJECT type, contour px: coordinate x of point py: coordinate y of point tabId: TABLE index, output parameter, distance from point to contour, negative value means the point is outside the contour, then positive means the point is inside the contour, 0 means point is on the contour.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list ZV_LISTGET(contList,cont,0) 'get the first contour ZV_DISTCONTP(cont,10,10,0) 'put the distance "cont" from the point to the contour in TABLE(0).

13.1.7. ZV_DISTCONTPEX – Min Distance of Point and Contour

Type	Geometry
Description	Calculate the minimal distance from point to contour, that is, the distance of point and the closest point of contour, output distance and corresponding contour point.
Grammar	ZV_DISTCONTPEX (cont, type, px, py, tabId) cont: ZVOBJECT type, contour type: distance type, 0: distance to contour node, 1: distance to contour line px: coordinate x of point py: coordinate y of point tabId: TABLE index, output parameter. minDist, x, y, are output in order, that is, min distance, corresponding contour coordinates.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list ZV_LISTGET(contList,cont,0) 'get the first contour ZV_DISTCONTPEX(cont,0,10,10,0) 'calculate minimal distance from point (10,10) to contour, and output and save distance and corresponding coordinates into TABLE (0)

13.1.8. ZV_DISTCONT – Min Distance of Two Contours

Type	Geometry

Description	Calculate the minimal distance of two contours, and output two contours' corresponding points when in minimal distance.
Grammar	ZV_DISTCONTPEX (cont, type, px, py, tabId) cont: ZVOBJECT type, contour type: distance type, 0: distance to contour node, 1: distance to contour line px: coordinate x of point py: coordinate y of point tabId: TABLE index, output parameter. minDist, x, y, are output in order, that is, min distance, corresponding contour coordinates.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, imgBw, cont, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list ZV_LISTGET(contList,cont,0) 'get the first contour ZV_DISTCONTPEX(cont,0,10,10,0) 'calculate minimal distance from point (10,10) to contour, and output and save distance and corresponding coordinates into TABLE (0)

13.1.9. ZV_INTERSECTLL – Straight Line Intersection

Type	Geometry
Description	Calculate intersection point of two straight lines, and return whether they intersect or not.
Grammar	is_intersect=ZV_INTERSECTLL(x11,y11,x12,y12,x21,y21,x22,y22,tabId) x11: coordinate x of the first point of straight line 1 y11: coordinate y of the first point of straight line 1

	x12: coordinate x of the second point of straight line 1 y12: coordinate y of the second point of straight line 1 x21: coordinate x of the first point of straight line 2 y21: coordinate y of the first point of straight line 2 x22: coordinate x of the second point of straight line 2 y22: coordinate y of the second point of straight line 2 tabId: TABLE index, calculated results are coordinate x, coordinate y in order is_intersect: whether straight lines intersect, 0: straight lines are parallel, no intersection point. 1: straight lines are intersected.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM is_intersect is_intersect=ZV_INTERSECTLL(100,100,200,200,30,30,60,60,0) 'calculate the intersection point of two straight lines and put the intersection coordinates x and y into the TABLE(0), and return is_intersect to check whether the straight lines intersect.

13.1.10. ZV_INTERSECTSS – Segment Intersection Point

Type	Geometry
Description	Calculate intersection point of two segments, and return whether they intersect or not.
Grammar	isIntersect=ZV_INTERSECTSS(x11,y11,x12,y12,x21,y21,x22,y22, tabId) x11: coordinate x of the first point of segment 1 y11: coordinate y of the first point of segment 1 x12: coordinate x of the second point of segment 1 y12: coordinate y of the second point of segment 1 x21: coordinate x of the first point of segment 2 y21: coordinate y of the first point of segment 2 x22: coordinate x of the second point of segment 2

	y22: coordinate y of the second point of segment 2 tabId: TABLE index, calculated results are coordinate x, coordinate y in order isIntersect: whether segments intersect, 0: segments are parallel, no intersection point. 1: segments are intersected.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM isIntersect isIntersect=ZV_INTERSECTSS(100,100,200,200,30,30,60,60,0) 'calculate the intersection point of two segments and put the intersection coordinates x and y into the TABLE(0), and return isIntersect to check whether segments intersect.

13.1.11. ZV_PROJECTPL – Projection of Point on the Straight Line

Type	Geometry
Description	Calculate foot point of point on the straight line or the projection of point on the straight line.
Grammar	ZV_PROJECTPL(px,py,lx1,ly1,lx2,ly2,tabId) Alias: ZV_INTERSECTPL px: coordinate x of the point py: coordinate y of the point 1x1: coordinate x of the first point of straight line 1 1y1: coordinate y of the first point of straight line 1 1x2: coordinate x of the second point of straight line 1 1y2: coordinate y of the second point of straight line 1 tabId: TABLE index, calculated results are coordinate x, coordinate y in order
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_PROJECTPL(10,20,100,100,200,200,0) ‘foot point coordinates x, y of point 10,20 on the straight line are put into TABLE (0)

13.1.12. ZV_PROJECTPC – Projection of Point and Circle

Type	Geometry
Description	Calculate the projection from the point to circle, and the projection point locates on the circle that closes to point most.
Grammar	ZV_PROJECTPC(px,py,cx,cy,radius,tabId) Alias: ZV_INTERSECTPL px: coordinate x of the point py: coordinate y of the point cx: coordinate x of the center cy: coordinate y of the center radius: center radius tabId: TABLE index, projection point coordinates, they are coordinate x, coordinate y in order
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_PROJECTPC(100, 100, 320, 240, 30, 0) 'calculate the projection from point to circle, and save results into TABLE (0)

13.1.13. ZV_PROJECTPE – Projection of Point and Ellipse

Type	Geometry
Description	Calculate the projection from the point to ellipse, and the projection point locates on the circle that closes to ellipse most.
Grammar	ZV_PROJECTPE(px,py,cx,cy,ra,rb,angle,tabId) px: coordinate x of the point py: coordinate y of the point cx: coordinate x of the ellipse cy: coordinate y of the ellipse ra: major semi-axis of ellipse rb: minor semi-axis of ellipse angle: the angle between the long axis and the horizontal direction, unit degree, range (-180, 180]

	tabId: TABLE index, projection point coordinates, they are coordinate x, coordinate y in order
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_PROJECTPE(100, 100, 320, 240, 30, 20, 0, 0) 'calculate the projection from point to ellipse, and save results into TABLE (0)

13.1.14. ZV_RECT2VERTEX – Rotate Rectangular Vertex

Type	Geometry
Description	Calculate four vertexes' coordinates of rotate rectangle, the direction is clockwise.
Grammar	ZV_RECT2VERTEX(cx,cy,w,h,angle,tabId) cx: coordinate x of the rotate rectangle cy: coordinate y of the rotate rectangle w: length of rotate rectangle in the x direction h: length of rotate rectangle in the y direction angle: rotate rectangular angle tabId: TABLE index, output parameters, save rectangular vertex coordinates x, y, x, y in order and in the clockwise.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_RECT2VERTEX(20,20,100,100,60,0) 'output rotate rectangular vertex coordinates x, y (clockwise) into TABLE

13.1.15. ZV_INTERSECTRECT2 – Vertex of Rotate Rectangle Intersection Area

Type	Geometry
Description	Calculate vertex's coordinates of two rotate rectangles in

	intersection are.
Grammar	ZV_INTERSECTRECT2(pts, r1x, r1y, r1w, r1h, r1Angle, r2x, r2y, r2w, r2h, r2Angle) pts: rectangle type, output parameter, N rows 2 columns, calculate vertex's coordinate r1x: the center of coordinate x of the first rotate rectangle r1y: the center of coordinate y of the first rotate rectangle r1w: the length of the first rotate rectangle in x direction r1h: the length of the first rotate rectangle in y direction r1Angle: the angle of the first of rotate rectangle r2x: the center of coordinate x of the second rotate rectangle r2y: the center of coordinate y of the second rotate rectangle r2w: the length of the second rotate rectangle in x direction r2h: the length of the second rotate rectangle in y direction r2Angle: the angle of the second of rotate rectangle
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT pts ZV_INTERSECTRECT2(pts, 20,20,100,100,60,20,20,100,100,0) 'output vertexes' coordinates of rotate rectangles in intersection area

13.1.16. ZV_ANGLELL – Straight Line Angle

Type	Geometry
Description	Calculate angle of straight line 1 and straight line 2. (-180, 180]
Grammar	Angle = ZV_ANGLELL(x11,y11,x12,y12,x21,y21,x22,y22) x11: the x coordinate of line 1 point 1 y11: the y coordinate of line 1 point 1 x12: the x coordinate of line 1 point 2 y12: the y coordinate of line 1 point 2 x21: the x coordinate of line 2 point 1 y21: the y coordinate of line 2 point 1

	x22: the x coordinate of line 2 point 2 y22: the y coordinate of line 2 point 2 returned value: angle: angle, the unit is degree (deg), (-180, 180]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM angle angle = zv_anglell(0,0,1,0,0,0,0,1) 'calculate angle of line 1 and line 2 PRINT angle 'print result is 90 degrees

13.1.17. ZV_ANGLELX – Angle of Line and Horizontal Axis

Type	Geometry
Description	Calculate angle of straight line and x positive direction. (-180, 180], clockwise is positive.
Grammar	angle = ZV_ANGLELX(x1, y1, x2, y2) x1: the x coordinate of line's point 1 y1: the y coordinate of line's point 1 x2: the x coordinate of line's point 2 y2: the y coordinate of line's point 2 returned value: angle: angle, the unit is degree (deg), (-180, 180]
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM angle angle = ZV_ANGLELX(0,0,1,1) 'calculate angle of straight line and x positive direction PRINT angle 'print result is 45 degrees

13.1.18. ZV_ANGLEBISECT – Angle Bisector

Type	Geometry

Description	Calculate angle bisector of two straight lines.
Grammar	ZV_ANGLEBISECT(x1,y1,angle1,x2,y2,angle2,tabId) x1: the x coordinate of line's point 1 y1: the y coordinate of line's point 1 angle 1: straight line 1's angle, the unit is degree, image coordinate system x2: the x coordinate of line's point 2 y2: the y coordinate of line's point 2 angle 2: straight line 2's angle, the unit is degree, image coordinate system tabId: output parameter, TABLE index, linear parameter, x, y, angle (degree)
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_ANGLEBISECT(0,0,0,0,0,90,0) 'calculate angle bisector pf two lines and save them into TABLE (0)

13.1.19. ZV_LINETOPARAM – From Line to Parameters

Type	Geometry
Description	Convert the line represented by two points to the line represented by parameters, that is, use another parameter form to express the same one straight line.
Grammar	ZV_LINETOPARAM(x1, y1, x2, y2, tabId) x1: the x coordinate of the first point of the line y1: the y coordinate of the first point of the line x2: the x coordinate of the second point of the line y2: the y coordinate of the second point of the line tabId: TABLE index, output line parameters, they are center x, y, and angle of x positive direction, line length "len" in order.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_LINETOPARAM(0,0,1,0,0)

	'output line represented by parameters into TABLE (0), and parameters are center x, y, angle (of x positive direction), line length len.

13.1.20. ZV_LINEFROMPARAM – Parameters Construct Line

Type	Geometry
Description	Convert the line represented by the parameters to the line represented by two points.
Grammar	ZV_LINEFROMPARAM(cx, cy, angle, len, tabId) cx: the x coordinate of the center of the line cy: the y coordinate of the center of the line angle: angle of line and axis x, angle value len: line length tabId: TABLE index, output line points coordinates, they are x1, y1, x2, y2
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_LINEFROMPARAM(20, 20, 30, 20, 0) 'output line represented by two points into TABLE (0)

13.1.21. ZV_FITLINE – Line Fitting

Type	Geometry
Description	Use least squares to fit a straight line based on input points
Grammar	ZV_FITLINE(points,tabId[,method=0]) points: fitting point set, matrix type N rows and 2 columns, one point in each row tabId: TABLE index, output parameter, coordinates of two points located on the fitting straight line method: method of straight-line fitting. 0 - least squares, easy to be interfered by favorable

	group points, it can be used for point clusters on a relatively standard straight line, otherwise interference from favorable group points will cause the fitted straight line to be inaccurate. 1 - Ransac, the random sampling consistency principle can remove outlier interference without affecting the accuracy of the fitted line. It can be used when the proportion of favorable points in the total point set is large, such as 1/4 - 1/3, which is more time-consuming. 2 - iterative least squares, which can remove the interference of favorable points without affecting the accuracy of fitting straight lines. It can be used when the proportion of favorable points in the total point concentration is relatively small, such as less than 1/4, which is more time-consuming. The time-consuming situation of the three methods: 1 > 2 > 0. Usually, it is recommended to use 2.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT points TABLE(0, 1, 1, 2, 2, 3, 3) 'save data into TABLE(0) ZV_MATGENDATA(points,3,2,0) 'generate matrix by data ZV_FITLINE(points,10) 'outputs the coordinates of the two endpoints of the straight line. The coordinates are stored in the TABLE (0)

13.1.22. ZV_FITPOLYN – Polynomial Fitting

Type	Geometry
Description	Fit polynomial, order specifies “order” to be fitted.
Grammar	ZV_FITPOLYN(pts, order, tabId) pts: ZVOBJECT type, point set for fitting polynomials order: order of fitting tabId: TABLE index, output parameter order+1 polynomial

	coefficients, fitting results, low-order terms first, such as second-order polynomial fitting c+bx+ax^2.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT points TABLE(0, 1, 1, 2, 2, 3, 3) 'Save data into TABLE(10) ZV_MATGENDATA(points,3,2,0) ZV_FITPOLYN(points,1,10) 'fit as a first-order polynomial, and the output results are 0 and 1, that is, y=x

13.1.23. ZV_ROTATEPOINT – Rotate Point

Type	Geometry
Description	Rotate one point around the center point.
Grammar	ZV_ROTATEPOINT(x,y,cx,cy,angle,tabId) x: input coordinate x y: input coordinate y cx: coordinate x of the center point cy: coordinate y of the center point angle: rotate angle, the unit is degree, clockwise is positive, image coordinate system tabId: TABLE index, output parameters, coordinates of rotate point, x, y in order.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_ROTATEPOINT(10,10,320,240,60,0) 'rotate point (10,10) 60 degrees around center (320,240)

13.1.24. ZV_PTSDIRECT – Calculate Direction of 3 Points

Type	Geometry
Description	Calculate continuous three points' rotate direction.

Grammar	ret = ZV_PTSDIRECT (x1, y1, x2, y2, x3, y3) x1: coordinate x of point 1 y1: coordinate y of point 1 x2: coordinate x of point 2 y2: coordinate y of point 2 x3: coordinate x of point 3 y3: coordinate y of point 3 return value: ret: direction, -1: clockwise, 0: shared line, 1: anticlockwise, image coordinate system
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM ret ret=ZV_PTSDIRECT(10,10,20,20,29,29) 'calculate the rotation direction of three consecutive points

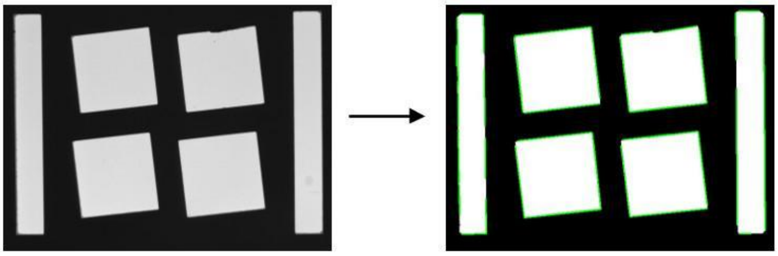
13.1.25. ZV_RECT2INSIZE – Whether Rectangle's Vertex Are in Range

Type	Geometry
Description	Judge whether 4 vertexes of rectangle are in the given right rectangle size or not.
Grammar	ret = ZV_RECT2INSIZE(width,height,cx,cy,w,h,angle) width: right rectangle width height: right rectangle height cx: coordinate x of center point cy: coordinate y of center point w: rectangle width h: rectangle height angle: rotate angle, the unit is degree, clockwise is positive, image coordinate system ret: 1: four vertexes are in the range. 0: four vertexes are out of the range
Controller	It is valid in controllers that support ZV function or they belong

	to 5XX series or above.
Example	DIM ret ret=ZV_RECT2INSIZE(640, 480, 320, 240, 120, 80, 0) 'judge whether 4 vertexes of rectangle are in the rectangle range

13.1.26. ZV_HOUGHLINE -- Hough Find Line

Type	Geometry
Description	Use probabilistic Hough to find straight lines. Use probabilistic Hough transform to find straight lines that meet the requirements from binary images.
Grammar	ZV_HOUGHLINE (img, lines, rho, theta, thresh, minLineLen, maxLineGap) img: ZVOBJECT type, single-channel binary image lines: ZVOBJECT type, nx4 matrix, the starting coordinates and end coordinates of the straight line are stored in the columns. rho: the distance accuracy of the accumulator in pixels. The higher the accuracy (the smaller the value), the more time-consuming it is. 1 is used commonly. theta: the angular accuracy of the accumulator in degrees. The higher the accuracy (the smaller the value), the more time-consuming it is. 1 is used commonly. thresh: accumulator threshold, that is, the vote value that it must reach in the accumulator to identify a part as a straight line in the graph. Only line segments larger than thresh can be detected and returned to the result. minLineLen: the minimum line segment length. Only line segments larger than this parameter are detected. maxLineGap: If the distance between the end points of two segments on a collinear straight line is less than this parameter, it is considered to be a line segment.
Controller	It is valid in controllers that support ZV function or they belong

	to 5XX series or above.
Example	 ZVOBJECT img, imgBw, imgCanny, dst, lines DIM row, rows ZV_READIMAGE(img,"test.png",0) 'read the image in the original image format ZV_THRESH(img,imgBw,120,255) 'image binarization ZV_CANNY(imgBw,imgCanny,10,200,3) 'extract edges ZV_HOUGHLINE(imgCanny,lines,1,1,30,10,10) 'HOUGHLINE ZV_GRAYTORGB(imgBw,dst) 'convert binary image to RGB rows = ZV_MATROWS(lines) 'get the number of lines in lines FOR row = 0 TO rows-1 ZV_MATGETROW(lines,row,4,0) 'get a certain line of lines ZV_LINE(dst,TABLE(0),TABLE(1),TABLE(2),TABLE(3),zv_color(0,255,0)) 'draw a straight line NEXT

13.1.27. ZV_HOUGCIRCLE -- Hough Find Circle

Type	Geometry
Description	Use canny edge detection on the image to detect edges, and then perform Hough voting on the edge points to find circles that meet the conditions. This operator can easily find the center of the circle, but it may not find a suitable circle radius, so a suitable minimum radius and a maximum radius are needed.
Grammar	ZV_HOUGHCIRCLE (img, circles, minDis, edgeThresh, thresh, minR, maxR) img: ZVOBJECT type, single-channel grayscale image circles: ZVOBJECT type, nx3 matrix, the column stores the center coordinates and radius in sequence.

	minDis: minimum distance. If the distance between the centers of two circles is less than this value, they are considered as the same circle. edgeThresh: canny edge detection high threshold, low threshold uses half of the high threshold by default thresh: the threshold for determining whether a point on the accumulation plane is the center of the circle. The larger it is, the closer the circle that can pass detection is to a perfect circle. 100 is used commonly. minR: minimum radius of circle maxR: maximum radius of circle
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img, imgBw, imgCanny, dst, lines DIM row, rows ZV_READIMAGE(img,"test.png",0) 'read the image in the original image format ZV_HOUGHCIRCLE (img, circles, 30, 233, 60, 20, 450) 'HOUGCIRCLE ZV_GRAYTORGB (img,dst) 'convert grayscale image to RGB rows = ZV_MATROWS(circles) 'get the row number of circles FOR row = 0 TO rows-1 ZV_MATGETROW(circles,row,3,0) 'get a certain row of circles ZV_CIRCLE(dst,TABLE(0),TABLE(1),TABLE(2),TABLE(3),zv_color(0,255,0)) 'draw a circle NEXT

13.1.28. ZV_GENCIRCLE – Make One Circle By 3 Points

Type	Geometry
Description	The circle is made by three points.
Grammar	ZV_GENCIRCLE(x1,y1,x2,y2,x3,y3,tabId) x1: coordinate x of point 1 y1: coordinate y of point 1 x2: coordinate x of point 2 y2: coordinate y of point 2 x3: coordinate x of point 3 y3: coordinate y of point 3 tabId: output parameters, output TABLE No. of circle information, they are cx, cy, radius.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZV_GENCIRCLE(-1, 0, 0, 1, 1, 0, 0) '3 points to make one circle

13.1.29. ZV_FITCIRCLE – Circle Fitting

Type	Geometry
Description	Use least squares to fit a circle based on input points
Grammar	ZV_FITCIRCLE(points,tabId[,method=0]) points: fitting point set, matrix type N rows and 2 columns, one point in each row tabId: TABLE index, output parameter, fit circle, they are cx, cy, radius. method: method of circle fitting. 0 - least squares, easy to be interfered by favorable group points, it can be used for point clusters on a relatively standard circle, otherwise interference from favorable group points will cause the fitted circle to be inaccurate. 1 - Ransac, the random sampling consistency principle can remove outlier interference without affecting the accuracy of the fitted circle. It can be used when the

	proportion of favorable points in the total point set is large, such as 1/4 - 1/3, which is more time-consuming. 2 - iterative least squares, which can remove the interference of favorable points without affecting the accuracy of fitting circle. It can be used when the proportion of favorable points in the total point concentration is relatively small, such as less than 1/4, which is more time-consuming. The time-consuming situation of the three methods: 1 > 2 > 0. Usually, it is recommended to use 2.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT points TABLE(0, 0, 0, 1, 1, 0, 0) 'save data into TABLE(0) ZV_MATGENDATA(points,3,2,0) 'generate matrix by data ZV_FITCIRCLE(points, 0, 2) 'fit circle and output circle information, and put them into TABLE (0) in order

13.1.30. ZV_FITELLIPSE – Ellipse Fitting

Type	Geometry
Description	Use least squares to fit an ellipse based on input points
Grammar	ZV_FITELLIPSE(pts, tabId) pts: fitted point set, matrix type, N rows 2 columns, one point of each row. tabId: TABLE index, output parameter, fit ellipse, they are cx, cy, xr, yr, angle
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT points TABLE(0, 0, 0, 1, 1, 0, 0) 'save data into TABLE(0) ZV_MATGENDATA(points,3,2,0) 'generate matrix by data ZV_FITELLIPSE(points, 0)

	'fit ellipse and output circle information, and put them into TABLE (0) in order

13.2. Transformation

13.2.1. ZV_MAT2DADDTRANS – Add Translation for Transformation Matrix

Type	Transform
Description	In transformation matrix, add translation. isBaseAfter is 0: $$\begin{bmatrix} a1 & a2 & a3 \\ a4 & a5 & a6 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$$ Transformation matrix "mat" translation isBaseAfter is 1: $$\begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} a1 & a2 & a3 \\ a4 & a5 & a6 \\ 0 & 0 & 1 \end{bmatrix}$$ Translation Transformation matrix "mat"
Grammar	ZV_MAT2DADDTRANS (mat, tx, ty, isBaseAfter) mat: ZVOBJECT type, transformation matrix tx: x direction offset parameter ty: y direction offset parameter isBaseAfter: whether the parameter is based on transformation, if it is 1, it is based on transformation, that is, the translation transformation is equivalent to executing after the mat transformation, which is equivalent to executing the original transformation of mat first and then executing the transformation specified by the parameter.

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat TABLE(0, 1, 0.2, 0, 0, 1, 0) 'save data into TABLE(0) ZV_MATGENDATA(mat,2,3,0) 'transform matrix ZV_MAT2DADDTRANS (affine_mat, 5, 5, 1) 'add offset (5,5) for transform matrix "mat".

13.2.2. ZV_MAT2DADDROT – Add Rotate for Transformation Matrix

Type	Transform
Description	In transformation matrix, add rotation. isBaseAfter is 0: $\begin{bmatrix} a1 & a2 & a3 \\ a4 & a5 & a6 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ Transformation matrix "mat" rotation isBaseAfter is 1: $\begin{bmatrix} 1 & 0 & cx \\ 0 & 1 & cy \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -cx \\ 0 & 1 & -cy \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} a1 & a2 & a3 \\ a4 & a5 & a6 \\ 0 & 0 & 1 \end{bmatrix}$ rotation base rotation rotate base Transformation matrix "mat"
Grammar	ZV_MAT2DADDROT (mat, angle, cx, cy, isBaseAfter) mat: ZVOBJECT type, transformation matrix angle: angle of rotation, clockwise is positive cx: x coordinate of rotation base point, it is valid when isBaseAfter is 1 cy: y coordinate of rotation base point, it is valid when isBaseAfter is 1 isBaseAfter: whether the parameter is based on after

	transformation, if it is 1, it is based on transformation, that is, the rotation transformation is equivalent to executing after the mat transformation.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat TABLE(0, 1, 0.2, 0, 0, 1, 0) 'save data into TABLE(0) ZV_MATGENDATA(mat,2,3,0) 'transformation matrix ZV_MAT2DADDROT (mat, -20, 0, 0, 1) 'rotate matrix "mat" 30 degrees at (0, 0) firstly, then scale out 0.8, then rotate -20 degrees at rotation base point (20, 20)

13.2.3. ZV_MAT2DADDSCALE – Add Scaling for Transformation Matrix

Type	Transform
Description	In transformation matrix, add scaling. isBaseAfter is 0: $\begin{bmatrix} a1 & a2 & a3 \\ a4 & a5 & a6 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$ Transformation matrix "mat" scaling isBaseAfter is 1: $\begin{bmatrix} 1 & 0 & cx \\ 0 & 1 & cy \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -cx \\ 0 & 1 & -cy \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} a1 & a2 & a3 \\ a4 & a5 & a6 \\ 0 & 0 & 1 \end{bmatrix}$ scaling base rotation scaling base Transformation matrix "mat"
Grammar	ZV_MAT2DADDSCALE (mat, sx, sy, cx, cy, isBaseAfter) mat: ZVOBJECT type, transformation matrix sx: scaling coefficient in x direction sy: scaling coefficient in y direction

	cx: x coordinate of scaling base point, it is valid when isBaseAfter is 1 cy: y coordinate of scaling base point, it is valid when isBaseAfter is 1 isBaseAfter: whether the parameter is based on after transformation, if it is 1, it is based on transformation, that is, the scaling transformation is equivalent to executing after the mat transformation.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat TABLE(0, 1, 0.2, 0, 0, 1, 0) 'save data into TABLE(0) ZV_MATGENDATA(mat,2,3,0) 'transformation matrix ZV_MAT2DAASCALE (mat, 0.8, 1.2, 20, 20, 1) ‘scale out 0.8 times in x direction for transformation matrix “mat”, and scale in 1.2 times in y direction, and the base point of scaling is 20, 20

13.2.4. ZV_GETSIMILARITYP – Build Similarity Transformation Matrix

Type	Transform
Description	Construct a similarity transformation matrix based on parameters. Similarity transformation is a type of transformation that transforms graphics. It can perform rotation, scaling, translation and other transformations on graphics. The length ratio and angle remain unchanged before and after transformation. It is similar to similarity triangles. In the same way, it can also be used to transform two-dimensional coordinates.
Grammar	ZV_GETSIMILARITYP (mat, cx, cy, angle, scale) mat: ZVOBJECT type, matrix type, calculated similarity transformation matrix, 2 rows and 3 columns cx: x coordinate of rotation center of similarity

	transformation cy: y coordinate of rotation center of similarity transformation angle: rotation angle of similarity transformation, clockwise is positive scale: scaling of similarity transformation
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat ZV_GETSIMILARITYP (mat, 0, 0, 30, 1) 'according to related parameters, build transformation matrix, it is 2 rows and 3 columns, build one similarity transformation matrix "mat" whose rotation center is (0,0), rotation angle is 45 and scaling is 1.

13.2.5. ZV_GETRIGIDVECTOR – Calculate Rigid Transformation Matrix

Type	Transform
Description	According to the vector transformation relationship, the rigid transformation matrix is calculated using the vectors before and after transformation, and the angles are all positive clockwise. Rigid transformation is a type of transformation that transforms graphics. It can perform rotation, translation and other transformations on graphics. The length and area of the graphics remain unchanged before and after the transformation, and the shape does not change. In the same way, the two-dimensional coordinates of the space can also be transformed.
Grammar	ZV_GETRIGIDVECTOR (mat, x1, y1, angle1, x2, y2, angle2) mat: ZVOBJECT type, matrix type, calculated rigid transformation matrix, 2 rows and 3 columns x1: x coordinate of vector before transformation y1: y coordinate of vector before transformation angle1: the direction of vector 1 before transformation

	x2: x coordinate of vector after transformation y2: y coordinate of vector after transformation angle2: the direction of vector 2 after transformation
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat ZV_GETRIGIDBVECTOR (mat, 0, 0, 30, 5, 5, 60) 'According to the relationship between vectors, the rigid transformation matrix mat is calculated using the vectors before and after transformation.

13.2.6. ZV_GETRIGID – Calculate Rigid Transformation Matrix

Type	Transform
Description	The rigid transformation matrix is calculated using two points before and after transformation. Rigid transformation is a type of transformation that transforms graphics. It can perform rotation, translation and other transformations on graphics. The length and area of the graphics remain unchanged before and after the transformation, and the shape does not change. In the same way, the two-dimensional coordinates of the space can also be transformed.
Grammar	ZV_GETRIGIDVECTOR (mat, tabIdSrc, tabIdDst) mat: ZVOBJECT type, matrix type, calculated rigid transformation matrix, 2 rows and 3 columns tabIdSrc: TABLE index, two points before transformation, they are x1, y1, x2, y2 in order tabIdDst: TABLE index, two points after transformation, they are x1, y1, x2, y2 in order
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat TABLE (0, 0, 0, 2, 2) 'save data into TABLE (0)

	TABLE (100, 1, 0, 7, 10) 'save data into TABLE (100) ZV_GETRIGID (mat, 0, 100) 'save two points before transformation into TABLE (0) and save two points after transformation into TABLE (100), then calculate two points before and after transformation to get rigid transformation matrix "mat".

13.2.7. ZV_GETSIMILARITY – Calculate Similarity Transformation Matrix

Type	Transform
Description	The similarity transformation matrix is calculated using two points before and after transformation. Similarity transformation is a type of transformation that transforms graphics. It can perform rotation, translation and other transformations on graphics. The length and area of the graphics remain unchanged before and after the transformation, and the shape does not change. In the same way, the two-dimensional coordinates of the space can also be transformed.
Grammar	ZV_GETSIMILARITY (mat, tabIdSrc, tabIdDst) mat: ZVOBJECT type, matrix type, output parameter, calculated transformation matrix, 2 rows and 3 columns tabIdSrc: TABLE index, two points before transformation, they are x1, y1, x2, y2 in order tabIdDst: TABLE index, two points after transformation, they are x1, y1, x2, y2 in order
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat TABLE (0, 0, 0, 2, 2) 'save data into TABLE (0) TABLE (100, 1, 0, 7, 10) 'save data into TABLE (100) ZV_GETSIMILARITY (mat, 0, 100) 'save two points before transformation into TABLE (0) and save two points after transformation into TABLE (100), then

	calculate two points before and after transformation to get similarity transformation matrix "mat".

13.2.8. ZV_GETAFFINE – Calculate Affine Transformation Matrix

Type	Transform
Description	The affine transformation matrix is calculated using three points before and after transformation. Affine transformation is a type of transformation that transforms graphics. It can perform rotation, scaling, translation, oblique cutting (also called tilting, cross-cutting) and other transformations on graphics. It has the characteristics of straightness and parallelism, that is, straight lines are still straight lines before and after the transformation., parallel lines are still parallel lines. In the same way, the two-dimensional coordinates of space can also be transformed.
Grammar	ZV_GETAFFINE (mat, tabIdSrc, tabIdDst) mat: ZVOBJECT type, matrix type, output parameter, calculated affine transformation matrix, 2 rows and 3 columns tabIdSrc: TABLE index, three points before transformation, they are coordinates x and y in order tabIdDst: TABLE index, three points after transformation, they are coordinates x and y in order
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat TABLE (0, 0, 0, 2, 2, 5, 5) 'save data into TABLE (0) TABLE (100, 1, 0, 7, 10, 5, 3) 'save data into TABLE (100) ZV_GETAFFINE (mat, 0, 100) 'calculate three points before and after transformation to get affine transformation matrix "mat".

13.2.9. ZV_ESTSIMILARITY -- Estimate Similarity Matrix

Type	Transform
Description	Estimate the similarity transformation matrix based on multiple points (at least 2 pairs) using the RANSAC (Random Sampling Consistency) algorithm. Similar transformation is a type of transformation that transforms graphics. It can perform rotation, scaling, translation and other transformations on graphics. The length ratio and angle remain unchanged before and after transformation. It is similar to similar triangles. In the same way, it can also be used to transform two-dimensional coordinates. Estimating the similarity transformation matrix based on the front and rear point pairs means that the front and rear point pairs may not be completely absolute one-to-one correspondence, and there may be a certain deviation. Points with a deviation greater than thresh are considered outliers, and points less than or equal to thresh are considered inline points. And outlier points will be eliminated during the estimation process and the remaining inline points will eventually be used to estimate the matrix. The estimated matrix will minimize the point error before and after the transformation.
Grammar	ZV_ESTSIMILARITY(from,to,mat,thresh,confidence,tabId) from: ZVOBJECT type, point before transformation, matrix representation to: ZVOBJECT type, points after transformation, matrix representation mat: ZVOBJECT type, matrix type, estimated transformation matrix, 2 rows and 3 columns thresh: point projection error threshold. Points with an error less than or equal to thresh are considered inline points. The recommended value is 3. confidence: confidence, ranging from 0-100, used for matrix estimation, usually between 95 and 99 is enough, 99 is recommended, too close to 100 will reduce the estimation

	speed, lower than 80-90 may lead to inaccurate estimation precise tabId: TABLE index, output parameter, corresponding to the selected state of the input parameter from or to point set after iteration, that is, the selected state of inline points is 1, and the selected state of outlier points is 0
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT matSrc, matDst, matAffine TABLE(0,0,0,2,2,5,5) 'stores the coordinates of the 3 points before transformation ZV_MATGENDATA(matSrc,2,3,0) 'matrix of points before transformation TABLE(100,1,0,7,10,5,3) 'stores the coordinates of the 3 points after transformation ZV_MATGENDATA(matDst,2,3,100) 'matrix of points before transformation ZV_ESTSIMILARITY(matSrc,matDst,matAffine,3,99,0) 'use the point "from" before transformation and the point "to" after transformation to estimate the similarity transformation matrix "mat"

13.2.10. ZV_ESTAFFINE -- Estimate Affine Matrix

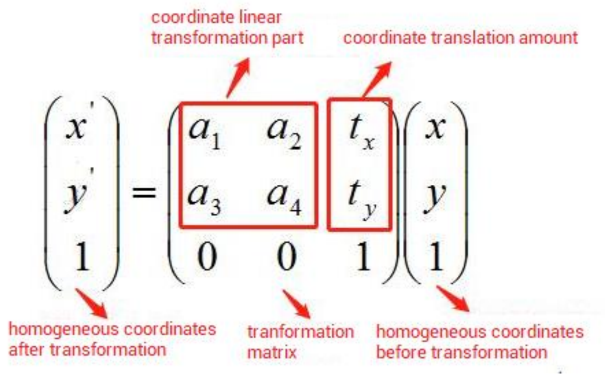
Type	Transform
Description	Estimate the affine transformation matrix based on multiple points (at least 3 pairs) using the RANSAC (Random Sampling Consistency) algorithm. Affine transformation is a type of transformation that transforms graphics. It can perform rotation, scaling, translation, oblique cutting (also called tilting, cross-cutting) and other transformations on graphics. It has the characteristics of straightness and parallelism, that is, straight

	lines are still straight lines before and after the transformation., parallel lines are still parallel lines. In the same way, the two-dimensional coordinates of space can also be transformed. Estimating the affine transformation matrix based on the front and rear point pairs means that the front and rear point pairs may not be completely absolute one-to-one correspondence, and there may be a certain deviation. Points with a deviation greater than thresh are considered outliers, and points less than or equal to thresh are considered inline points. And outlier points will be eliminated during the estimation process and the remaining inline points will eventually be used to estimate the matrix. The estimated matrix will minimize the point error before and after the transformation.
Grammar	ZV_ESTAFFINE(from,to,mat,thresh,confidence,tabId) from: ZVOBJECT type, point before transformation, matrix representation to: ZVOBJECT type, points after transformation, matrix representation mat: ZVOBJECT type, matrix type, estimated transformation matrix, 2 rows and 3 columns thresh: point projection error threshold. Points with an error less than or equal to thresh are considered inline points. The recommended value is 3. confidence: confidence, ranging from 0-100, used for matrix estimation, usually between 95 and 99 is enough, 99 is recommended, too close to 100 will reduce the estimation speed, lower than 80-90 may lead to inaccurate estimation precise tabId: TABLE index, output parameter, corresponding to the selected state of the input parameter from or to point set after iteration, that is, the selected state of inline points is 1, and the selected state of outlier points is 0
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT matSrc, matDst, matAffine

	TABLE(0,0,0,2,2,5,5) 'stores the coordinates of the 3 points before transformation ZV_MATGENDATA(matSrc,2,3,0) 'matrix of points before transformation TABLE(100,1,0,7,10,5,3) 'stores the coordinates of the 3 points after transformation ZV_MATGENDATA(matDst,2,3,100) 'matrix of points before transformation ZV_ESTAFFINE(matSrc,matDst,matAffine,3,99,0) 'use the point "from" before transformation and the point "to" after transformation to estimate the similarity transformation matrix "mat"

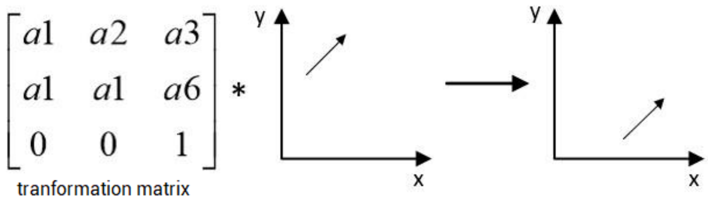
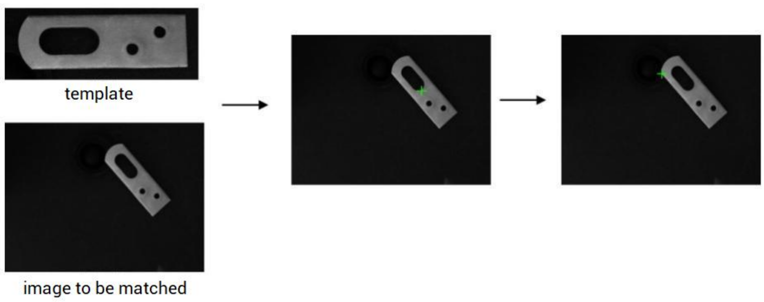
13.2.11. ZV_AFFINETRANS – Affine Transformation

Type	Transform
Description	Performs an affine transformation on num points. Affine transformation has a wide range of transformations, including rigid transformation and similarity transformation. Therefore, rigid transformation matrices, similarity transformation matrices, and affine transformation matrices can all use this command to transform coordinate points. The transformation matrix of this instruction is 2 rows and 3 columns because the last row of the homogeneous transformation matrix with 3 rows and 3 columns is fixed data 0, 0, 1, and the two-dimensional coordinates (x, y) are converted into homogeneous coordinates (x, y,1) only needs to add 1 to the third dimension, so the formula for transforming the two-dimensional coordinate point is as follows:

	 The linear transformation part a1, a2, a3, a4 of the transformation matrix is responsible for linear transformation such as rotation, scaling, oblique cutting, etc. on the coordinate point (x, y). The coordinate translation amount tx, ty is responsible for performing translation on the coordinate point (x, y). The transformation equation is as follows: $x' = a_1 * x + a_2 * y + t_x$ $y' = a_3 * x + a_4 * y + t_y$
Grammar	ZV_AFFINETRANS(mat,num,tabIdSrc,tabIdDst) mat: ZVOBJECT type, transformation matrix num: the number of coordinate points tabIdSrc: TABLE index, the coordinate point to be transformed, x and y are stored in sequence starting from the TABLE index tabIdDst: output parameters, TABLE index, transformed coordinate points, store x, y in sequence
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat TABLE(0, 1, 0.2, 0, 0, 1, 0) 'save data into TABLE(0) ZV_MATGENDATA(mat,2,3,0) 'transformation matrix TABLE(10,0,0,2,2,5,5) 'store the coordinates of the three points before transformation ZV_AFFINETRANS(mat,3,10,100) 'use the affine transformation matrix mat to perform affine transformation on the input coordinate points, and stores

	the transformed points into TABLE (100)

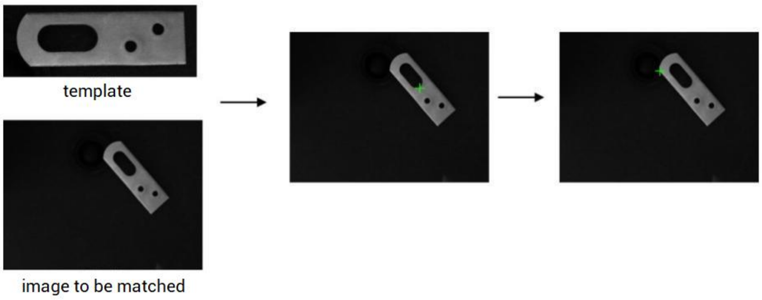
13.2.12. ZV_VECTORCORRECT – Vector Correction

Type	Transform
Description	Correct input vector. 
Grammar	ZV_VECTORCORRECT(mat,vecx,vecy,veca,tabId) mat: ZVOBJECT type, corrected transformation matrix, the matrix is 2*3 or 3*3 vecx: starting x coordinate of the input vector vecy: the starting y coordinate of the input vector veca: angle of the input vector, clockwise is positive tabId: TABLE index, output parameters, corrected vector parameters, x, y, angle in order
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img,clrImg,mat,mod,matchImg,rlts ZVOBJECT matRigid,modContList,dstContList ZV_READIMAGE(img, "model.jpg", 0) 'read the image in the original image format ZV_SHAPECREATE(img,mod,0,360,1,1,50,0,0,0,0) 'create template ZV_SHAPECONTOURS(mod, modContList, 0)

	'get template contour ZV_READIMAGE(matchImg, "1.png", 0) 'read the image in the original image format ZV_SHAPEFIND(mod,matchImg,rlts,90,1,0,-1,3,9,0) 'template matching ZV_MATGETROW(rlts,0,5,0) 'obtain the first row of the matching result matrix, which are: matching score "score", x coordinate, y coordinate, rotation angle "angle", and scaling ratio "scale" ZV_GRAYTORGB(matchImg,clrImg) 'convert grayscale image to RGB image TABLE(10, 1, 0, -95, 0, 1, -55) 'save data into TABLE(0) ZV_MATGENDATA(mat,2,3,10) 'transformation matrix ZV_VECTORCORRECT(mat,TABLE(1),TABLE(2),TABLE(3),20) 'use the transformation matrix mat to correct the input vector (x1, y1, angle1), and the corrected vector is stored in TABLE (20) ZV_MARKER(clrImg,TABLE(20),TABLE(21),0,30,zv_color(0,255,0)) 'draw mark

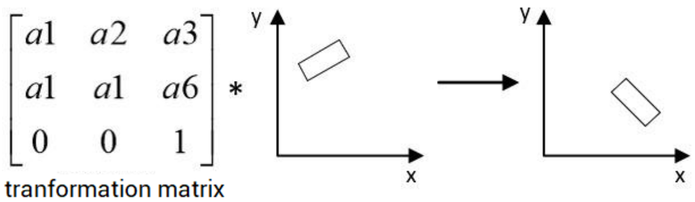
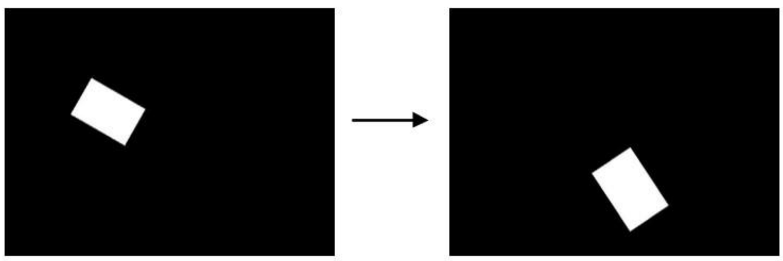
13.2.13. ZV_POSECORRECT – Vector Correction

Type	Transform
Description	To correct the input vector, add a translation amount to the point position in the vector to achieve a customized position. It is usually used to customize the positioning output point. For example, if the coordinates of the mark point in the positioning output are x1, y1, angle1, then if you want to specify the positioning coordinates as a position x2, y2 next to the mark point. Then, the positioning point can be corrected, that is, transx = x2-x1, transy = y2-y1, ZV_POSECORRECT (x1, y1, angle1, transx, transy, 0)
Grammar	ZV_POSECORRECT(vecx,vecy,veca,transx,transy,tabId) vecx: starting x coordinate of the input vector

	vecy: the starting y coordinate of the input vector veca: angle of the input vector, clockwise is positive transx: translation x of vector coordinate transy: translation y of vector coordinate tabId: TABLE index, output parameters, corrected vector parameters, x, y, angle in order
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img,clrImg,mat,mod,matchImg,rlts ZVOBJECT matRigid,modContList,dstContList ZV_READIMAGE(img, "model.jpg", 0) 'read the image in the original image format ZV_SHAPECREATE(img,mod,0,360,1,1,50,0,0,0,0) 'create template ZV_SHAPECONTOURS(mod, modContList, 0) 'get template contour ZV_READIMAGE(matchImg, "1.png", 0) 'read the image in the original image format ZV_SHAPEFIND(mod,matchImg,rlts,90,1,0,-1,3,9,0) 'template matching ZV_MATGETROW(rlts,0,5,0) 'obtain the first row of the matching result matrix, which are: matching score "score", x coordinate, y coordinate, rotation angle "angle", and scaling ratio "scale" ZV_GRAYTORGB(matchImg,clrImg) 'convert grayscale image to RGB image TABLE(10, 1, 0, -95, 0, 1, -55) 'save data into TABLE(0) ZV_MATGENDATA(mat,2,3,10) 'transformation matrix ZV_POSECORRECT(mat, TABLE(1), TABLE(2), TABLE(3), -105,

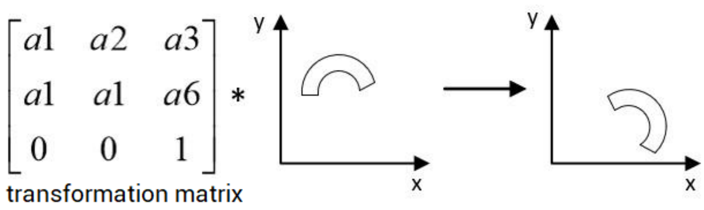
	35, 20) 'translate -105 and 35 pixels in the x and y direction respectively for input vector (1, 1, 30), the correct it, and the corrected vector is stored in TABLE (20) ZV_MARKER(clrImg,TABLE(20),TABLE(21),0,30,zv_color(0,255,0)) 'draw mark

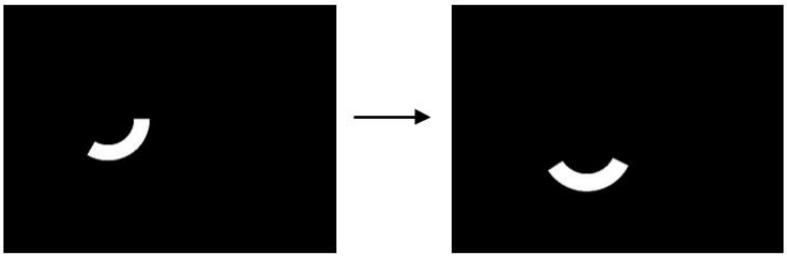
13.2.14. ZV_RECT2RCORRECT – Rectangle Correction

Type	Transform
Description	Correct the input rectangular ROI. Usually, the rectangle is used as the ROI parameter combined with the positioning offset correction matrix to correct the ROI.  tranformation matrix
Grammar	ZV_RECT2CORRECT(mat,cx,cy,width,height,angle,tabId) mat: ZVOBJECT type, corrected transformation matrix cx: x coordinate of the input rectangle center cy: y coordinate of the input rectangle center width: input rectangle width height: input rectangle height angle: input rectangle angle tabId: TABLE index, output parameters, corrected rectangle parameters, they are cx, cy, width, height, angle in order
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	

	ZVOBJECT img,reSrc,reDst,mat ZV_READIMAGE(img,"1.png",0) 'read the image in the original image format ZV_IMGSETCONST(img,0) 'constant fill image ZV_REGENRECT2(reSrc,200,200,120,80,30) 'generate an angled rectangular area TABLE(0, 1, 0.5, 50, 0.5, 1, 50) 'save data into TABLE(0) ZV_MATGENDATA(mat,2,3,0) 'transformation matrix ZV_RECT2CORRECT(mat,200,200,120,80,30,10) 'use the transformation matrix mat to correct the input rectangle, and the corrected rectangle is stored in TABLE (10) ZV_REGENRECT2(reDst,TABLE(10),TABLE(11),TABLE(12),TABLE(13),TABLE(14)) 'generate an angled rectangular area ZV_REGION(img,reDst,0,ZV_COLOR(255,255,255)) 'region to binarization

13.2.15. ZV_SECTRCORRECT – Sector Correction

Type	Transform
Description	Correct the input sector ROI. Usually, the sector is used as the ROI parameter combined with the positioning offset correction matrix to correct the ROI. 
Grammar	ZV_SECT2CORRECT(mat,cx,cy,r1,r2,stAngle,extAngle,tabId) mat: ZVOBJECT type, corrected transformation matrix cx: x coordinate of the input sector center cy: y coordinate of the input sector center r1: inner circle radius of input sector, >0 r2: outer circle radius of input sector, >0 and r2 > r1

	stAngle: starting angle of input sector, unit is degree extAngle: angle range of input sector, unit is degree, >0 tabId: TABLE index, output parameters, corrected sector parameters, they are cx,cy,r1,r2,stAngle,extAngle,tabId in order.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img,reSrc,reDst,mat ZV_READIMAGE(img,"1.png",0) 'read the image in the original image format ZV_IMGSETCONST(img,0) ZV_REGENRECT2(reSrc,200,220,50,80,0,120) 'generate a sector area TABLE(0, 1, 0.5, 50, 0.5, 1, 50) 'save data into TABLE(0) ZV_MATGENDATA(mat,2,3,0) 'transformation matrix ZV_SECTCORRECT(mat,200,220,50,80,0,120,10) 'use the transformation matrix mat to correct the input sector, and the corrected rectangle is stored in TABLE (10) ZV_REGENRECT2(reDst,TABLE(10),TABLE(11),TABLE(12),TABLE(13),TABLE(14),TABLE(15)) 'generate a sector area ZV_REGION(img,reDst,0,ZV_COLOR(255,255,255)) 'region to binarization

13.2.16. ZV_AFFINETOPARAM – Transformation Parameter

Type	Transform
Description	Get the transformation parameters of the transformation matrix Calculate the affine transformation parameters corresponding

	to the homogeneous two-dimensional transformation matrix mat. The parameters sx and sy determine how the transformation scales the original x- and y-axes respectively. The angle slant describes whether the transformed coordinate axis is tilted. If |slant| > 90°, the transformation includes a mirror. The angle "angle" determines the rotation angle of the transformed x-axis relative to the original x-axis. The parameters tx and ty determine the translation of the coordinate system. The matrix can be generated by the following transformation parameters in steps of scaling, tilting, rotation and translation.
Grammar	ZV_AFFINETOPARAM(mat,tabId) mat: ZVOBJECT type, matrix tabId: TABLE index of the output parameters, which are sx, sy, angle, slant, tx, ty. The meaning of each parameter is as follows: sx scaling coefficient in x-axis direction, > 0 sy scaling coefficient in the y-axis direction, > 0 angle: rotation of the transformed x-axis relative to the original x-axis slant: y direction slope, an absolute value > 90° indicates the existence of a mirror image tx: x direction translation ty: y direction translation
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT mat TABLE(0, 1, 0.2, 0, 0, 1, 0) 'save data into TABLE(0) ZV_MATGENDATA(mat,2,3,0)'transformation matrix ZV_AFFINETOPARAM(mat,10) 'obtain the transformation parameters of the mat matrix and stores them in the TABLE (10)

13.3. Correction

13.3.1. ZV_GENCORRECTION – Generate Position Correction Model

Type	Transformation
Description	Generate a position correction model through the reference point coordinates and reference direction.
Grammar	ZV_GENCORRECTION (ref,x,y,angle) ref: generated correction model, ZVOBJECT type x: base x coordinate of corrected reference point y: base y coordinate of corrected reference point angle: base direction of corrected reference point, the unit is angle
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

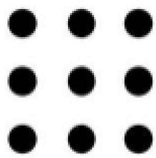
13.3.2. ZV_APPLYCORRECTION – Execution Position Correction

Type	Transformation
Description	Perform position correction on obj, which only supports contour matching defect detectors. The correction is based on the change of the actual coordinates and direction of the reference point relative to the reference point, and the correction object is adjusted so that the relationship between the corrected object and the reference point remains consistent.
Grammar	ZV_APPLYCORRECTION (ref,obj,x,y,angle) ref: correction reference point model obj: the object to be corrected, ZVOBJECT type, it supports contour matching defect detectors. x: real x coordinate of corrected reference point

	y: real y coordinate of corrected reference point angle: real direction of reference point, the unit is angle
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

13.4. Calibration

13.4.1. ZV_CALGENSCATAB – Generate Solid Circle Array Calibration Plate Image

Type	Calibration
Description	Generate solid circle array calibration plate image. Generate image size of [width, height] = [(cols + 3) * 4 * radius, (rows + 3) * 4 * radius], the maximum data volume of the generated image cannot exceed 2G, that is, width *height<= 2048*1024 *1024
Grammar	ZV_CALGENSCATAB (img, rows, cols, radius, polar) img: ZVOBJECT type, output parameter, generated calibration plate grayscale image rows: the number of circles in vertical direction, >0 cols: the number of circles in horizontal direction, >0 radius: the radius of circle point, the unit is pixel, >0 polar: circle color, 0: black circle with white background, 1: white circle with black background.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	 ZVOBJECT img ZV_CALGENSCATAB (img, 4, 3, 30, 0)

	'generate the calibration image that belongs to black circle with white background, there are black circles of 4 rows and 3 columns in the image.

13.4.2. ZV_CALGENCHESSTAB – Generate Chess Calibration Plate Image

Type	Calibration
Description	Generate chess calibration plate image. Generate image size of [width, height] = [(cols + 2) * blockSize, (rows + 2) * blockSize], the maximum data volume of the generated image cannot exceed 2G, that is, width * height < = 2048*1024 *1024
Grammar	ZV_CALGENCHESS (img, rows, cols, blockSize) img: ZVOBJECT type, output parameter, generated calibration plate grayscale image rows: the number of points in vertical direction, >0 cols: the number of points in horizontal direction, >0 blockSize: the size of black and white blocks in chess board, > 0, the unit of pixel
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_CALGENCHESS (img, 7, 7, 30) 'generate the chess board calibration image

13.4.3. ZV_CALGETSCAPTS – Generate Center Coordinate of Solid Circle Calibration Plate

Type	Calibration
Description	Calculate the center coordinates of the solid circle array calibration plate image. The calibration plate image requires at

	least nine points, and the output circle center coordinates are irregular.
Grammar	ZV_CALGETSCAPTS(img,ppts,thresh,polar,minArea,maxArea) img: ZVOBJECT type, single-channel image of calibration board ppts: ZVOBJECT type, output parameters, calculated point coordinates, matrix type N rows and 2 columns thresh: threshold for extracting dots polar: dot polarity, 0-black, 1-white minArea: search the minimum pixel area range of dots maxArea: search the maximum pixel area range of dots
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, ppts ZV_READIMAGE(img,"test.png",0) 'read the image in the original image format ZV_CALGETSCAPTS(img,ppts,128,0,500,5000) 'use 128 threshold to segment the image, search for black dots with a pixel area within the range of [500, 5000], extract the position coordinates of each dot, generate a matrix of N rows and 2 columns and store it in ppts

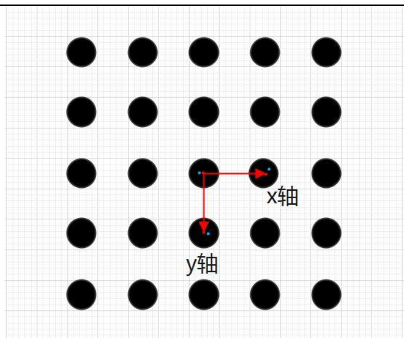
13.4.4. ZV_CALGENCHESSPTS – Get the Corner Point Coordinates of the Checkerboard Calibration Plate

Type	Calibration
Description	Calculate the corner point coordinates of the checkerboard calibration plate. The corner point requires at least nine points, and the output corner point coordinates are irregular.
Grammar	ZV_CALGETCHESSPTS (img,ppts) img: ZVOBJECT type, single-channel image of calibration board

	ppts: ZVOBJECT type, output parameters, calculated point coordinates, matrix type N rows and 2 columns
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, ppts ZV_READIMAGE(img,"test.png",0) 'read the image in the original image format ZV_CALGETSHESSPTS(img,ppts) 'obtain corner point coordinates of chess calibration board, generate a matrix of N rows and 2 columns and store it in ppts

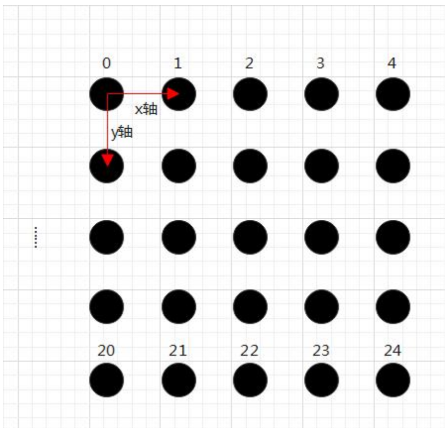
13.4.5. ZV_CALGETBASE – Get Base Coordinate System

Type	Calibration
Description	Select a reference coordinate system (consisting of 3 points) from the input point set. The selection method is to match the points in the input point set according to the input coordinate system points, the origin point is to the origin point, the x-axis point is to the x-axis point, and the y-axis point is to the y-axis points, and finally select the 3 points closest to these 3 points as the output reference coordinate system. It is recommended that the origin, x-axis point, and y-axis point are three adjacent points that constitute the rectangular coordinate system.
Grammar	ZV_CALGETBASE (pptsIn, baseIn, baseOut) pptsIn: ZVOBJECT type, input pixel coordinate point set, single-channel nx2 matrix baseIn: ZVOBJECT type, input coordinate system, single-channel 3x2 matrix, respectively the origin, x-axis point, and y-axis point, all > 0 baseOut: ZVOBJECT type, output base coordinate system, single-channel 3x2 matrix, respectively the origin, x-axis point, and y-axis point like below image:

	 blue are 3 input points, read point means selected output base coordinate system.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, ppts, baseIn, baseOut ZV_READIMAGE(img,"test.jpg",0) 'read the image in the original image format ZV_CALGETSCAPTS(img,ppts,128,0,500,5000) 'get the coordinates of the center of the circle TABLE(0, 361, 362, 482, 362, 361, 482) 'save the input coordinate system data ZV_MATGENDATA(baseIn,3,2,0) 'generate coordinate system matrix ZV_CALGETBASE(ppts,baseIn,baseOut) 'get the base coordinate system

13.4.6. ZV_CALGETPTSMAP – Calculate Map Point Pair of Pixel Coordinate And Word Coordinate

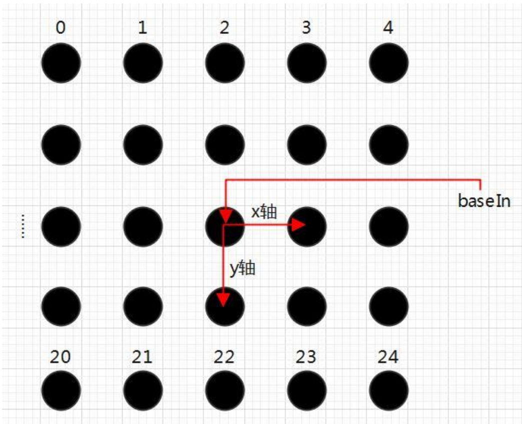
Type	Calibration
Description	Calculate the mapping point pair of pixel coordinates and world coordinates based on the actual distance between two adjacent points of the input pixel coordinates, and output the sorted pixel coordinates and world coordinates. This operator implements two functions: First, it sorts and outputs the pixel coordinates and world coordinates according to the world coordinate system selected

<table>
<tr>
<td></td>
<td>by default (usually 3 points in the upper left corner or 3 points in the lower left corner are selected to form the world coordinate system according to the actual situation of the calibration board), sorting method: sort according to the world coordinate value y from small to large, x from small to large.
Second, it corresponds one-to-one between pixel coordinates and world coordinates. The value at the world coordinate origin in wpts is (0,0). For size measurement or area measurement, only the relative distance of the target needs to be measured, so there is no need to know the real world coordinate origin and can be used directly for calibration. When the machine takes absolute machine coordinates, it needs to modify each world coordinate value corresponding to the pixel in wpts and then calibrate it.</td>
</tr>
<tr>
<td>Grammar</td>
<td>ZV_CALGETPTSMAP(pptsIn,ppts,wpts,dis)
pptsIn: ZVOBJECT type, input parameters, input pixel coordinates, matrix type N rows and 2 columns
ppts: ZVOBJECT type, output parameters, sorted pixel coordinates, matrix type N rows and 2 columns
wpts: ZVOBJECT type, output parameters, sorted world coordinates, matrix type N rows and 2 columns
dis: input parameter, the actual distance between two adjacent points horizontally or vertically, dis is a numerical value, the unit can be (millimeters mm), (centimeters cm), (decimeters dm), (meters m), > 0
the sorting method of outputs:

</td>
</tr>
</table>

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, ppts, ppts_out, wpts ZV_READIMAGE(img,"test.jpg",0) 'read the image in the original image format ZV_CALGETSCAPTS(img,ppts,128,0,500,5000) 'get the coordinates of the center of the circle ZV_CALGETPTSMAP(ppts,ppts_out,wpts,10) 'extract the corresponding pixel coordinates and world coordinates from the input point set ppts based on the actual distance between two adjacent world coordinate points, and stores them in ppts_out and wpts respectively.

13.4.7. ZV_CALGETPTSMAPBASE – Calculate Map Point Pair of Pixel Coordinate And Word Coordinate

Type	Calibration
Description	The function is similar to the ZV_CALGETPTSMAP instruction. The difference is that the world coordinate system used by the ZV_CALGETPTSMAP instruction is at the upper left corner or lower left corner of the solid circle feature point, while the coordinate system used by the ZV_CALGETPTSMAPBASE instruction is the coordinate system input by the instruction.
Grammar	ZV_CALGETPTSMAPBASE(pptsIn,baseIn,ppts,wpts,dis) pptsIn: ZVOBJECT type, input pixel coordinates, matrix type N rows and 2 columns baseIn: ZVOBJECT type, input base coordinate system, single-channel 3x2 matrix, respectively the origin, x-axis point, and y-axis point, all > 0 ppts: ZVOBJECT type, output parameters, sorted pixel coordinates, matrix type N rows and 2 columns wpts: ZVOBJECT type, output parameters, sorted world coordinates, matrix type N rows and 2 columns dis: the actual distance between two adjacent points

	horizontally or vertically, dis is a numerical value, the unit can be (millimeters mm), (centimeters cm), (decimeters dm), (meters m), greater than 0 the sorting method of outputs: 
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, ppts, basIn, baseOut, pptsOut, wpts ZV_READIMAGE(img,"test.jpg",0) 'read the image in the original image format ZV_CALGETSCAPTS(img,ppts,128,0,500,5000) 'get the coordinates of the center of the circle TABLE(0, 361, 362, 482, 362, 361, 482) 'save the input coordinate system data ZV_MATGENDATA(baseIn,3,2,0) 'generate coordinate system matrix ZV_CALGETBASE(ppts,baseIn,baseOut) 'get the base coordinate system ZV_CALGETPTSMAPBASE(ppts,baseOut,pptsOut,wpts,10) 'extract the corresponding pixel coordinates and world coordinates from the input point set ppts based on the actual distance between two adjacent world coordinate points, and store them in pptsOut and wpts respectively.

13.4.8. ZV_CALCAM -- Calibration

Type	Calibration
Description	Calibrate the camera based on pixel coordinates and world coordinates. After calibration, ensure that the relative height of the camera plane and the photographing plane remains unchanged. If the relative height changes, recalibration is required. Note: 1 and 2 in the calibration type are non-linear calibration methods. In this case, the image coordinate system is uneven, that is, the scaling, rotation, etc. at different positions may be different. Therefore, absolute coordinates must be used for coordinate conversion, relative coordinates can't be used.
Grammar	ZV_CALCAM (ppts,wpts,param,width,height,type) ppts: ZVOBJECT type, input parameters, pixel coordinates, matrix type, N rows and 2 columns, N is ≥ 9 wpts: ZVOBJECT type, input parameters, world coordinates, matrix type, N rows and 2 columns, N is ≥ 9 param: ZVOBJECT type, output parameter, generated calibration coefficient width: input parameter, image width when pixel coordinates are obtained height: input parameter, image height when pixel coordinates are obtained type: type of camera calibration. 0 - linear coordinate system calibration, which is a calibration of the coordinate conversion relationship between two coordinate systems. This method can be used when the camera plane is perpendicular to the photographing plane and the lens has no distortion. 1 - nonlinear coordinate system calibration, which is a calibration of the coordinate conversion relationship between two coordinate systems. This method can be used when the camera plane is not perpendicular to the photographing plane and the lens has no distortion.

	2 - camera full parameter calibration, due to the complexity of lens design and craftsmanship and other factors, the actual lens imaging system produces so-called lens distortion, such as radial distortion, tangential distortion, etc. The camera calibration process is to determine the geometric model and optical parameters of the camera. This method can be used when the camera plane is not perpendicular to the photographing plane and there is lens distortion.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM w,h ZVOBJECT img, ppts, param, wpts ZV_READIMAGE(img,"test.jpg",0) 'read the image in the original image format ZV_CALGETSCAPTS(img,ppts,128,0,500,5000) 'get the coordinates of the center of the circle TABLE(0, 361, 362, 482, 362, 361, 482) 'save the input coordinate system data ZV_MATGENDATA(wpts,3,2,0) 'generate coordinate system matrix w = ZV_IMGWIDTH(img) 'get the width of the image h = ZV_IMGHEIGHT(img) 'get the height of the image ZV_CALCAM(ppts,wpts,param,w,h,2) 'use type 2 to perform camera calibration on the pixel coordinates extracted from the image with a width of 640 and a height of 480, combined with the world coordinates, and the coefficients after calibration are stored in param

13.4.9. ZV_CALUNDISTORTPARAM – Get Undistort Parameters

Type	Calibration

Description	Obtain new camera calibration parameters with perspective distortion or radial distortion + perspective distortion removed. The new calibration parameters are used for the distortion-corrected image and convert pixel coordinates into world coordinates. Alias: ZV_CALUNDISTORTCAMPRA
Grammar	ZV_CALUNDISTORTPARAM (param, newParam) param: ZVOBJECT type, input parameters, calibration parameters newParam: ZVOBJECT type, output parameters, calibration parameters
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM w,h ZVOBJECT img, ppts, param, wpts, newParam ZV_READIMAGE(img,"test.jpg",0) 'read the image in the original image format ZV_CALGETSCAPTS(img,ppts,128,0,500,5000) 'get the coordinates of the center of the circle TABLE(0, 361, 362, 482, 362, 361, 482) 'save the input coordinate system data ZV_MATGENDATA(wpts,3,2,0) 'generate coordinate system matrix w = ZV_IMGWIDTH(img) 'get the width of the image h = ZV_IMGHEIGHT(img) 'get the height of the image ZV_CALCAM(ppts,wpts,param,w,h,2) ‘calibrate ZV_CALUNDISTORTPARAM (param, newParam) ‘get new calibration parameters

13.4.10. ZV_CALDECOMPOSE – Calibration Parameters Decomposition

Type	Calibration
Description	Obtain internal parameters and external parameters of

	calibration parameters and radial distortion coefficient.
Grammar	ZV_CALDECOMPOSE(param,interParam,outParam,tabId) param: ZVOBJECT type, input parameters, calibration parameters interParam: ZVOBJECT type, internal parameters, output parameters, 3x3 64F matrix outParam: ZVOBJECT type, external parameters, output parameter, 4x4 64F matrix tabId: TABLE id used to store radial distortion coefficient k1, output
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

13.4.11. ZV_CALGETPIXSCALE – Get Pixel Scale

Type	Calibration
Description	Get the pixel scale from the calibration parameters. The pixel scale represents the actual size represented by the unit pixel. The actual size unit is consistent with the world coordinate unit used during calibration. It is usually more convenient to use the pixel scale multiplied by the pixel length (the pixel length obtained on the image) to obtain the actual length. If the image is not corrected, the calculated actual length will not be accurate enough.
Grammar	ZV_CALGETPIXSCALE (param, tabId) param: ZVOBJECT type, input parameters, calibration parameters tabId: used to store TABLE id of pixel scale
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM w,h ZVOBJECT img, ppts, param, wpts ZV_READIMAGE(img,"test.jpg",0) 'read the image in the original image format

	ZV_CALGETSCAPTS(img,ppts,128,0,500,5000) 'get the coordinates of the center of the circle TABLE(0, 361, 362, 482, 362, 361, 482) 'save the input coordinate system data ZV_MATGENDATA(wpts,3,2,0) 'generate coordinate system matrix w = ZV_IMGWIDTH(img) 'get the width of the image h = ZV_IMGHEIGHT(img) 'get the height of the image ZV_CALCAM(ppts,wpts,param,w,h,2) 'calibrate ZV_CALGETPIXSCALE (param, 0) 'get pixel scale and save result into TABLE (0)

13.4.12. ZV_CALERROR – Calibrate Error

Type	Calibration
Description	Used to calibrate pixel coordinate and world coordinate of camera, and use calibration coefficient to evaluate pixel errors calibrated by camera.
Grammar	ZV_CALERROR (param, ppts, wpts, tabId) param: ZVOBJECT type, calibration coefficient ppts: ZVOBJECT type, matrix type, pixel coordinates wpts: ZVOBJECT type, matrix type, world coordinates tabId: TABLE index, output parameters, calibration error, in order, average error, minimum error, maximum error. Among them, if the average error is less than 0.5, it is considered excellent, 0.5--1 is good, 1--1.5 is average, and above 1.5, it is recommended to recalibrate.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM w,h ZVOBJECT img, ppts, param, wpts ZV_READIMAGE(img,"test.jpg",0) 'read the image in the original image format ZV_CALGETSCAPTS(img,ppts,128,0,500,5000)

	'get the coordinates of the center of the circle TABLE(0, 361, 362, 482, 362, 361, 482) 'save the input coordinate system data ZV_MATGENDATA(wpts,3,2,0) 'generate coordinate system matrix w = ZV_IMGWIDTH(img) 'get the width of the image h = ZV_IMGHEIGHT(img) 'get the height of the image ZV_CALCAM(ppts,wpts,param,w,h,2) 'calibrate ZV_CALERROR (param, ppts, wpts, 0) 'use pixel coordinate "ppts" and word coordinate "wpts" that are used to calibrate camera and use calibrated calibration coefficient "param" to evaluate pixel error calibrated by pixel, and then save result into TABLE (0).

13.4.13. ZV_CALGETERROR – Calibrate Error

Type	Calibration
Description	Get calibration error.
Grammar	ZV_CALGETERROR (param, tabId) param: ZVOBJECT type, calibration coefficient tabId: TABLE index, output parameters, calibration error, in order, average error, minimum error, maximum error. Among them, if the average error is less than 0.5, it is considered excellent, 0.5--1 is good, 1--1.5 is average, and above 1.5, it is recommended to recalibrate.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM w,h ZVOBJECT img, ppts, param, wpts ZV_READIMAGE(img,"test.jpg",0) 'read the image in the original image format ZV_CALGETSCAPTS(img,ppts,128,0,500,5000) 'get the coordinates of the center of the circle

	TABLE(0, 361, 362, 482, 362, 361, 482)
	    'save the input coordinate system data
	ZV_MATGENDATA(wpts,3,2,0)
	    'generate coordinate system matrix
	w = ZV_IMGWIDTH(img) 'get the width of the image
	h = ZV_IMGHEIGHT(img) 'get the height of the image
	ZV_CALCAM(ppts,wpts,param,w,h,2) ‘calibrate
	ZV_CALGETERROR (param, 0)
	    ‘get calibration error and save it into TABLE (0)

13.4.14. ZV_CALTRANSI – From World to Pixel Coordinate

Type	Calibration
Description	Use calibration coefficient to convert world coordinate to pixel coordinate.

Grammar	ZV_CALTRANSI (param, pwx, pwy, tabId)
	  param: ZVOBJECT type, calibration coefficient
	  pwx: world coordinate x
	  pwy: world coordinate y
	  tabId: TABLE index, output parameters, they are pixel coordinate x and pixel coordinate y.

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

Example	DIM w,h
	ZVOBJECT img, ppts, param, wpts, pwx, pwy
	ZV_READIMAGE(img,"test.jpg",0)
	    'read the image in the original image format
	ZV_CALGETSCAPTS(img,ppts,128,0,500,5000)
	    'get the coordinates of the center of the circle
	TABLE(0, 361, 362, 482, 362, 361, 482)
	    'save the input coordinate system data
	ZV_MATGENDATA(wpts,3,2,0)
	    'generate coordinate system matrix
	w = ZV_IMGWIDTH(img) 'get the width of the image

	h = ZV_IMGHEIGHT(img) 'get the height of the image ZV_CALCAM(ppts,wpts,param,w,h,2) ‘calibrate ZV_CALTRANSI (param, 10, 10, 0) ‘convert world coordinate to pixel coordinate, and save pixel coordinate x, y into TABLE (0)

13.4.15. ZV_CALTRANSW – From Pixel to World

Type	Calibration
Description	Use calibration coefficient to convert pixel coordinate to world coordinate.
Grammar	ZV_CALTRANSW (param, pwx, pwy, tabId) param: ZVOBJECT type, calibration coefficient pwx: world coordinate x pwy: world coordinate y tabId: TABLE index, output parameters, they are world coordinate x and world coordinate y.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM w,h ZVOBJECT img, ppts, param, wpts ZV_READIMAGE(img,"test.jpg",0) 'read the image in the original image format ZV_CALGETSCAPTS(img,ppts,128,0,500,5000) 'get the coordinates of the center of the circle TABLE(0, 361, 362, 482, 362, 361, 482) 'save the input coordinate system data ZV_MATGENDATA(wpts,3,2,0) 'generate coordinate system matrix w = ZV_IMGWIDTH(img) 'get the width of the image h = ZV_IMGHEIGHT(img) 'get the height of the image ZV_CALCAM(ppts,wpts,param,w,h,2) ‘calibrate ZV_CALTRANSW (param, 10, 10, 0) ‘convert pixel coordinate to world coordinate, and save

	world coordinate x, y into TABLE (0)

13.4.16. ZV_CALTRANSWCONTS – From Pixel to World

Type	Calibration
Description	Use calibration coefficient to convert pixel coordinate of contour point in contour or contour list to world coordinate.
Grammar	ZV_CALTRANSWCONT (param, src, dst) param: ZVOBJECT type, calibration coefficient src: ZVOBJECT type, input contour or contour list dst: ZVOBJECT type, output contour or contour list
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM w,h ZVOBJECT img, imgBw, contImg, ppts, wpts, param, contList, contListDst ZV_READIMAGE(img,"test.jpg",0) 'read the image in the original image format ZV_CALGETSCAPTS(img,ppts,128,0,500,5000) 'get the coordinates of the center of the circle TABLE(0, 361, 362, 482, 362, 361, 482) 'save the input coordinate system data ZV_MATGENDATA(wpts,3,2,0) 'generate coordinate system matrix w = ZV_IMGWIDTH(img) 'get the width of the image h = ZV_IMGHEIGHT(img) 'get the height of the image ZV_CALCAM(ppts,wpts,param,w,h,2) ‘calibrate ZV_READIMAGE(contImg,"test.jpg",0) 'read the image in the original image format ZV_THRESH(contImg,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list ZV_CALTRANSWCONTS (param, contList, contListDst) 'convert contour point pixel coordinates to world

	coordinates

13.4.17. ZV_CALUNDISTORT – Distort Image Correction

Type	Calibration
Description	Use the calibration coefficient to correct the distorted image, correct the image to the world coordinate system plane z=0, it supports perspective correction and radial distortion + perspective correction, the corrected image is perpendicular to the camera plane.
Grammar	ZV_CALUNDISTORT (param, src, dst) param: ZVOBJECT type, calibration coefficient, the calibration type is 1/2, if it is 0, original image is output src: ZVOBJECT type, input distort image dst: ZVOBJECT type, output corrected image
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	DIM w,h ZVOBJECT img, imgSrc, imgDst, ppts, param, wpts ZV_READIMAGE(img,"test.jpg",0) 'read the image in the original image format ZV_CALGETSCAPTS(img,ppts,128,0,500,5000) 'get the coordinates of the center of the circle TABLE(0, 361, 362, 482, 362, 361, 482) 'save the input coordinate system data ZV_MATGENDATA(wpts,3,2,0) 'generate coordinate system matrix w = ZV_IMGWIDTH(img) 'get the width of the image h = ZV_IMGHEIGHT(img) 'get the height of the image ZV_CALCAM(ppts,wpts,param,w,h,2) ‘calibrate ZV_READIMAGE(imgSrc,"test.jpg",0) 'read the image in the original image format ZV_CALUNDISTORT (param, imgSrc, imgDst) ‘correct distort image

Chapter XIV Defect

14.1. Measurement Type Defect

14.1.1. ZV_DEFCREATEMRCONT2 –Create Contour Pair Defect Detection Handle

Type	Measurement type defect
Description	Create a contour pair measurement type defect handle, also called a defect detector, for detecting contour pair defects. The contour pair area to be detected is specified by a standard contour describing its centerline. Absolute threshold mode is used by default.
Grammar	ZV_DEFCREATEMRCONT2(cont,detector[,targetSize=30]) cont: contour to standard center contour, ZVOBJECT type detector: generated defect detector handle, ZVOBJECT type targetSize: ideal width of contour, the unit is pixel, default value 30, range greater than or equal to 5
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

14.1.2. ZV_DEFSETPARAMMR – Set Measurement Type Defect Detection Parameters

Type	Measurement type defect
Description	Set detection parameters that are used to measure defects, parameters are different according to different measurement types.

<table>
<tr><td>Grammar</td><td>ZV_DEFSETPARAMMR (detector,paramType,paramValue)
detector: defect detector, ZVOBJECT type, input also is output
paraType: parameter type to be set:
contour pair measurement type defect parameters
<table>
<tr><td>Value</td><td>Type</td><td>Default value</td><td>Description</td></tr>
<tr><td>1</td><td>Ideal width</td><td>30</td><td>Ideal width of edge pair, ≥ 5</td></tr>
<tr><td>2</td><td>Width error threshold</td><td>15</td><td>Allowable error of edge pair width, ≥ 0</td></tr>
<tr><td>3</td><td>Position offset threshold</td><td>20</td><td>Allowable fluctuation range of edge pair width, ≥ 0</td></tr>
<tr><td>21</td><td>Filter size</td><td>7</td><td>Measured filter size, 1-31</td></tr>
<tr><td>22</td><td>Edge threshold</td><td>25</td><td>Measured edge gradient threshold, 1-255</td></tr>
<tr><td>23</td><td>Polarity</td><td>0</td><td>Color polarity of edge pair region, 0: black, 1: white</td></tr>
</table>
paramValue: parameter value that is set, it must be in the range.</td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
</table>

14.1.3. ZV_DEFGETPARAMMAR – Get Measurement Type Defect Detection Parameters

Type	Measurement type defect
Description	Get measurement type defect detection parameters.
Grammar	ZV_DEFGETPARAMMAR (detector, paramType, tabId) detector: defect detector, ZVOBJECT type

	paramType: parameter type to be obtained, refer to ZV_DEFSETPARAMMR tabId: obtained parameter value, output parameters, TABLE index
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

14.1.4. ZV_DEFAPPLYMR – Detect Measurement Type Defects

Type	Measurement type defect
Description	Execute detection of measurement type defects, but the specific is determined by measurement handle "detector", so detect corresponding defects according to description of creating command handle.
Grammar	ZV_DEFAPPLYMR(detector,src,def) detector: measurement type defect detector, ZVOBJECT type src: input single-channel image def: defect result, ZVOBJECT type, output
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

14.2. Result Obtaining

14.2.1. ZV_DEFGETOBJECT – Obtain ZVOBJECT Object in Defects Result

Type	Defect result
Description	Get ZVOBJECT object from defect result.

<table>
<tr><td>Grammar</td><td>ZV_DEFGETOBJECT (def, obj, type)
def: defect result, obtained by defect detection type algorithm
obj: obtained ZVOBJECT type data of defect result.
type: type to obtain ZVOBJECT object:
<table>
<tr><td>Value</td><td>rst type</td><td>obj type</td><td>Description</td></tr>
<tr><td>2</td><td rowspan="2">Contour pair defect</td><td rowspan="2">Contour list</td><td>Defect contour list, each contour means one defect, that is external contour of defect region.</td></tr>
<tr><td>5</td><td>Standard contour list of defect, each contour means one defect, that is standard contour corresponding to defect.</td></tr>
</table>
</td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
</table>

14.2.2. ZV_DEFGETVALUE – Obtain Value Parameters in Defect Result

Type	Defect result
Description	Get value parameters in defect result.

<table>
<tr><td>Grammar</td><td>ZV_DEFGETVALUE (rst, type, maxNum, tabId)
rst: defect result, obtained by defect detection type algorithm
type: defect value parameter type
<table>
<tr><td>Value</td><td>rst type</td><td>Description</td></tr>
<tr><td>1</td><td>All types</td><td>The number of defects, one parameter</td></tr>
<tr><td>12</td><td>Contour pair defect</td><td>The defect type of the contour pair defect, multiple defects are arranged in sequence, the number can be obtained according to the number of defects or the length of the defect contour list, the value range is 1 - disconnection, 2 - spacing is too small, 3 - spacing is too large, 4 - position deviation is large</td></tr>
</table>maxNum: the max valid number of TABLE
tabId: obtained value, output parameter, TABLE index</td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
</table>

14.2.3. ZV_DEFGETINFO – Obtain Middle Information of Defect Detection

Type	Defect result
Description	Get ZVOBJECT type middle information in defect detection, which can be used to help showing.

<table>
<tr><td>Grammar</td><td>ZV_DEFGETINFO (def, obj, type)
def: defect result, obtained by defect detection type algorithm
obj: obtained middle ZVOBJECT object, output parameters.
type: type of middle object information
<table>
<tr><td>Value</td><td>rst type</td><td>obj type</td><td>Description</td></tr>
<tr><td>31</td><td rowspan="2">Contour pair defect</td><td rowspan="2">Contour list</td><td>The first measured contour of contour pair defect detection, the contour composed of the first measurement points along the measurement scanning direction.</td></tr>
<tr><td>32</td><td>The second measured contour of contour pair defect detection, the contour composed of the second measurement points along the measurement scanning direction.</td></tr>
</table></td></tr>
<tr><td>Controller</td><td>It is valid in controllers that support ZV function or they belong to 5XX series or above.</td></tr>
</table>

Chapter XV Drawing

15.1. ZV_COLOR – Generate Color

Type	Drawing
Description	Use r, g, b to generate color value.
Grammar	value = ZV_COLOR (r, g, b) r: red color value, [0, 255] g: green color value, [0, 255] b: blue color value, [0, 255] returned value: color value
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	value = ZV_COLOR (255, 0, 0) 'generate red, select which color you want according to RGB chromatogram, usually it is used as parameters.

15.2. ZV_POINTS – Point Set

Type	Drawing
Description	Draw point set.
Grammar	ZV_POINTS (img, pts, color) img: ZVOBJECT type, target image to be drawn pts: ZVOBJECT type, n x 2 matrix color: line color, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.

15.3. ZV_LINE – Straight Line

Type	Drawing
Description	Draw straight line.
Grammar	ZV_LINE(img,x1,y1,x2,y2,color) img: ZVOBJECT type, target image to be drawn x1: x coordinate of line's point 1 y1: y coordinate of line's point 1 x2: x coordinate of line's point 2 y2: y coordinate of line's point 2 color: color of line, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img,"test.jpg",0) 'read the image in the original image format ZV_LINE(img,0,0,5,5,ZV_COLOR(255,0,0)) 'draw a red straight line in the target image

15.4. ZV_CONTOUR – Contour

Type	Drawing
Description	Draw contour
Grammar	ZV_CONTOUR (img, cont, color) img: ZVOBJECT type, target image to be drawn cont: ZVOBJECT type, contour color: color of contour, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255.
Controller	It is valid in controllers that support ZV function or they belong

	to 5XX series or above.
Example	DIM count ZVOBJECT img, gray, dst, imgBw, contList, contSrc ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(imgBw,contList,1,0) 'save all the found contours into the contour list ZV_IMGCOPY(img,gray) 'copy image ZV_IMGSETCONST(gray,0) 'constant fill image ZV_GRAYTORGB(gray,dst) 'convert grayscale image to color image count = ZV_LISTCOUNT(contList) 'get the number of contour lists FOR i = 0 TO count-1 ZV_LISTGET(contList, contSrc,i) 'get a certain contour ZV_CONTOUR(dst,contSrc,ZV_COLOR(0,255,0)) 'draw the outline in green NEXT

15.5. ZV_CONLIST – Contour List

Type	Drawing
Description	Draw contour list
Grammar	ZV_CONTLIST(img, contlist, color, autoColor) img: ZVOBJECT type, target image to be drawn cont: ZVOBJECT type, contour list color: color of contour list, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255. autoColor: whether sets color automatically, 3 means different colors are set automatically, color will not be used, for black and white image, only white will be drawn.

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, gray, imgBw, dst, contList ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_THRESH(img,imgBw,200,255) 'image binarization ZV_CONTGEN(img, imgBw,contList,1,0) 'save all the found contours into the contour list ZV_IMGCOPY(img,gray) 'copy image ZV_IMGSETCONST(gray,0) 'constant fill image ZV_GRAYTORGB(gray,dst) 'convert grayscale image to color image ZV_CONTLIST (dst, contList, ZV_COLOR (0, 255, 0), 0) 'draw the contour

15.6. ZV_RECT – Rectangle

Type	Drawing
Description	Draw rectangle.
Grammar	ZV_RECT (img, x, y, w, h, color) img: ZVOBJECT type, target image to be drawn x: left corner x coordinate of rectangle y: left corner y coordinate of rectangle w: rectangle width h: rectangle height color: color of rectangle, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format

	ZV_RECT (img, 200, 200, 100, 100, ZV_COLOR (255, 0, 0) 'draw red rectangle at position 200, 200 in target image

15.7. ZV_RECT2 – Rotate Rectangle

Type	Drawing
Description	Draw rotate rectangle.
Grammar	ZV_RECT2 (img, cx, cy, w, h, angle, color) img: ZVOBJECT type, target image to be drawn cx: rotate rectangle center x coordinate cy: rotate rectangle center y coordinate w: rotate rectangle width h: rotate rectangle height angle: rectangle rotate angle color: color of rotate rectangle, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_RECT2 (img, 20, 20, 100, 100, 30, ZV_COLOR (255, 0, 0) 'draw rotate rectangle in target image

15.8. ZV_CIRCLE – Circle

Type	Drawing
Description	Draw circle.
Grammar	ZV_CIRCLE (img, cx, cy, r, color) img: ZVOBJECT type, target image to be drawn cx: circle center x coordinate

	cy: circle center y coordinate r: circle's radius color: color of circle, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_CIRCLE (img, 30, 30, 10, ZV_COLOR (255, 0, 0) 'draw one red circle with a radius of 10 at position 30, 30 of image "img".

15.9. ZV_ELLIPSE -- Ellipse

Type	Drawing
Description	Draw ellipse.
Grammar	ZV_ELLIPSE (img, cx, cy, xr, yr, angle, color) img: ZVOBJECT type, target image to be drawn cx: ellipse center x coordinate cy: ellipse center y coordinate xr: major-axis length of ellipse in the x direction yr: minor-axis length of ellipse in the y direction angle: rotate angle of ellipse color: color of ellipse, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img, "test.jpg",0)

	'read the image in the original image format ZV_ELLIPSE (img, 10, 10, 5, 5, 20, ZV_COLOR (255, 0, 0) 'draw the ellipse in the target image

15.10. ZV_ELLIPSEARC – Ellipse Arc

Type	Drawing
Description	Draw the ellipse arc from starting angle to end angle in clockwise, if the starting angle is less than end angle, then reverse the direction (from end angle to starting angle).
Grammar	ZV_ELLIPSEARC (img, cx, cy, xr, yr, angle, startAngle, endAngle, color) img: ZVOBJECT type, target image to be drawn cx: ellipse arc center x coordinate cy: ellipse arc center y coordinate xr: major-axis length of ellipse arc in the x direction yr: minor-axis length of ellipse arc in the y direction angle: rotate angle of ellipse arc, in clockwise, unit – degree startAngle: starting angle of ellipse arc, in clockwise, unit – degree endAngle: end angle of ellipse arc, in clockwise, unit – degree color: color of ellipse arc, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_ELLIPSEARC (img, 100, 100, 200, 200, 60, 120, 160, ZV_COLOR (255, 0, 0) 'draw the ellipse arc in the target image

15.11. ZV_ELLIPSEARCBYPTS – Ellipse Arc

Type	Drawing
Description	Draw the ellipse arc from starting point to end point in given direction.
Grammar	ZV_ELLIPSEARCBYPTS (img, cx, cy, xr, yr, angle, stx, sty, endx, endy, dir, color) img: ZVOBJECT type, target image to be drawn cx: ellipse arc center x coordinate cy: ellipse arc center y coordinate xr: major-axis length of ellipse arc in the x direction yr: minor-axis length of ellipse arc in the y direction angle: rotate angle of ellipse arc, in clockwise, unit – degree stx: x coordinate of ellipse arc's starting point sty: y coordinate of ellipse arc's starting point endx: x coordinate of ellipse arc's end point endy: y coordinate of ellipse arc's end point dir: the direction, 1: anticlockwise, -1: clockwise color: color of ellipse arc, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255. Note: starting point and end point are used to help calculate starting and end angles of ellipse arc, but the real angles depend on ellipse center and major and minor axes.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_ELLIPSEARCBYPTS (img, 320, 240, 80, 60, 0, 400, 240, 320, 300,-1,ZV_COLOR(255,0,0) 'in the target image, draw from start point (400, 240) top end point (320, 300) in clockwise

15.12. ZV_POLYGON – Polygon

Type	Drawing
Description	Draw polygon.
Grammar	ZV_POLYGON (img, pts, idClosed, color) img: ZVOBJECT type, target image to be drawn pts: ZVOBJECT type, polygon point sequence to be drawn, rectangle type isClosed: whether is closed, 0 – open, 1 - closed color: color, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format TABLE (0, 0, 1, 2, 3, 4, 5, 6, 7) 'save data into TABLE (0) ZV_MATGENDATA (pts, 4, 2, 0) ZV_POLYGON (img, pts, 0, ZV_COLOR (255, 0, 0) 'draw the polygon in the target image

15.13. ZV_ARROW – Arrow

Type	Drawing
Description	Draw arrow.
Grammar	ZV_ARROW (img, x1, y1, x2, y2, size, color) img: ZVOBJECT type, target image to be drawn x1: x coordinate of arrow's starting point y1: y coordinate of arrow's starting point x2: x coordinate of arrow's end point y2: y coordinate of arrow's end point size: size of arrow

	color: color, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_ARROW (img, 10, 10, 50, 50, 10, ZV_COLOR (255, 0, 0) 'draw the green arrow in the target image

15.14. ZV_MARKER – Mark

Type	Drawing
Description	Draw markable shaped, like, star, triangle, cross, etc.
Grammar	ZV_MARKER (img, x, y, type, size, color) img: ZVOBJECT type, target image to be drawn x: x coordinate of drawing position y: y coordinate of drawing position type: mark type to be drawn <table><tr><td>type</td><td>Description</td></tr><tr><td>0</td><td>Cross</td></tr><tr><td>1</td><td>Oblique cross (X)</td></tr><tr><td>2</td><td>Star</td></tr><tr><td>3</td><td>Diamond shape</td></tr><tr><td>4</td><td>Square</td></tr><tr><td>5</td><td>Triangle</td></tr><tr><td>6</td><td>Inverted triangle</td></tr></table>size: mark size to be drawn color: color, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255.

Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_MARKER (img, 100, 100, 0, 10, ZV_COLOR (255, 0, 0) 'draw one cross

15.15. ZV_TEXT -- Text

Type	Drawing
Description	Output character string "str" in image "img", Chinese is supported, and font information is set by system parameters.
Grammar	ZV_TEXT (img, str, x, y, size, color) img: ZVOBJECT type, target image to be drawn str: character string to be drawn x: x coordinate of drawing position y: y coordinate of drawing position size: font pixel size color: color of text, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_TEXT (img, "hello", 100, 100, 10, ZV_COLOR (255, 0, 0) 'write the text with a pixel of 10 at 100, 100 in "img", the color is white

15.16. ZV_MASK – Mask Image

Type	Drawing
Description	Draw "mask", that is, set mask whose pixel is 0 corresponding to image "img" as 0 (fillFore is 0), or set whose pixel is 255 as 255 (fillFore is 1).
Grammar	ZV_MASK (img, mask, fillFore) img: ZVOBJECT type, target image to be drawn mask: ZVOBJECT type, mask image fillFore: whether fills the foreground color, if it is 1, the foreground color 255 will be filled, otherwise, 0 will be filled.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, mask ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_IMGCOPY (img, mask) 'copy image ZV_IMGSETCONST (mask, 0) 'constant fills image FOR i=10 TO 10 FOR j=10 TO 10 ZV_IMGSETVAL(mask,i,j,0,1) 'modify image's value NEXT NEXT ZV_MASK(img,mask,1) 'draw target image, fill in foreground color

15.17. ZV_REGION – Region

Type	Drawing
Description	Draw the region that is based on run-length coding.
Grammar	ZV_REGION (img, re, type, color) img: ZVOBJECT type, target image to be drawn, single-channel or 3-channel 8U type re: ZVOBJECT type, region of run-length coding type: drawing type, 0 – draw valid part of re, 1 – draw invalid

	part of re, 2 – draw edge part outside re color: color of drawing region, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, re ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_REGENRECT (re, 0, 0, 100, 100) ZV_REGION (img, re, 0, 255) 'in image "img", draw specified region, 255 color value.

15.18. ZV_MEASURER – Measurement Region

Type	Drawing
Description	Draw the measurer.
Grammar	ZV_MEASURER (img, mr, color, subColor) img: ZVOBJECT type, target image to be drawn mr: ZVOBJECT type, the measurer to be drawn. color: main color of measurer subColor: measurer sub-region's color, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, re ZV_READIMAGE(img, "test.jpg",0) 'read the image in the original image format ZV_REGENRECT (mr, 20, 20, 100, 100) 'generate rectangle measurer ZV_MEASURER (img, mr, ZV_COLOR (255, 255, 255), ZV_COLOR

	(255, 255, 255) 'generate measurer in target image

15.19. ZV_DRASHAPEMATCH – Shape Template

Type	Drawing
Description	Draw shape template.
Grammar	ZV_DRASHAPEMATCH (img, mod, matchRst, stats, color1, color2) img: ZVOBJECT type, target image to be drawn mod: ZVOBJECT type, shape template or shape template list matchRst: ZVOBJECT type, matching result, matrix type, n rows 5 columns, one matching target of each row, they are matching score "score", coordinate x, coordinate y, rotate angle "angle", scaling "scale" in order. stats: ZVOBJECT type, template contour point matching state, image type, n rows m columns, each row has one template contour, and matching state of template contour point can be saved in order, 1 – success, 0 – failure, and the number of rows are same as the rows number of matchs. color 1: color of successful matching, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255. color 2: color of fail matching, ZV_COLOR (r, g, b) can be used to generate the image when img is color image, and when img is black and white image, get value [0, 255], for example, black 0, gray 128, white 255.
Controller	It is valid in controllers that support ZV function or they belong to 5XX series or above.
Example	ZVOBJECT img, clrImg, mod, matchImg, rlts, stats ZV_READIMAGE(img, "model.jpg",0) 'read the image in the original image format

	ZV_SHAPECREATE(img,mod,0,360,1,1,50,0,0,0,0) 'create template ZV_READIMAGE (matchImg, "1.png", 0) 'read the image in the original image format ZV_SHAPEFINDST (mod, matchImg, rlts, stats, 90, 1, 0, -1, 3, 9, 0) 'template matching ZV_GRAYTORGB (matchImg, clrImg) 'convert grayscale image to RGB image ZV_DRASHAPEMATCH (clrImg, mod, rlts, stats, ZV_COLOR(0,255,0), ZV_COLOR(255,0,0)) 'draw the template on the color image, and the contour points that match successfully are drawn in green, and the contour points that fail to match are drawn in red.

Chapter XVI Vision Usage Examples

16.1. Coordinate System Calibration

Example 1:

Extract the pixel coordinates of the mark point through the solid circle calibration plate (the world coordinate corresponding to the known mark point), and calibrate the relationship between the pixel and the world coordinate system.

‘calibrate parameter array, they are calibration type, contrast, polarity, minimal area, maximal area in order.

```
GLOBAL DIM d_ca_param(5)
DIM w,h                         'width and height of calibration plate image
w = 640
h = 480
d_ca_param(0) = 0               'calibration type
d_ca_param(1) = 120             'contrast
d_ca_param(2) = 0               'polar-black mark point
d_ca_param(3) = 80              'minimum area
d_ca_param(4) = 20000           'maximum area
ZVOBJECT calImg                 'calibration plate image
ZVOBJECT pptsIn, ppts, wpts     'pixel coordinates and world coordinates
ZVOBJECT ca_param               'calibration parameters
```

```
'extract the pixel coordinates of the mark point on the calibration plate
ZV_CALGETSCAPTS (calImg, pptsIn, d_ca_param(1), d_ca_param(2), d_ca_param(3),
d_ca_param(4))
```

```
'process the world coordinates wpts corresponding to the pixel coordinates ppts
ZV_CALGETPTSMAP (pptsIn, ppts, wpts, 5)
```

```
'calibration
ZV_CALCAM (ppts, wpts, ca_param, w, h, d_ca_param(0))
```

'calculate the calibration error, TABLE(0), TABLE(1), TABLE(2) are the average error, minimum error, and maximum error respectively

ZV_CALERROR(ca_param, ppts, wpts, 0)

'save calibration parameters
ZV_CALWRITE(ca_param, "calib.zvb")

Example 2:

Extract the pixel coordinates of the mark point through the solid circle calibration plate, and use distance of "mark" points and take left upper corner "mark" point as world origin to calculate word coordinates automatically (this method is used for size measurement that doesn't concern about world coordinate system origin), and calibrate the relationship between the pixel and the world coordinate system.

'calibrate parameter array, they are calibration type, contrast, polarity, minimal area, maximal area in order.

```
GLOBAL DIM d_ca_param(6)
DIM w,h                          'width and height of calibration plate image
w = 640
h = 480
d_ca_param(0) = 0              'calibration type
d_ca_param(1) = 120            'contrast
d_ca_param(2) = 0              'polar-black mark point
d_ca_param(3) = 80             'minimum area
d_ca_param(4) = 20000          'maximum area
d_ca_param(5) = 10             'mark points' distance
ZVOBJECT calImg                'calibration plate image
ZVOBJECT pptsIn, ppts, wpts    'pixel coordinates and world coordinates
ZVOBJECT ca_param              'calibration parameters
```

'extract the pixel coordinates of the mark point on the calibration plate
ZV_CALGETSCAPTS (calImg, pptsIn, d_ca_param(1), d_ca_param(2), d_ca_param(3), d_ca_param(4))

'calculate world coordinate wpts according to mark point distance
ZV_CALGETPTSMAP (pptsIn, ppts, wpts, d_ca_param(5))

'calibration

ZV_CALCAM (ppts, wpts, ca_param, w, h, d_ca_param(0))

'calculate the calibration error, TABLE(0), TABLE(1), TABLE(2) are the average error, minimum error, and maximum error respectively
ZV_CALERROR(ca_param, ppts, wpts, 0)

'save calibration parameters
ZV_CALWRITE(ca_param, "calib.zvb")

Example 3:

Use 9-point calibration method (get pixel coordinate of "mark" through positioning 9 times, and read machine world coordinates 9 times at the same time) to calibrate the relationship between the pixel and the world coordinate system, the method is to control machine to move 9 times, then moves the field-shaped nine-square grid in rows first and then in rows.

```
'TABLE 61-78, pixel coordinate, 9 coordinates, 18 data, x, y, x, y....
'TABLE 81-98, world coordinate, 9 coordinates, 18 data, x, y, x, y....

GLOBAL DIM d_ca_param(7)      'match parameters
DIM w,h                       'width and height of image obtained by camera
d_ca_param(0) = 120           'set left corner x position of nine-square (under machine
                               coordinate system)
d_ca_param(1) = -220          'set left corner y position of nine-square (under machine
                               coordinate system)
d_ca_param(2) = 10            'distance of nine-square (under machine coordinate system)
d_ca_param(3) = 1             'calibration type
d_ca_param(4) = 0             'average error
d_ca_param(5) = 0             'minimal error
d_ca_param(6) = 0             'maximum error
d_ca_param(5) = 10            'mark points' distance

ZVOBJECT grabImg              'camera samples image
ZVOBJECT ppts, wpts           'pixel coordinates and world coordinates
ZVOBJECT ca_param             'calibration parameters
ZVOBJECT s_mod                'created template
BASE (0, 2)                   'set axis x and axis y
```

```
'move nine-square in cycle
DIM cnt
cnt = 0
FOR i = 0 to 2
	FOR j = 0 to 2
		'motion delays 500ms
		MOVE_DELAY (500)

		'move to sample position
		MOVEABS( d_calib_param(0) + j * d_calib_param(2), d_calib_param(1) + i * d_calib_param(2))
		WAIT until IDLE(0) and IDLE(2)

		'send soft trigger signal
		CAM_SETPARAM ("TriggerSoftware", 0)

		'sample one image
		CAM_GET (grabImg, 0)

		'shape matching and positioning mark point
		ZV_SHAPEFIND(s_mod,grabImg,match_rst,d_match_param(0),d_match_param(1),d_match_param(2),d_match_param(3),d_match_param(4),d_match_param(5),d_match_param(6))

		'extract matching result
		ZV_MATGETROW (match_rst, 0, 5, 0)

		'save pixel coordinate into TABLE
		TABLE (61 + cnt*2) = TABLE (1)
		TABLE (61 + cnt*2+1) = TABLE (2)

		'save machine coordinate into TABLE
		TABLE (81 + cnt*2) = DPOS (0)
		TABLE (81 + cnt*2+1) = DPOS (2)
		cnt = cnt + 1
	NEXT
NEXT

'convert pixel and world coordinates in TABLE into matrix
ZV_MATGENDATA(ppts, 9, 2, 61)
```

```
ZV_MATGENDATA(wpts, 9, 2, 81)

'calibration
ZV_CALCAM (ppts, wpts, ca_param, w, h, d_ca_param(3))

'calculate the calibration error, TABLE(0), TABLE(1), TABLE(2) are the average error, minimum error, and maximum error respectively
ZV_CALERROR(ca_param, ppts, wpts, 0)

'save calibration parameters
ZV_CALWRITE(ca_param, "calib.zvb")
```

16.2. Acquisition by Soft Trigger

```
'acquire a single image in soft trigger mode
DIM cam_cnt
ZVOBJECT img
CAM_SCAN("mvision")           'scan Hikvision camera
cam_cnt = CAM_COUNT()         'get the number of scanned cameras
IF (0 = cam_cnt) THEN         'If the number of scanned cameras is 0, return
    PRINT "camera not found"
    RETURN
ENDIF
CAM_SEL(0)                    'select the camera with serial number 0
CAM_SETMODE(0)                'set the trigger mode as soft trigger mode
CAM_START(1)          'enable camera acquisition and specify the number of buffer as 1
CAM_SETPARAM("TriggerSoftware", 0)
                              'use soft trigger command parameters to trigger the
                              camera to take pictures, and take pictures once every time
                              it is triggered.
CAM_GET(img,0)                'get the image with the specified id No 0 in the camera
                              buffer and store it in the img image
```

16.3. Acquisition By External Trigger

```
'acquire a single image in external trigger mode
DIM cam_cnt
ZV0BJECT img
CAM_SCAN("mvision")          'scan Hikvision camera
cam_cnt = CAM_COUNT()      'get the number of scanned cameras
IF (0 = cam_cnt) THEN          'If the number of scanned cameras is 0, return
    PRINT "camera not found"
    RETURN
ENDIF
CAM_SEL(0)                          'select the camera with serial number 0
CAM_SETMODE(0)                 'set the trigger mode as external trigger mode
CAM_START(1)           'enable camera acquisition and specify the number of buffer as 1
MOVE_OP (0, ON)
MOVE_OP (0, OFF)                  'set as falling edge to trigger taking photos, operate OUT
                                    when falling edge, then trigger to take photos
CAM_GET(img,0)                    'get the image with the specified id No 0 in the camera
                                    buffer and store it in the img image
```

16.4. Contour Position

```
DIM num
ZVOBJECT img                   'image
ZVOBJECT subImg                'sub image
ZV_IMGGETSUB(img, subImg, s_x, s_y, s_w, s_h) 'get the sub-image from img into subImg
ZV_THRESH(subImg, subImg, thresh0, thresh1) 'threshold image
ZV_OPENING(dst, dst, 5)       'opening operation
ZV_CONTGEN(dst, contours, 0, 0) 'get the outer contour through point set method
ZV_CONTFILTER(contours, 0, 1500, 3000, 0)
                               'retain contours with an area within the range of 1500-3000
ZV_CONTFILTER(contours, 5, 0.9, 1.0, 0)
                               'retain contours with roundness in the range of 0.9-1.0
```

```
ZV_CONTSORT(contours,5,0) 'sort contours in descending order based on roundness
ZV_CONTFILTER(contours, -1, 0, 0, 0) 'select the contour with serial number 0
ZV_LISTGET(contours,contour,0) 'get the contour numbered 0 from the contour list
num = ZV_CONTCOUNT(contour) 'get the number of contour points
IF num < 1 THEN
    PRINT "ERROR: Contour Abnormal!"
ENDIF
ZV_CONTCENTER(contour,0) 'output the x y of the contour position into TABLE(0)TABLE(1)
```

16.5. Line Intersection Positioning

```
'Intersection and endpoint of two straight lines
DIM sectx, secty
DIM x11,y11,x12,y12,x21,y21,x22,y22

'generate the measurement area and set the measurement parameters, and generate the measurer of first straight line
ZV_MRGENLINE(mr1, cx, cy, w, h, angle, interp, sub_num, sub_width)
ZV_MRSETADV(mr1,filter_size,thresh,polar,select)

'generate the measurement area and set the measurement parameters, and generate the measurer of second straight line
ZV_MRGENLINE(mr2, cx2, cy2, w2, h2, angle2, interp, sub_num, sub_width)
ZV_MRSETADV(mr2,filter_size,thresh,polar,select)

'measure the straight line and put the end points of the straight line into the TABLE (0) and TABLE (10) respectively.
ZV_MRLINE(mr1, img, rstMat, 0)
ZV_MRLINE(mr2, img, rstMat, 10)
x11 = TABLE(0)
y11 = TABLE(1)
x12 = TABLE(2)
y12 = TABLE(3)
x21 = TABLE(4)
```

```
y21 = TABLE(5)
x22 = TABLE(6)
y22 = TABLE(7)

'calculate whether two points intersect and put the intersection point into TABLE (0), and return whether the straight lines intersect.
LOCAL is_paral
is_paral = ZV_INTERSECTLL(x11,y11,x12,y12,x21,y21,x22,y22, 0)
IF 0 <> is_paral THEN
	print "straight lines parallel"
ENDIF

'get the straight line intersection point
sectx = TABLE(0)
secty = TABLE(1)
```

16.6. Vector Correction

```
'calculate the transformation matrix based on the current and reference vectors
ZV_GETRIGIDVECTOR(mat, new_x, new_y, new_angle, base_x, base_y, base_angle)

'the position before correction is stored in the TABLE (0)
TABLE(0,s_x,s_y)

'position correction
ZV_AFFINETRANS(mat, 1, 0, 10)
d_x = TABLE(10)
d_y = TABLE(11)
```

16.7. Two-Point Correction

'put the two points before and after the transformation into TABLE(0) and TABLE(10)

```
respectively
TABLE(0, base_x1, base_y1, base_x2, base_y2)
TABLE(10, new_x1, new_y1, new_x2, new_y2)

'get the rigid transformation matrix mat based on two points
ZV_GETRIGID(mat, 0, 10)

'the position before correction is stored in the TABLE (20)
TABLE(20,s_x,s_y)

'position correction
ZV_AFFINETRANS(mat, 1, 20, 30)
d_x = TABLE(30)
d_y = TABLE(31)
```

16.8. Measurement Position Correction

```
ZVOBJECT modImg, model, mr

'take the sub-image of the interesting part as a template image
ZV_IMGGETSUB(img1, modImg, s_x, s_y, s_w, s_h)

'create a template or read the template directly. Generally, it is loaded from a file during software initialization.
ZV_SHAPECREATE(modImg,model,angle_st,angle_end,scale_min,scale_max,thresh,num
_level,pt_reduce,angle_step,scale_step)

'generate a circle measurer and set measurement parameters
ZV_MRGENCIRCLE(mr,cx,cy,r,ann_r,start_angle,ext_angle,interp,sub_num,sub_w)
ZV_MRSETADV(mr,filter_size,thresh,polar,select)

'the reference position of positioning matching can be combined with the current matching results to correct the measurement area
DIM base_pos(3)
base_pos(0, base_x, base_y, base_a)

'shape matching gets the current matching result
ZV_SHAPEFIND(model,img,matchs,min_score,nums,min_dist,min_thresh,
accuracy, speed, polar)

'get the matching results of the first row into TABLE(0)
```

ZV_MATGETROW(matchs,0,5,0)

'generate transformation matrix based on baseline and current results
ZV_GETRIGIDVECTOR(mat,base_pos(0),base_pos(1),base_pos(2),TABLE(1),TABLE(2),
TABLE(3))

'use the transformation matrix to correct the measurement area, that is, how much the current matching result is translated or rotated relative to the benchmark matching result, the measurement area also translates and rotates by the same amount.
ZV_MRCORRECT(mr, mat, corr_mr)

'measure the circle using the corrected measurement area
ZV_MRCIRCLE(corr_mr,img,mat_pts,10)

'get the measured circle result
cx = TABLE(10)
cy = TABLE(11)
radius = TABLE(12)

16.9. File Operation

For 5XX series controllers, C: represents the /zmc/flash/ directory.

- Creation of file directory
  FILE "MAKE_DIR", "C:/test/"
  'create a test directory under the /zmc/flash/ directory

- Copy of files
  FILE "FLASH_COPY", "C:/src.bmp","C:/dst.bmp"
  'copy the image src.bmp in the /zmc/flash/ directory to dst.bmp

- Deletion of files
  FILE "FLASH_DEL", "C:/src.bmp"
  'delete the image src.bmp in the /zmc/flash/ directory

Chapter XVII Appendix

17.1. Knowledge Expansion

17.1.1. Matrix

A rectangular number table with m rows and n columns arranged by m × n numbers a_{ij} (i = 1,2⋯m, j = 1,2⋯n) is called a matrix of m rows and n columns (m × n matrix), and this number of m × n are elements of matrix A, a_{ij} is called the element i rows j columns of matrix A, the i is the rows, the j is the columns. So, matrix can be written as A = (a_{ij}) or A = $(a_{ij})_{m \times n}$ / $A_{m \times n}$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

17.1.1.1. Transpose

Replace the rows of the m × n matrix A with columns of the same order to obtain an n × m matrix. This matrix is called the transposed matrix of A, denoted A^T or A'.

Basis Properties:

- $(A^T)^T = A$
- $(A + B)^T = A^T + B^T$
- $(kA)^T = kA^T$
- $(AB)^T = B^T + A^T$

For example:

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 3 & -1 & 4 \end{bmatrix}$$

The transpose matrix is

$$A = \begin{bmatrix} 1 & 3 \\ 2 & -1 \\ 0 & 4 \end{bmatrix}$$

17.1.1.2. Reverse Torque

Suppose A is a n order matrix, and there is another n order matrix B, then make AB = BA = E, that is, matrix A is reverse, and matrix B is the reverse matrix of A.

Basis Properties:

- The status of A and B is equal, so the two matrices A and B are inverse matrices of each other. It is also said that A is the inverse matrix of B.
- Unit matrix E is reverse, that is, $E = E^{-1}$
- Zero matrix is not reverse.
- If A is reverse, then reverse matrix of matrix A is unique.
- If A is reverse, then A^{-1} also is reverse, and $(A^{-1})^{-1} = A$
- If A is reverse, then A^T also is reverse, and $(A^T)^{-1} = (A^{-1})^T$
- If A and B are the same order matrix and they are both reverse, then AB are reverse, and $(AB)^{-1} = B^{-1}A^{-1}$

For example:

$$\begin{pmatrix}1&1&1\\2&3&2\\3&8&2\end{pmatrix}\to\left(\begin{array}{ccc|ccc}1&1&1&1&0&0\\2&3&2&0&1&0\\3&8&2&0&0&1\end{array}\right)\to\left(\begin{array}{ccc|ccc}1&1&1&1&0&0\\0&1&0&-2&1&0\\0&5&-1&-3&0&1\end{array}\right)\to\left(\begin{array}{ccc|ccc}1&0&1&3&-1&0\\0&1&0&-2&1&0\\0&0&-1&7&-5&1\end{array}\right)$$

$$\to\left(\begin{array}{ccc|ccc}1&0&1&3&-1&0\\0&1&0&-2&1&0\\0&0&1&-7&5&-1\end{array}\right)\to\left(\begin{array}{ccc|ccc}1&0&0&10&-6&1\\0&1&0&-2&1&0\\0&0&1&-7&5&-1\end{array}\right)\to\begin{pmatrix}10&-6&1\\-2&1&0\\-7&5&-1\end{pmatrix}$$

17.1.1.3. Matrix Multiplication

Suppose A is the matrix of m × P, B is the matrix of n × P, then the multiple of m × n 's matrix A and B as C = AB, and i rows and j columns of matrix C can be represented as:

$$(AB)_{ij} = \sum_{k=0}^{p-1} a_{ik}b_{kj} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{ip}b_{pj}$$

Basis Properties:

- (AB)C = A(BC)
- (A + B) C = AC + BC
- C (A + B) = CA + CB

- k (AB) = (kA)B + A(kB)

For Example:

$$\begin{bmatrix}2 & 1\\4 & 3\end{bmatrix} * \begin{bmatrix}1 & 2\\1 & 0\end{bmatrix} = \begin{bmatrix}2*1+1*1 & 2*2+1*0\\4*1+3*1 & 4*2+3*0\end{bmatrix} = \begin{bmatrix}3 & 4\\7 & 8\end{bmatrix}$$

17.1.2. Image

17.1.2.1. Image Multiplication

Image multiplication is the point-to-point multiplication of two images, suppose the image size is m × n, G = FH, then i rows and j columns element of matrix G can be represented as: $g(x, y)_{ij} = f(x, y)_{ij} \cdot h(x, y)_{ij}$

For Example:

$$\begin{pmatrix}1 & 1 & 1\\2 & 3 & 2\\3 & 8 & 2\end{pmatrix} \cdot \begin{pmatrix}1 & 1 & 0\\1 & 1 & 0\\0 & 0 & 0\end{pmatrix} = \begin{pmatrix}1 & 1 & 0\\2 & 3 & 0\\0 & 0 & 0\end{pmatrix}$$

17.1.2.2. Image Division

Image division is the point-to-point division of two images, suppose the image size is m ÷ n, G = F ÷ H, then i rows and j columns element of matrix G can be represented as: $g(x, y)_{ij} = f(x, y)_{ij} \div h(x, y)_{ij}$

For Example:

$$\begin{pmatrix}1 & 1 & 1\\2 & 3 & 2\\3 & 8 & 2\end{pmatrix} \div \begin{pmatrix}1 & 1 & 1\\2 & 1 & 1\\3 & 4 & 2\end{pmatrix} = \begin{pmatrix}1 & 1 & 1\\1 & 3 & 2\\1 & 2 & 1\end{pmatrix}$$

17.1.2.3. Norm

Norm is the function that is with "length" concept. Suppose image F size is m × n, then,

- Norm 1:

$$L_1(F) = \|F\|_1 = \sum_{x=0}^{m}\sum_{y=0}^{n}|f(x,y)|$$

- Norm 2:

$$L_2(F) = \|F\|_2 = \sum_{x=0}^{m}\sum_{y=0}^{n}\sqrt{f(x,y)^2}$$

- Infinite norm:

$$L_\infty(F) = \|F\|_\infty = \max_{0\le x<m,\,0\le y<n}|f(x,y)|$$

Basic Properties:

- For any set of basis for a finite-dimensional normed linear space, the norm is a continuous function of the coordinates of the elements.
- All norms of finite-dimensional linear spaces are equivalent.
- A finite-dimensional linear space over the real number field must be complete.
- The necessary and sufficient condition for a sequence in a finite-dimensional normed linear space to converge according to coordinates is that it converges according to any norm.

For Example: if image F is:

$$\begin{pmatrix}1 & 1 & 1\\ 2 & 3 & 2\\ 3 & 8 & 2\end{pmatrix}$$

Then:

- Norm 1:

$$\left\|\begin{pmatrix}1 & 1 & 1\\ 2 & 3 & 2\\ 3 & 8 & 2\end{pmatrix}\right\|_1 = |1|+|1|+|1|+|2|+|3|+|2|+|3|+|8|+|2| = 23$$

- Norm 2:

$$\left\|\begin{pmatrix}1 & 1 & 1\\ 2 & 3 & 2\\ 3 & 8 & 2\end{pmatrix}\right\|_2 = \sqrt{1^2+1^2+1^2+2^2+3^2+2^2+3^2+8^2+2^2} = 9.85$$

- Infinite norm:

$$\left\|\begin{pmatrix}1 & 1 & 1\\ 2 & 3 & 2\\ 3 & 8 & 2\end{pmatrix}\right\|_\infty = \max_{0\le x<3,\,0\le y<3}(|1|+|1|+|1|+|2|+|3|+|2|+|3|+|8|+|2|) = 8$$

17.1.2.4. Distance Between Pixels

An important concept describing the connection between pixels is the distance between pixels. Given three pixels p, q, r, the coordinates are (x, y), (s, t), (u, v) respectively, if the following conditions are met, the function D is said to be a **distance measurement function.**

(1) $D(p, q) \geq 0$ ($D(p, q) = 0$, and this is only valid when p = q.

(2) $D(p, q) = D(p, q)$

(3) $D(p, r) \leq D(p, q) + D(q, r)$

Among the above three conditions, the first condition indicates that the distance between two pixels is always positive (when the two pixels have the same spatial position, the distance between them is zero). The second condition indicates that the distance between two pixels is irrelevant to the choice of start and end points. The condition 3 indicates that the shortest distance between two pixels is along a straight line.

In digital images, distances are measured in different ways. The **Euclidean distance** between points p and q (that is, the distance with norm 2) is defined as:

$$D_E(p,q) = [(x-s)^2 + (y-t)^2]^{1/2}$$

According to this distance measure, pixels whose distance from (x, y) is less than or equal to some value d are included in a circle with (x, y) as the center and d as the radius.

The D_4 distance between points p and q (that is, the distance with a norm of 1) is also called the **urban distance** and is defined as:

$$D_4(p,q) = |x-s| + |y-t|$$

According to this distance measure, pixels whose distance from D_4 (x, y) is less than or equal to some value d are included in a diamond with (x, y) as the center.

The D_8 distance between points p and q (that is, the distance with norm ∞) is also called the **checkerboard distance** and is defined as:

$$D_8(p,q) = \max(|x-s|, |y-t|)$$

According to this distance measure, pixels whose distance from D_8 (x, y) is less than or equal to some value d are included in a square with (x, y) as the center.

17.1.2.5. Image Average Value

Suppose the image F size is m × n, then:

$$\mathrm{F_{mean}} = \frac{\sum_{x=0}^{m-1}\sum_{y=0}^{n-1} f(x,y)}{\mathrm{m \times n}}$$

For Example: if image F is:

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 3 & 2 \\ 3 & 8 & 2 \end{pmatrix}$$

Then:

$$\mathrm{F_{mean}} = \frac{1+1+1+2+3+2+3+8+2}{3 \times 3} = 2.56$$

17.1.2.6. Image Variance

Suppose the image F size is m × n, then:

$$\mathrm{F_{variance}} = \frac{\sum_{x=0}^{m-1}\sum_{y=0}^{n-1} (f(x,y) - F_{mean})^2}{\mathrm{m \times n}}$$

For Example: if image F is:

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 3 & 2 \\ 3 & 8 & 2 \end{pmatrix}$$

Then:

$$\begin{aligned}\mathrm{F_{variance}} = [&(1-2.56)^2 + (1-2.56)^2 + (1-2.56)^2 + (2-2.56)^2 + (3-2.56)^2 \\ &+ (2-2.56)^2 + (3-2.56)^2 + (8-2.56)^2 + (2-2.56)^2] \div (3 \times 3) = 2.06\end{aligned}$$

17.1.2.7. Histogram

A histogram is an abstract representation of an image. By modifying or changing the image histogram, the grayscale distribution of the image pixels can be changed, thereby achieving image enhancement. Histograms are obtained through statistics of images. For a grayscale image, its grayscale histogram reflects the statistics of different grayscale levels in the image.

$$h(f) = n_f \quad f = 0,1, \cdots, L-1$$

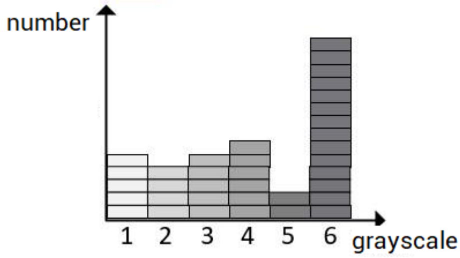
n_f is the number of pixels that are with grayscale value f in image f(x,y).

For Example:

1	2	3	4	5	6
6	4	3	2	2	1
1	6	6	4	6	6
3	4	5	6	6	6
1	4	6	6	2	3
1	3	6	4	6	6

mat

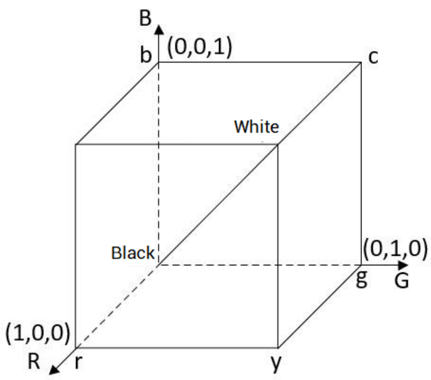
gray scale	1	2	3	4	5	6
number	5	4	5	6	2	14



17.1.2.8. Color Space

- RGB

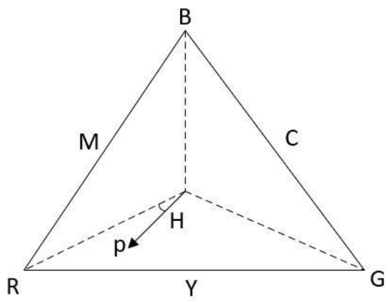
According to the structure of the human eye, there are three basic color-sensing cone cells in the human retina, and human perception of color is the result of the three types of cells working together. In this way, all colors can be considered as the three basic colors of red (R), green (G) and blue (B).

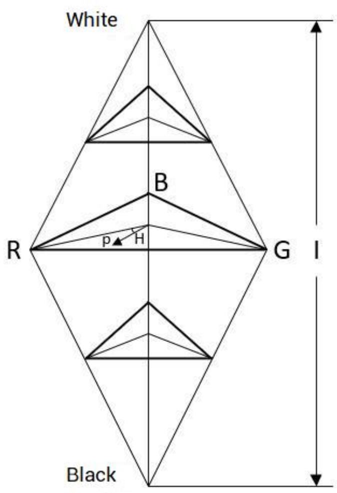


C = rR + gG + bB: C means one certain color, R, G, B mean three basic colors, r, g, b represent scaling coefficient.

- HIS

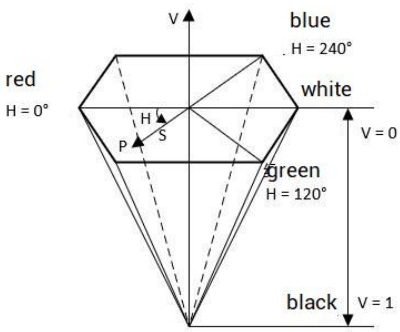
The most commonly used model for color processing is the HSI model, where H represents hue, S represents saturation, and I represents density. The three components of HSI correspond to the three basic characteristic quantities commonly used by people to describe colors, namely brightness, hue and saturation.





- HSV

The HSV model is closer to human perception of color than the HSI model. H in the HSV model represents hue, S represents saturation, and V represents brightness value.



17.1.2.9. Grayscale Image

Gray Scale Image is also called gray order image. The relationship between white and black is divided into several levels according to the logarithmic relationship, called grayscale, which is divided into 256 levels in total.

$$Gray = R * 0.3 + G * 0.59 + B * 0.11$$

For Example:

$$R \rightarrow \begin{pmatrix} 255 & 255 & 255 \\ 255 & 255 & 255 \\ 255 & 255 & 255 \end{pmatrix}, \ G \rightarrow \begin{pmatrix} 255 & 255 & 255 \\ 255 & 255 & 255 \\ 255 & 255 & 255 \end{pmatrix}, \ B \rightarrow \begin{pmatrix} 255 & 255 & 255 \\ 255 & 255 & 255 \\ 255 & 255 & 255 \end{pmatrix}$$

$$\begin{aligned} \text{Gray} &= 0.3\begin{pmatrix} 255 & 255 & 255 \\ 255 & 255 & 255 \\ 255 & 255 & 255 \end{pmatrix}_R + 0.59\begin{pmatrix} 255 & 255 & 255 \\ 255 & 255 & 255 \\ 255 & 255 & 255 \end{pmatrix}_G + 0.11\begin{pmatrix} 255 & 255 & 255 \\ 255 & 255 & 255 \\ 255 & 255 & 255 \end{pmatrix}_B \\ &= \begin{pmatrix} 76.5 & 76.5 & 76.5 \\ 76.5 & 76.5 & 76.5 \\ 76.5 & 76.5 & 76.5 \end{pmatrix} + \begin{pmatrix} 150.45 & 150.45 & 150.45 \\ 150.45 & 150.45 & 150.45 \\ 150.45 & 150.45 & 150.45 \end{pmatrix} + \begin{pmatrix} 28.05 & 28.05 & 28.05 \\ 28.05 & 28.05 & 28.05 \\ 28.05 & 28.05 & 28.05 \end{pmatrix} \\ &= \begin{pmatrix} 255 & 255 & 255 \\ 255 & 255 & 255 \\ 255 & 255 & 255 \end{pmatrix} \end{aligned}$$

17.1.2.10. Mirror

There are two types of image mirroring transformation, horizontal mirroring and vertical mirroring. Horizontal mirroring takes the vertical centerline of the image as the axis and swaps the pixels of the image, that is, swapping the left and right halves of the image. Vertical mirroring takes the horizontal centerline of the image as the axis and swaps the upper and lower parts of the image.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Invert along axis x:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Invert along axis y:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Mirroring of origin:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

17.1.2.11. Rotation

Rotation transformation is to change one figure into another figure. During the change process, all points on the original image change around a fixed point in the same direction and rotate at the same angle.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$
$$\begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

17.1.2.12. Scaling

Image scaling is the process of adjusting the size of a digital image.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$
$$\begin{bmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

17.1.2.13. Affine

Affine transformation in space transformation corresponds to five transformations, translation, scaling, rotation, flipping, and cross-cutting. The process of these five changes from the original image to the transformed image, which can be described by an affine transformation matrix.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Cut along with axis x:

$$\begin{bmatrix} 1 & tan\theta & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Cut along with axis y:

$$\begin{bmatrix} 1 & 0 & 0 \\ tan\theta & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

17.1.2.14. Median Filtering

Median filtering is a nonlinear signal processing technology based on sorting statistical theory that can effectively suppress noise. The basic principle of median filtering is to replace the value of a point in a digital image or digital sequence with the values of points in a neighborhood of that point. Instead of the median value, the surrounding pixel values are close to the true value, thereby eliminating isolated noise points and having a good filtering effect on impulse noise. In particular, while filtering out noise, it can protect the edges of the signal so that it does not be blurred. Principle: it sets the gray value of each pixel to the median value of the gray value of all pixels in a neighborhood window of that point.

For example: suppose image F is

$$\begin{pmatrix} 31 & 12 & 23 \\ 43 & 1 & 32 \\ 7 & 3 & 43 \end{pmatrix},$$

Then, filter middle pixel points. All pixels are ordered from small to large: 1, 3, 7, 13, 23, 31, 32, 43, 43

$$\begin{pmatrix} 31 & 12 & 23 \\ 43 & 1 & 32 \\ 7 & 3 & 43 \end{pmatrix} \rightarrow \begin{pmatrix} 31 & 12 & 23 \\ 43 & 23 & 32 \\ 7 & 3 & 43 \end{pmatrix}$$

17.1.2.15. Mean Filtering

Mean filtering is a typical linear filtering algorithm. Its principle is to replace each pixel value in the original image with the mean value. Boundaries are treated as elemental symmetries (see Custom Morphology).

Calculation formula:

$$g(x,y) = \frac{\Sigma f(x,y)}{m}$$

m is the total number of pixels in the template including the current pixel.

For Example:

Suppose image F is

$$\begin{pmatrix} 2 & 3 & 3 \\ 5 & 6 & 4 \\ 9 & 6 & 7 \end{pmatrix}$$

Then, filter middle pixel point.

Mean value of all pixels is (2+3+3+5+6+4+9+6+7) / 9 = 5

$$\begin{pmatrix} 2 & 3 & 3 \\ 5 & 6 & 4 \\ 9 & 6 & 7 \end{pmatrix} \rightarrow \begin{pmatrix} 2 & 3 & 3 \\ 5 & 5 & 4 \\ 9 & 6 & 7 \end{pmatrix}$$

17.1.2.16. Gaussian Filter

Gaussian filter is a linear smoothing filter, suitable for eliminating Gaussian noise, and is widely used in the noise reduction process of image processing. Boundaries are treated as elemental symmetries (see Custom Morphology).

Two-dimensional Gaussian function:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Commonly used 3*3 and 5*5 Gaussian templates are as follows (standard deviation = 1.3):

$$\frac{1}{16} \times \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

17.1.2.17. Bilateral Filtering

Bilateral filtering is a nonlinear filter that can maintain edges, reduce noise, and smooth. Bilateral filtering uses a weighted average method. The weighted average of the brightness values of surrounding pixels is used to represent the intensity of a certain pixel. The weighted average used is based on Gaussian distribution. The most important thing is that the weight of bilateral filtering not only considers the Euclidean distance of the pixel, but also takes the radiation difference in the pixel range domain into account. These two weights are considered simultaneously when calculating the center pixel.

17.1.2.18. Sobel Edge Detection

The Sobel operator is a discrete differentiation operator mainly used for edge detection. It combines Gaussian smoothing and differential derivation to calculate the approximate gradient of the image grayscale function. Using this operator at any point in the image will produce the corresponding gradient vector or its normal vector.

To detect horizontal transformation, the 3*3 kernel is:

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

To detect vertical transformation, the 3*3 kernel is:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

17.1.2.19. SCHARR Filter

Although the Sobel operator can effectively extract image edges, it has a poor effect on weak edges in the image. Therefore, in order to effectively extract weak edges, the gap between pixel values needs to be increased, so the Scharr operator is introduced. The Scharr operator is an enhancement to the difference of the Sobel operator, so the principle and usage of detecting image edges between the two are the same. The size of the edge detection filter of the Scharr operator is 3×3, so it is also called the Scharr filter. The difference between pixel values can be increased by amplifying the weight coefficient in the filter. This idea is adopted by the Scharr operator, which is an edge detection operator in the X and Y directions.

$$G_x = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \quad G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix}$$

17.1.2.20. Laplacian Edge Detection

Laplacian edge extraction uses the second-order derivative to extract edges, which is an isotropic edge extraction operator. Isotropy means that using this operator, you can sharpen the boundaries and lines in any direction without directionality. Like sobel, they use different operators to extract edges in the x and y directions. This Laplacian operator has advantages that distinguish it from other first-order differential operators. But its disadvantage is that it is sensitive to noise compared to first-order differential. It responds more strongly to isolated pixels than to edges or lines, so it is only suitable for noise-free images. It is precisely because the Laplacian operator is sensitive to outliers and noise. Before using the Laplacian operator to extract edges, we first use Gaussian smoothing of the image. This process is the Laplacian-Gauss (LOG) operator. It combines the Gaussian smoothing filter and the Laplacian sharpening filter to smooth out the noise first and then perform edge detection, so the effect will be better.

17.1.2.21. Canny Edge Detection

Canny edge detection operator is a multi-level detection algorithm. Proposed by John F. Canny in 1986, he also proposed three major criteria for edge detection:

- Edge detection with low error rate: the detection algorithm should accurately find as many edges as possible in the image and reduce missed detections and false detections as much as possible.
- Optimal positioning: the detected edge point should be accurately positioned at the center of the edge.
- Any edge in the image should be marked only once, and image noise should not produce false edges.

The Canny algorithm has been used as a standard edge detection algorithm, since then, various improved algorithms based on the Canny algorithm have appeared. Now, the Canny algorithm and its various variants are still an excellent edge detection algorithm.

17.1.2.22. Gradient

The image gradient is used to calculate speed of image changing. For edge part of image, the grayscale value is with big change, so the gradient value is also big. Generally, image gradient calculates image edge information. Strictly speaking, image gradient requires derivative, but it gets approximate value of gradient through calculating pixel deviation value. The commonly used operators are Sobel, Scharr and Lapacian.

17.1.2.23. Frequency Domain

Frequency domain means analyze function from the angle of function's frequency, and the corresponding aspect is time domain. That is, if signals are analyzed from time domain, the time is horizontal coordinate, amplitude is vertical coordinate. But if from frequency domain, the coordinates are opposite.

For images, Fourier transform is used to transform the image from the spatial domain to the frequency domain, so the Fourier spectrum characteristics can be used for image processing.

17.1.2.24. Dilation and Erosion

Image dilation and erosion are two basic morphological operations, which are mainly used to find maximum and minimum areas in the image.

The expansion is similar to "field expansion", which expands the highlighted area or white part of the image, and the resulting image is larger than the highlighted area of the original image.

Corrosion is similar to "area encroachment", which reduces and refines the highlighted area or white part of the image, and the resulting image is smaller than the highlighted area of the original image.

17.1.2.25. Opening Operation and Closed Operation

The image opening and closing operations are related to the dilation and erosion operations, and are composed of operations composed of the compound and set operations (union, intersection, complement, etc.) of the expansion and erosion operations.

Opening operation: first erode the image and then dilate it.

Closed operation: first dilate the image and then erode it.

17.1.2.26. Histogram Equalization

Histogram equalization is a typical automatic method to obtain image enhancement by correcting the histogram of the image. Histogram equalization is mainly used to enhance the contrast of images with a small dynamic range. The basic idea of this method is to transform the histogram of the original image into a form that is evenly distributed throughout the entire grayscale range, thereby increasing the dynamic range of pixel grayscale values and thus achieving enhancement of the overall contrast of the image.

Write the grayscale histogram into a general probability expression:

$$p(f) = \frac{n_f}{n} \quad f = 0,1,\cdots,L-1$$

Among them, n is the total number of pixels in the image. By normalizing the total number of pixels in the image, each column of the obtained histogram expresses the proportion of each gray value pixel in the image.

The basic idea of histogram equalization is to transform the histogram of the original image into a uniformly distributed form. Here, a transformation function needs to be determined, that is, an enhancement function. This enhancement function needs to meet two conditions:

- It is a single-valued single-increasing function in the range of $0 \le f \le L - 1$. This is to ensure that the gray levels of the original image still maintain the original order from black to white (or from white to black) after transformation.
- If the equalized image is g(x, y), then $0 \le f \le L - 1$ should be $0 \le g \le L - 1$. This condition is used to ensure that the dynamic range of the gray value of the image before and after transformation is consistent.

It can be proved that the functional relationship that satisfies the above two conditions and can convert the original distribution of f into a uniform distribution in g can be obtained from the cumulative histogram of the image f(x, y). The transformation from f to g is

$$g_f = \sum_{i=0}^{f} \frac{n_i}{n} = \sum_{i=0}^{f} p(i) \quad f = 0,1,\cdots,L-1$$

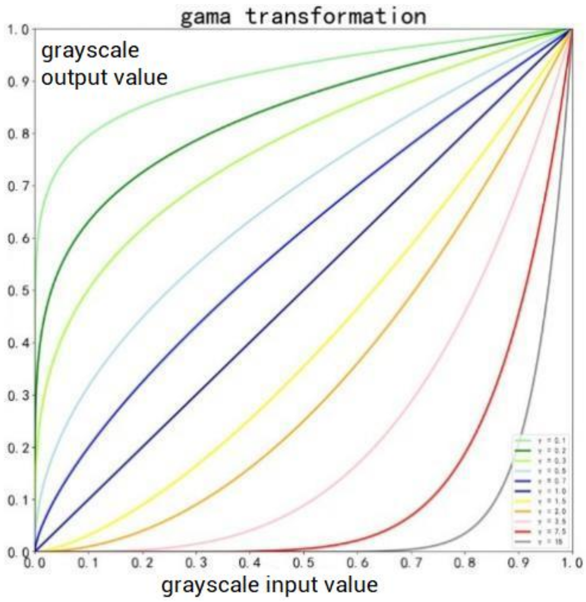
According to the above formula, the gray value of each pixel in the image after histogram equalization can be directly calculated from the original image histogram.

17.1.2.27. Gamma Transform

Gamma transform is performed on the image. Gamma transform is often used to adjust the contrast of overexposed or underexposed (too dark) grayscale images. The calculation formula is as follows:

$$I_{out} = cI_{in}^{\gamma}$$

Among them, c and y are positive constants, c is the grayscale scaling coefficient, usually 1. y is the gamma factor size, which controls the scaling degree of the entire transformation.



17.1.2.28. Grayscale Stretching

Grayscale stretching is also called contrast stretching. It is the most basic grayscale transformation and uses the simplest piecewise linear transformation function. Its main idea is to improve the dynamic range of grayscale levels during image processing.

17.1.2.29. Image normalization

Image normalization refers to a series of standard processing transformations on an image to transform it into a fixed standard form. This standard image is called a normalized image.

17.1.2.30. Image Enhancement

To enhance useful information in an image, it can be a distortion process whose purpose is to improve the visual effect of the image for the application of a given image.

Purposefully emphasize the overall or local characteristics of the image, that is, make the original unclear image clear or emphasize some features of interest, then expand the differences between the features of different objects in the image, suppress uninteresting features, and improve the image quality, rich information, enhance image interpretation and recognition effects, to meet the needs of some special analysis.

17.1.2.31. Binarization

Image Binarization is to set the grayscale value of the pixels on the image to 0 or 255, which is the process of making the entire image appear obviously black and white.

17.1.2.32. Adaptive Binarization

Perform adaptive thresholding on the input image to generate a binary image. The effect of adaptive thresholding is similar to high-pass filtering an image -- extracting the contour of the target, and the size of target contour depends on the size of the filter and the gradient magnitude of the target contour itself.

17.1.2.33. Automatic Binarization

Automatic binarization uses the OTSU. The Otsu algorithm is also called the maximum inter-class variance method. It is an algorithm that can automatically determine the threshold in binarization. Then, the foreground and background images can be separated.

17.1.3. Matching

17.1.3.1. Shape Matching

Shape-based matching uses the contour shape of the target object to describe the template. Shape matching is to calculate the similarity or dissimilarity of two shapes based on shape description and certain judgment criteria. The matching result between two shapes is represented by a numerical value called shape similarity. The greater the shape similarity, the more similar the two shapes are. Dissimilarity is also called shape distance, which is contrary to similarity, the smaller the shape distance, the more similar the two shapes are.

17.1.3.2. NCC Matching

Based on NCC, it is used to compare the similarity of two images to match the target. And this is applied in industrial production detection and monitoring, object detection and identification, and so on. The NCC algorithm can effectively reduce the impact of lighting on image comparison results.

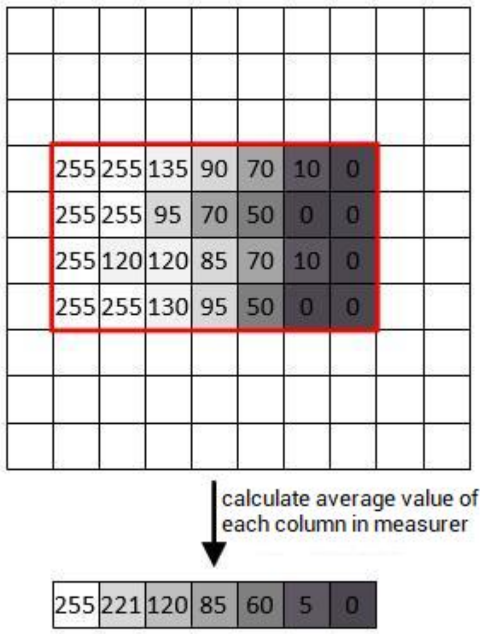
17.1.3.3. Grayscale Matching

Template matching based on gray value is suitable for detection targets whose gray changes in the image are relatively stable, noise is relatively small, and gray differences are obvious. This is a matching method that is not recommended because it is highly complex, and it can only detect one target at a time, so it is time-consuming, and very sensitive to lighting and size changes.

17.1.4. Measurement

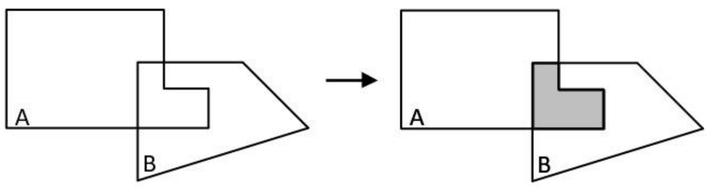
17.1.4.1. Grayscale Projection

Calculate the grayscale projection value in horizontal and vertical direction.

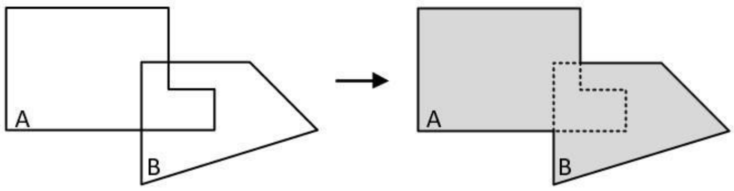


17.1.5. Region

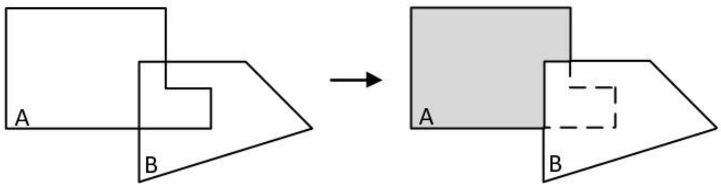
17.1.5.1. Intersection, Union and Difference



Intersection



Union



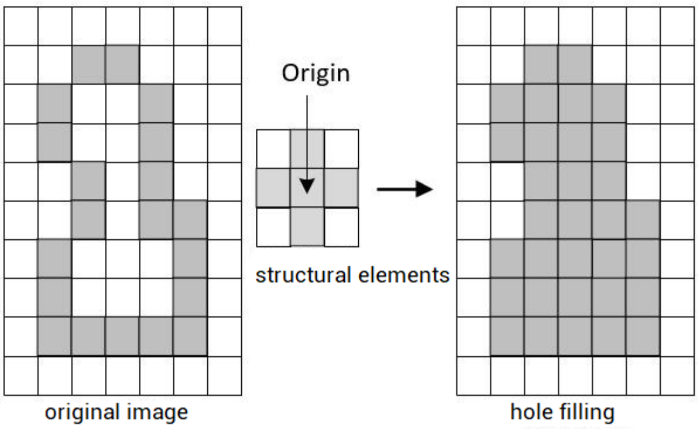
Difference

17.1.5.2. Connected Component

Connected component generally refers to the image area (Region, Blob) composed of foreground pixels with the same pixel value and adjacent positions in the image. Connected region analysis (Connected Component Analysis, Connected Component Labeling) refers to finding and labeling each connected region in the image.

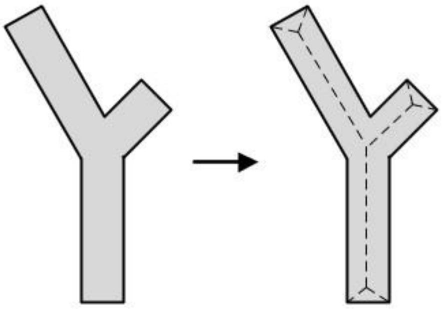
17.1.5.3. Hole Filling

A hole can be defined as a background region surrounded by a border connected by foreground pixels. Hole filling is based on dilation, complementation and intersection algorithms. When given a point in each hole, the goal is to fill all the holes with 255.



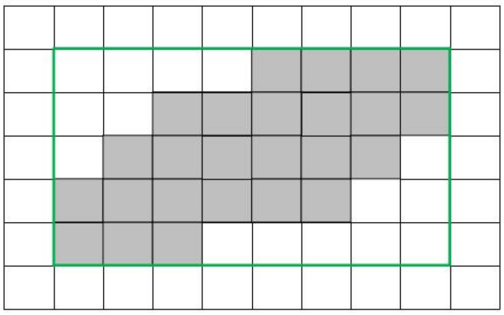
17.1.5.4. Skeletonization

Reduce foreground pixels as much as possible while maintaining the connectivity of the foreground area of a binary image, and finally obtains the "skeleton" of the image.

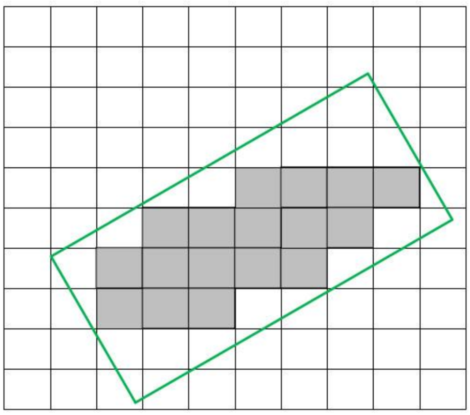


17.1.5.5. External Rectangle & Rotate External Rectangle

The minimum external moment of a region parallel to the horizontal axis, that is, the smallest rectangle parallel to the horizontal axis that can enclose the region.

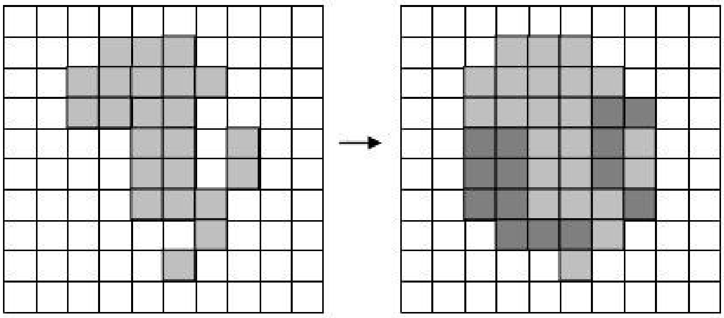


The minimum external moment of the area. This external moment is with angle, that is, the smallest rectangle with an angle that can surround the area.



17.1.5.6. Convexity

The shape factor of the region - convexity, the area of the region/the area of the convex hull corresponding to the region. If F_c is the area of the convex hull, and F_o is the original area of the region, then the convexity C is defined as: $C = F_O / F_C$



17.1.5.7. Compactness

The shape factor of the region - compactness. If L is the length of the contour and F is the area of the region, then the compactness C is defined as:

$$C' = \frac{L^2}{4F\pi}$$

$$C = max(1, C')$$

17.1.5.8. Rectangularity

The rectangularity of a region measures how close a shape is to a rectangle. The calculation of the rectangularity measure is ultimately based on the calculated area of the normalized difference between the rectangle and the input area relative to the area of the rectangle.

17.1.6. Recognition

17.1.6.1. Barcode

One-dimensional barcode refers to the arrangement rules of barcode bars and spaces. Code systems of commonly used one-dimensional code include: EAN code, 39 code, crossed 25 code, UPC code, 128 code, 93 code, ISBN code, and Codabar etc. A barcode is a mark composed of a set of regularly arranged bars, spaces and corresponding characters. The "bar" refers to the part with low light reflectivity, and the "space" refers to the part with high light reflectivity. The data composed by these bars and spaces expresses certain information and can be read by the device, then it can be converted into binary and decimal information compatible with the computer.

QR (Quick Response) code is also called two-dimensional barcode. It is a very popular encoding method on mobile devices in recent years. It can store more information than the traditional Bar Code, and more data types can be represented.

17.1.6.2. SVM

Support Vector Machine (SVM) is a type of generalized linear classifier that performs binary classification of data in a supervised learning manner, whose decision boundary is the maximum-margin hyperplane that solves the learning sample. SVM uses the hinge loss function to calculate the empirical risk and adds a regularization term into the solution system to optimize the structural risk. It is a classifier with sparseness and robustness. SVM can perform nonlinear classification through the kernel method and is one of the common kernel learning methods.

17.1.6.3. MLP

MLP Multi-layer Perceptron is a forward-structured artificial neural network ANN that maps a set of input vectors to a set of output vectors. MLP can be viewed as a directed graph consisting of multiple layers of nodes, and each layer is fully connected to the next layer. Except for the input node, each node is a neuron with a nonlinear activation function. The MLP is trained using the supervised learning method of BP backpropagation algorithm. MLP is a generalization of the perceptron, which overcomes the weakness of the perceptron that cannot identify linearly inseparable data.

17.1.7. Tool

17.1.7.1. Hough Transform

Hough transformation, transforming the image coordinate system into a parametric coordinate system according to mathematical expressions (such as straight lines or circles), points (m points) on the same line or circle will change into lines in the parametric coordinate system (m lines, several points in the image coordinate system will become several lines after conversion), these lines will intersect at one point, and then vote in the parameter coordinate system, the candidate object is obtained through the local maximum value

- **Linear detection**

In the rectangular coordinate system, a straight line:

$$L1: y = a_0 x + b_0$$

Among them, b_0 is the intercept of the straight line, a_0 is the slope of the straight line, a_0 and b_0 are constants, x and y are variables. Assume that a certain point (x_0, y_0) is on the straight line L1, there are countless straight lines passing through this point, so it will correspond to different a and b, in this parameter coordinate system, the point (x_0, y_0) becomes a straight line $b =- x_0 a + y_0$.

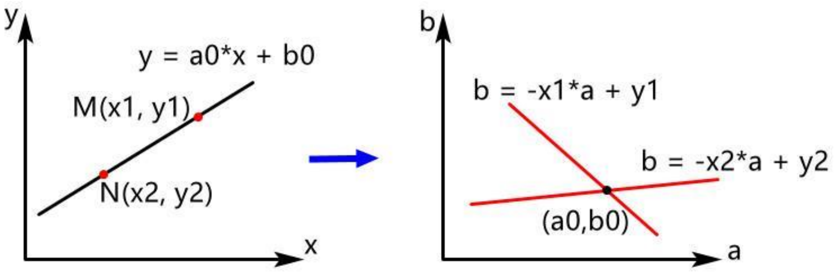
In the rectangular coordinate, a and b are variables, x_0 and y_0 are constants, then the line L1 can be:

$$L1: b = -x_0 a + y_0$$

If a line L1 was converted to parameter coordinate system, it will become one point (a_0, b_0).

$$L1: b = -x_0a + y_0$$

Under the rectangular coordinate system, assuming that there are M points on the straight line L1, it will become M straight lines under the parametric coordinate system. These M straight lines will intersect at a point (a_0, b_0), and the coordinates of this point represent L1's slope and intercept in the rectangular coordinates. Summarizing the above two transformations, it can be known that points in image space correspond to straight lines in parameter space one-to-one, and straight lines in image space correspond to points in parameter space one-to-one.



- **Circle detection**

For a circle, three parameters are needed to determine a circle (center coordinates and radius). The standard Hough circle transformation still converts the rectangular coordinates into a three-dimensional parameter space describing the circle, and then uses these three dimensions to perform cumulative measurements (voting), and determines whether it is a circle based on the voting results.

17.1.7.2. Camera Distortion

In actual shooting, camera distortion is a problem that is often encountered, such as radial distortion, tangential distortion, etc. Radial distortion is divided into pincushion distortion and barrel distortion, while tangential distortion is generally caused by the lens not being completely parallel to the image. The shape or process difference of the lens may also cause a certain degree of image distortion, so it is necessary to obtain the internal parameters of the camera through calibration and correct the image distortion.

The pinhole camera model is an ideal perspective model. It will obtain near and far images due to perspective. It will also produce distortion due to lens deviation, that is, geometric distortion. Unlike keystone distortion caused by perspective changes, geometric distortion is a deformation from the center of the image to the edge. The closer to the edge, the more severe the distortion will be.

Telecentric lenses will not produce perspective errors due to lens movement, and the image size will not be affected by the shooting distance. Within a fixed imaging distance range, the magnification is consistent and the distortion is minimal.

17.1.7.3. Camera Internal and External Parameters

In order to correspond the pixel distance in the image coordinate system to the coordinate distance in the world coordinate system, it is necessary to know the external parameter information of the camera and convert its actual distance in the world coordinate system through the transformation of the coordinate system.

- Internal parameters

The internal camera parameters obtained through calibration describe the characteristics of the camera used and are generally related to the internal structure of the camera itself. Internal parameters generally include the focal length of the camera, distortion coefficient, pixel pitch, center point coordinates, image width and height, etc.

- External parameters

The external parameters of the camera represent the three-dimensional position of the camera in the world coordinate system, such as the camera's X-axis coordinate, Y-axis coordinate, Z-axis coordinate, and the camera's orientation (such as the angle of rotation around the X-axis, Y-axis, Z-axis), etc.

17.1.7.4. Calibration

Camera calibration can establish the correspondence between points in a two-dimensional image and points in a three-dimensional space. **Camera calibration** is the process of obtaining the internal and external parameters of the camera. And accurate calibration can improve the accuracy of measurement and reduce errors.

17.1.8. Defect

17.1.8.1. Smooth Surface Defect Detection

The glossy defect detection system integrates machine vision technology such as cameras and image processing algorithms to efficiently detect, display and identify object's common surface defects (such as holes, damage, edge cracks, scratches, edge damage, etc.), defects, dirty spots, water and oil drop marks, streaks, missed coatings, wrinkles, dark spots, bright spots, dust, etc. in real-time, especially suitable for object production industry that requires appearance strictly and specific index, such as, plastics, paper, glass, electronics, metal, films, foils, etc., it can be seen the application range is very wide.

17.2. Camera Parameters

Note: the camera parameters of each camera series will be slightly different. The following parameters are just examples. If you encounter parameter setting errors, it is recommended to refer to the corresponding camera SKD to view the parameters.

17.2.1. Hikvision (Area Array)

Parameter Type	Parameter Name	Description
Command Type	"TriggerSoftware"	Under camera soft trigger mode, set soft trigger command parameters to trigger camera shooting, take photo once when triggered once.
Enumeration type	"PixelFormat"	Pixel format, the enumeration value of grayscale image is

		17301505 or the enumeration name is "Mono8", the enumeration value of RGB color image is 35127316 or the enumeration name is "RGB8".
	“LineSelector”	Line selector, select the external wiring to be configured, that is, select an external wiring and configure some properties, such as configuring the external wiring as input or output properties. The enumeration value of external line Line0 is 0 or the enumeration name is "Line0", the enumeration value of external line Line1 is 1 or the enumeration name is "Line1", the enumeration value of external line Line2 is 2 or the enumeration name is "Line2". For the connection between camera wiring and external devices, please refer to the document "Hikvision Camera IO Cable Connection Instructions"
	“LineMode”	Line mode controls whether the external wiring is used as an input or output signal. First use the "LineSelector" line selector to select a line, and then use the line mode to set it as an input or output

		signal. The enumeration value of the input signal is 0 or the enumeration name is "InPut", the enumeration value of the output signal is 8 or the enumeration name is "Strobe".
	“TriggerSelector”	Trigger selector is to select the trigger type for configuration. The enumeration value of frame trigger mode is 6 or the enumeration name is "FrameBurstStart".
	“TriggerSource”	Trigger source, that is, the source of the trigger signal in trigger mode. The enumeration value of soft trigger is 7 or the enumeration name is "Software", the enumeration value of external trigger Line0 is 0 or the enumeration name is "Line0", the enumeration value of external trigger Line2 is 2 or the enumeration name is " Line2".
	“TriggerActivation”	Trigger response mode, that is, what method to choose for triggering. The enumeration value for rising edge triggering is 0 or the enumeration name is "RisingEdge". The enumeration value for falling edge triggering is 1 or the

		enumeration name is "FallingEdge".
Boolean Type	“GevGVCPHeartbeatDisable”	Heartbeat packet disable, 1-enabled, 0-disabled. It is usually necessary to disable the heartbeat packet during program debugging.
	“AcquisitionFrameRateEnable”	Image acquisition frame rate enablement, 1-enabled, 0-disabled, the frame rate can be set only after enabling it.
	“ReverseX”	Horizontal image reversion is enabled, that is, the image is flipped left and right with the vertical axis as the flip axis, 1-enabled, 0-disabled.
	“CammaEnable”	Gamma enable, 1-enabled, 0-disabled, only after enabling can the gamma correction operation of pixel brightness be performed
Integer Type	“OffsetX”	The offset in the X direction of the ROI, that is, the x coordinate of the upper left corner of the ROI. When setting "OffsetX", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the offset in advance to correctly set "OffsetX".

		You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetX" + "Width" < maxValWidth
	“OffsetY”	The offset in the Y direction of the ROI, that is, the y coordinate of the upper left corner of the ROI. When setting "OffsetY", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the offset in advance to correctly set "OffsetY". You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetY" + "Height" < maxValHeight
	“Width”	ROI image's width. When setting "Width", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the image width in advance to correctly set "Width". And different resolution images are

		with different widths. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetX" + "Width" < maxValWidth
	“Height”	ROI image's height. When setting "Height", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the image height in advance to correctly set "Height". And different resolution images are with different heights. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetY" + "Height" < maxValHeight.
	“GevHeartbeatTimeout”	Heartbeat packet timeout, unit ms, range [1,600000], step size 1
	“GevSCPD”	Packet delay controls the delay (in timestamp counter units) inserted between each packet. This can be used as a rough flow control mechanism if the application or network infrastructure cannot

		keep up with packets coming from the device. Setting the packet sending delay can indirectly control the frame rate, range [1,1000000], step size 1
Floating-Point Type	“ExposureTime”	Camera exposure time, unit microsecond (us), range [34, 1e+6]
	“TriggerDelay”	Camera trigger delay, unit microsecond (us), range [0, 1.6e+7]
	“ResultingFrameRate”	The maximum frame rate allowed in the given current area of interest (default is the whole image), exposure time and bandwidth, unit FPS/s. This parameter cannot be written and can only be read. The purpose is to view the camera frames in the current environment.
	“AcquisitionFrameRate”	Set the image acquisition frame rate, unit FPS/s, range [1, 100000]. Before writing this parameter, "AcquisitionFrameRateEnable" parameter must be true, then the parameter can be written successfully.
	“Gamma”	Perform gamma correction on image pixel brightness, range [0, 4], < 1, improve image brightness, the smaller the value, the stronger the improvement. > 1, compress image

		brightness, the larger the value, the stronger the compression. Before writing this parameter, "GammaEnable" parameter must be true, then the parameter can be written successfully.

17.2.2. Hikvision (Line Array)

Parameter Type	Parameter Name	Description
Command Type	"TriggerSoftware"	Under camera soft trigger mode, set soft trigger command parameters to trigger camera shooting, take photo once when triggered once.
Enumeration type	"PixelFormat"	Pixel format, the enumeration value of grayscale image is 17301505 or the enumeration name is "Mono8", the enumeration value of RGB color image is 17301513 or the enumeration name is "BayerRGB".
	"TriggerSelector"	Trigger selector is to select the trigger type for configuration. The enumeration value of frame trigger mode is 6 or the enumeration name is "FrameBurstStart", and the enumeration value of row trigger mode is 9 or the enumeration

		name is "LineStart".
	"TriggerSource"	Trigger source, that is, the source of the trigger signal in trigger mode. The enumeration value of soft trigger is 7 or the enumeration name is "Software", the enumeration value of external trigger Line0 is 0 or the enumeration name is "Line0", the enumeration value of external trigger Line2 is 2 or the enumeration name is " Line2".
Boolean Type	"AcquisitionLineRateEnable"	Camera row frequency enable, 1-enabled, 0-disabled.
	"StrobeEnable"	Enable the strobe signal to be output to the selected line, 1-enabled, 0-disabled. It will only take effect when the camera's external wiring is used as an output signal after being enabled.
	"FrameTimeoutEnable"	Frame timeout enable, 1 – enabled, 0 – disabled.
Integer Type	"OffsetX"	The offset in the X direction of the ROI, that is, the x coordinate of the upper left corner of the ROI. When setting "OffsetX", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum

		value: Maximum value: step increment]) of the offset in advance to correctly set "OffsetX". You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetX" + "Width" < maxValWidth
	"OffsetY"	The offset in the Y direction of the ROI, that is, the y coordinate of the upper left corner of the ROI. When setting "OffsetY", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the offset in advance to correctly set "OffsetY". You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetY" + "Height" < maxValHeight
	"Width"	ROI image's width. When setting "Width", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step

		increment]) of the image width in advance to correctly set "Width". And different resolution images are with different widths. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetX" + "Width" < maxValWidth
	“Height”	ROI image's height. When setting "Height", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the image height in advance to correctly set "Height". And different resolution images are with different heights. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetY" + "Height" < maxValHeight.
	“StrobeLineDuration”	Strobe line output level time, that is, the level duration of the output signal line output selected by the line selector, unit us, [minVal: maxVal: increment] = [0: 1000000 :

		1], when the value is 0, the duration of level output is consistent with the exposure time by default.
	"AcqusitionLineRate"	Set the line frequency of the camera in Hz, which is the frequency of scanning lines.
	"ResultingLineRate"	The maximum line rate allowed in the given current area of interest (default is the whole image), exposure time and bandwidth, unit line/s. This parameter cannot be written and can only be read. The purpose is to view the camera line rate in the current environment.
	"LineDebouncerTimeNs"	Set the row debounce time, unit ns
Floating-Point Type	"ExposureTime"	Camera exposure time, unit microsecond (us), range [34, 1e+6]
	"TriggerDelay"	Camera trigger delay, unit microsecond (us), range [0, 1.6e+7]
	"ResultingFrameRate"	The maximum frame rate allowed in the given current area of interest (default is the whole image), exposure time and bandwidth, unit FPS/s. This parameter cannot be written and can only be read. The purpose is to view the camera frames in the current environment.

17.2.3. Basler

Parameter Type	Parameter Name	Description
Command Type	“TriggerSoftware”	Under camera soft trigger mode, set soft trigger command parameters to trigger camera shooting, take photo once when triggered once.
Enumeration type	“PixelFormat”	Pixel format, the enumeration signal name of grayscale image is "Mono8", the enumeration signal name of RGB color image is "BayerGB8".
	“LineSelector”	Line selector, select the external wiring to be configured, that is, select an external wiring and configure some properties, such as configuring the external wiring as input or output properties. The enumeration name of external wiring Line1 is "Line1", and the enumeration name of external wiring OutputLine1 is "OutputLine1". Please refer to the document "basler ace Camera Link Users Manual" for wiring of camera and external equipment.
	“LineMode”	Line mode controls whether the external wiring is used as an input

		or output signal. First use the "LineSelector" line selector to select a line, and then use the line mode to set it as an input or output signal. The enumeration name of input signal is "InPut", and the enumeration name of output signal is "Output".
	“TriggerMode”	Whether to enable trigger mode, that is, soft trigger or external trigger can only be used. The enumeration name for turning on trigger mode is "On", the enumeration name for turning off trigger mode is "Off".
	“TriggerSource”	Trigger source, that is, the source of the trigger signal in trigger mode. The enumeration name of soft trigger is "Software", The enumeration name of external trigger Line1 is "Line1".
Boolean Type	“AcquisitionFrameRateEnable”	Image acquisition frame rate enablement, 1-enabled, 0-disabled, the frame rate can be set only after enabling it.
	“ReverseX”	Horizontal image reversion is enabled, that is, the image is flipped left and right with the vertical axis as the flip axis, 1-

		enabled, 0-disabled.
	“CammaEnable”	Gamma enable, 1-enabled, 0-disabled, only after enabling can the gamma correction operation of pixel brightness be performed
Integer Type	“OffsetX”	The offset in the X direction of the ROI, that is, the x coordinate of the upper left corner of the ROI. When setting "OffsetX", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the offset in advance to correctly set "OffsetX". You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetX" + "Width" < maxValWidth
	“OffsetY”	The offset in the Y direction of the ROI, that is, the y coordinate of the upper left corner of the ROI. When setting "OffsetY", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the offset in

		advance to correctly set "OffsetY". You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetY" + "Height" < maxValHeight
	“Width”	ROI image's width. When setting "Width", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the image width in advance to correctly set "Width". And different resolution images are with different widths. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetX" + "Width" < maxValWidth
	“Height”	ROI image's height. When setting "Height", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the image height in advance to correctly set "Height".

		And different resolution images are with different heights. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetY" + "Height" < maxValHeight.
	“GevHeartbeatTimeout”	Heartbeat packet timeout, unit ms, step size 1
	“GevSCPD”	Packet delay controls the delay (in timestamp counter units) inserted between each packet. This can be used as a rough flow control mechanism if the application or network infrastructure cannot keep up with packets coming from the device. Setting the packet sending delay can indirectly control the frame rate, step size 1
Floating-Point Type	“ExposureTimeAbs”	Camera exposure time, unit microsecond (us)
	“ResultingFrameRateAbs”	The maximum frame rate allowed in the given current area of interest (default is the whole image), exposure time and bandwidth, unit FPS/s. This parameter cannot be written and can only be read. The purpose is to view the camera frames in the current environment.

	“AcquisitionFrameRateAbs”	Set the image acquisition frame rate, unit FPS/s, range [1, 100000]. Before writing this parameter, “AcquisitionFrameRateEnable” parameter must be true, then the parameter can be written successfully.
	“Gamma”	Perform gamma correction on image pixel brightness, range [0, 4], < 1, improve image brightness, the smaller the value, the stronger the improvement. > 1, compress image brightness, the larger the value, the stronger the compression. Before writing this parameter, “GammaEnable” parameter must be true, then the parameter can be written successfully.

17.2.3.1. Dahua

Parameter Type	Parameter Name	Description
Command Type	“TriggerSoftware”	Under camera soft trigger mode, set soft trigger command parameters to trigger camera shooting, take photo once when triggered once.
Enumeration type	“PixelFormat”	Pixel format, the enumeration value of grayscale image is

		17301505 or the enumeration name is "Mono8", the enumeration value of RGB color image is 35127316 or the enumeration name is "RGB8Packed".
	“LineSelector”	Line selector, select the external wiring to be configured, that is, select an external wiring and configure some properties, such as configuring the external wiring as input or output properties. The enumeration value of external line Line0 is 0 or the enumeration name is "Line0", the enumeration value of external line Line1 is 1 or the enumeration name is "Line1", the enumeration value of external line Line2 is 2 or the enumeration name is "Line2". For the connection between camera wiring and external devices, please refer to the document "Dahua Camera IO Cable Connection Instructions"
	“LineMode”	Line mode controls whether the external wiring is used as an input or output signal. First use the "LineSelector" line selector to select a line, and then use the line mode to set it as an input or output

		signal. The enumeration value of the input signal is 0 or the enumeration name is "InPut", the enumeration value of the output signal is 1 or the enumeration name is "Output".
	“AcquisitionMode”	The image acquisition mode indicates whether the camera acquires a single frame image or a continuous frame image. The enumeration value for obtaining a single frame image is 1 or the enumeration name is "SingleFrame", indicating that the camera device will only obtain one frame of image. The enumeration value for obtaining continuous frame images is 0 or the enumeration name is "Continuous", indicating that the image will acquire frame image continuously.
	“TriggerMode”	Whether to enable trigger mode, that is, soft trigger or external trigger can only be used. The enumeration value for turning on trigger mode is 1, or the enumeration name is “On”. The enumeration value for turning off

		trigger mode is 0, or the enumeration name is “Off”.
	“TriggerSource”	Trigger source, that is, the source of the trigger signal in trigger mode. The enumeration value of soft trigger is 0 or the enumeration name is "Software", the enumeration value of external trigger Line1 is 2 or the enumeration name is "Line1", the enumeration value of external trigger Line2 is 3 or the enumeration name is " Line2".
Boolean Type	“GevGVCPHeartbeatDisable”	Heartbeat packet disable, 1-enabled, 0-disabled. It is usually necessary to disable the heartbeat packet during program debugging.
	“AcquisitionFrameRateEnable”	Image acquisition frame rate enablement, 1-enabled, 0-disabled, the frame rate can be set only after enabling it.
	“ReverseX”	Horizontal image reversion is enabled, that is, the image is flipped left and right with the vertical axis as the flip axis, 1-enabled, 0-disabled.
	“ReverseY”	Vertical image reversion is enabled, that is, the image is flipped up and down with the

		horizontal axis as the flip axis, 1-enabled, 0-disabled.
Integer Type	“OffsetX”	The offset in the X direction of the ROI, that is, the x coordinate of the upper left corner of the ROI. When setting "OffsetX", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the offset in advance to correctly set "OffsetX". You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetX" + "Width" < maxValWidth
	“OffsetY”	The offset in the Y direction of the ROI, that is, the y coordinate of the upper left corner of the ROI. When setting "OffsetY", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the offset in advance to correctly set "OffsetY". You need to use the corresponding SDK software to view it. Note: ROI

		cannot exceed the maximum image range, that is "OffsetY" + "Height" < maxValHeight
	“Width”	ROI image's width. When setting "Width", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the image width in advance to correctly set "Width". And different resolution images are with different widths. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetX" + "Width" < maxValWidth
	“Height”	ROI image's height. When setting "Height", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the image height in advance to correctly set "Height". And different resolution images are with different heights. You need to use the corresponding SDK

		software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetY" + "Height" < maxValHeight.
	“GevHeartbeatTimeout”	Heartbeat packet timeout, unit ms, range [500, 4294967295], step size 1
	“GevSCPD”	Packet delay controls the delay (in timestamp counter units) inserted between each packet. This can be used as a rough flow control mechanism if the application or network infrastructure cannot keep up with packets coming from the device. Setting the packet sending delay can indirectly control the frame rate, range [0, 4294967295], step size 1
Floating-Point Type	“ExposureTime”	Camera exposure time, unit microsecond (us), range [1, 1000000]
	“TriggerDelay”	Camera trigger delay, unit microsecond (us), range [0, 1000000]
	“AcquisitionFrameRate”	Set the image acquisition frame rate, range [1, 2000]. This parameter, Before writing “AcquisitionFrameRateEnable” parameter must be true, then the

		parameter can be written successfully.
	"Gamma"	Perform gamma correction on image pixel brightness, range [0, 3.99998], < 1, improve image brightness, the smaller the value, the stronger the improvement. > 1, compress image brightness, the larger the value, the stronger the compression.

17.2.3.2. MindVision

Parameter Type	Parameter Name	Description
Command Type	"TriggerSoftware"	Under camera soft trigger mode, set soft trigger command parameters to trigger camera shooting, take photo once when triggered once.
Enumeration type	"PixelFormat"	Pixel format, the enumeration value of grayscale image is 17301505 or the enumeration name is "Mono8", the enumeration value of RGB color image is 35127316 or the enumeration name is "RGB8Packed".
Boolean Type	"ReverseX"	Horizontal image reversion is enabled, that is, the image is flipped left and right with the

		vertical axis as the flip axis, 1-enabled, 0-disabled.
	“ReverseY”	Vertical image reversion is enabled, that is, the image is flipped up and down with the horizontal axis as the flip axis, 1-enabled, 0-disabled.
Integer Type	“OffsetX”	The offset in the X direction of the ROI, that is, the x coordinate of the upper left corner of the ROI. When setting “OffsetX”, you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the offset in advance to correctly set “OffsetX”. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is “OffsetX” + “Width” < maxValWidth
	“OffsetY”	The offset in the Y direction of the ROI, that is, the y coordinate of the upper left corner of the ROI. When setting “OffsetY”, you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum

		value: Maximum value: step increment]) of the offset in advance to correctly set “OffsetY”. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is “OffsetY” + “Height” < maxValHeight
	“Width”	ROI image's width. When setting “Width”, you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the image width in advance to correctly set “Width”. And different resolution images are with different widths. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is “OffsetX” + “Width” < maxValWidth
	“Height”	ROI image's height. When setting “Height”, you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step

		increment]) of the image height in advance to correctly set “Height”. And different resolution images are with different heights. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is “OffsetY” + “Height” < maxValHeight.
Floating-Point Type	“ExposureTime”	Camera exposure time, unit microsecond (us), range [1, 1000000]
	“TriggerDelay”	Camera trigger delay, unit microsecond (us), range [0, 1000000]
	“AcquisitionFrameRate”	Set the image acquisition frame rate, range [1, 2000]. This parameter, Before writing “AcquisitionFrameRateEnable” parameter must be true, then the parameter can be written successfully.

17.2.3.3. Do3Think

Parameter Type	Parameter Name	Description
Command Type	“TriggerSoftware”	Under camera soft trigger mode, set soft trigger command parameters to trigger camera

		shooting, take photo once when triggered once.
Enumeration type	“PixelFormat”	Pixel format, the enumeration value of grayscale image is 30, and the enumeration value of RGB color image is 10.
Boolean Type	“ReverseX”	Horizontal image reversion is enabled, that is, the image is flipped left and right with the vertical axis as the flip axis, 1-enabled, 0-disabled.
	“ReverseY”	Vertical image reversion is enabled, that is, the image is flipped up and down with the horizontal axis as the flip axis, 1-enabled, 0-disabled.
Integer Type	“OffsetX”	The offset in the X direction of the ROI, that is, the x coordinate of the upper left corner of the ROI. When setting “OffsetX”, you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the offset in advance to correctly set “OffsetX”. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum

		image range, that is “OffsetX” + “Width” < maxValWidth
	“OffsetY”	The offset in the Y direction of the ROI, that is, the y coordinate of the upper left corner of the ROI. When setting “OffsetY”, you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the offset in advance to correctly set “OffsetY”. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is “OffsetY” + “Height” < maxValHeight
	“Width”	ROI image's width. When setting “Width”, you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the image width in advance to correctly set “Width”. And different resolution images are with different widths. You need to use the corresponding SDK software to view it. Note: ROI

		cannot exceed the maximum image range, that is "OffsetX" + "Width" < maxValWidth
	"Height"	ROI image's height. When setting "Height", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the image height in advance to correctly set "Height". And different resolution images are with different heights. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetY" + "Height" < maxValHeight.
Floating-Point Type	"ExposureTime"	Camera exposure time, unit microsecond (us).
	"TriggerDelay"	Camera trigger delay, unit microsecond (us).

17.2.3.4. Daheng

Parameter Type	Parameter Name	Description
Command Type	"TriggerSoftware"	Under camera soft trigger mode, set soft trigger command parameters to trigger camera

		shooting, take photo once when triggered once.
Enumeration type	"PixelFormat"	Pixel format, the enumeration value of grayscale image is 17301505, and the enumeration value of RGB color image is 35127316.
	"LineSelector"	Line selector, select the external wiring to be configured, that is, select an external wiring and configure some properties, such as configuring the external wiring as input or output properties. The external line Line0 as the input, Line1 as the output, Line2 and Line3 can be as both input and output. The enumeration value of Line0 is 1, the enumeration value of Line1 is 2, the enumeration value of Line2 is 3, the enumeration value of Line3 is 4. For the connection between camera wiring and external devices, please refer to the document "Daheng Camera IO Cable Connection Instructions"
	"LineMode"	Line mode controls whether the external wiring is used as an input or output signal. First use the "LineSelector" line selector to

		select a line, and then use the line mode to set it as an input or output signal. The enumeration value of the input signal is 0, and the enumeration value of the output signal is 1.
	“TriggerMode”	Whether to enable trigger mode, that is, soft trigger or external trigger can only be used. The enumeration value for turning on the trigger mode is 1, the enumeration value for turning off the trigger mode is 0.
	“TriggerSource”	Trigger source, that is, the source of the trigger signal in trigger mode. The enumeration value for closing trigger source is 0, and the enumeration value for soft trigger source is 1. Only Line0, Line2, Line3 can be used as external trigger source. The enumeration value of Line0 is 1, the enumeration value of Line2 is 3, the enumeration value of Line3 is 4.
Boolean Type	“ReverseX”	Horizontal image reversion is enabled, that is, the image is flipped left and right with the vertical axis as the flip axis, 1-enabled, 0-disabled.

	"CammaEnable"	Gamma enable, 1-enabled, 0-disabled, only after enabling can the gamma correction operation of pixel brightness be performed
Integer Type	"OffsetX"	The offset in the X direction of the ROI, that is, the x coordinate of the upper left corner of the ROI. When setting "OffsetX", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the offset in advance to correctly set "OffsetX". You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetX" + "Width" < maxValWidth
	"OffsetY"	The offset in the Y direction of the ROI, that is, the y coordinate of the upper left corner of the ROI. When setting "OffsetY", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the offset in advance to correctly set "OffsetY".

		You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetY" + "Height" < maxValHeight
	“Width”	ROI image's width. When setting "Width", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the image width in advance to correctly set "Width". And different resolution images are with different widths. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetX" + "Width" < maxValWidth
	“Height”	ROI image's height. When setting "Height", you need to know the minimum, maximum, and incremental information ([minVal: maxVal: increment] = [minimum value: Maximum value: step increment]) of the image height in advance to correctly set "Height". And different resolution images are

		with different heights. You need to use the corresponding SDK software to view it. Note: ROI cannot exceed the maximum image range, that is "OffsetY" + "Height" < maxValHeight.
	“GevHeartbeatTimeout”	Heartbeat packet timeout, unit ms, range [500, 3600000], step size 1
	“GevSCPD”	Packet delay controls the delay (in timestamp counter units) inserted between each packet. This can be used as a rough flow control mechanism if the application or network infrastructure cannot keep up with packets coming from the device. Setting the packet sending delay can indirectly control the frame rate, range [0, 180000], step size 1
Floating-Point Type	“ExposureTime”	Camera exposure time, unit microsecond (us), range [36, 1000000]
	“TriggerDelay”	Camera trigger delay, unit microsecond (us), range [0, 3000000]
	“AcquisitionFrameRate”	Set the image acquisition frame, range [1, 10000]
	“CurrentAcquisitionFrameRate	Current frame rate, it only can be obtained, that is, it can't be set.

17.3. Error Codes

Error Code	Information	Remark
0	Normal	OK
8001	Undefined error	Undefined error
8002	Assertion error	Assert
8003	Null pointer	Null pointer
8004	C++ abnormal	C++ exception
8005	Std library abnormal	C++ exception1
8006	LIB library function execution abnormal	Libs error 1
8007	LIB library assertion fails	Libs error 2
8008	LIB library function execution error	Libs error
8009	SDK abnormal	SDK exception
8010	Running timeout	Timeout
8011	Zero-division error	Divide zero
8012	Array length is not matched	Vector size error
8013	Buffer area access out of range	Index out of range
8014	Buffer area length is not enough	Buffer overflow
8015	Object doesn't exist	Object not exist
8016	Unsupported: Input and output use the same buffer area	No inplace
8017	Error of ZVOBJECT type creating	ZVOBJECT create error
8018	Composite type that cannot be recognized.	Unknown type
8019	Error when converting multiple byte characters.	Multiple-byte character convert error
8020	Error when modifying system parameters	System param set error
8021	Error when reading system parameters	System param get error
8022	Internal parameters' values exceed	Param value out of range

8061	Value exceeds valid range	Value exceeds supported range
8091	Error of creating thread	Thread create error
8092	Error of exiting thread	Thread stop error
8093	Error of getting thread handle	Thread get id error
8095	Functions are not opened or not achieved	Not Implemented
8096	Unsupported function	Not supported
8097	Unsupported function	Not supported
8099	Unsupported function	Not supported
8400	Task parameter buffer area is not enough	Task param buffer overflow
8401	Task thread ID out of range	Task thread id error
8402	Task function ID out of range	Task function id error
8403	Task function is empty	Task function null
8404	The number of tasks exceed the max	Task thread num error
8420	Task communication creating fails	Task link create error
8421	Task communication connection fails	Task link error
8422	The number of communication channels is wrong	Task link channel error
8423	Task message buffer area length is not enough	Task message buffer overflow
8424	Task message receiving and sending sequence No. is not matched	Task message acq no. unmatch
8425	Error of task message sending	Task message send error
8426	Error of task message receiving	Task message recv error
8440	The number of ZV types out of range	ZVOBJECT number exceeded
8441	ZV types are used incorrectly (reused)	ZVOBJECT param cannot be reused
8480	Memory allocation fails	Memory alloc fail

8481	The allocated memory is corrupted	Memory corrupted
8500	File doesn't exist	File not exist
8501	Error of file opening	File open error
8502	Error of file saving	File save error
8503	Error of file format	File format error
8504	Version can't be compatible	File version incompatible
8505	Error of file reading	File read error
8506	Error of file writing	File write error
8507	Error of file positioning	File seek error
8508	Error of file correction	File checksum error
8509	Error of file expansion name	File extension error
8510	Path form is not matched	Path format error
8511	Path is empty	Path is empty
8512	File has been existed	File exist
8515	Error of creating directory	Create directory error
8516	Error of file finding	File Find error
8520	Error of acquisition	Camera grab error
8522	Camera scanning index out of range	Camera scan id out of range
8523	Camera index out of range	Camera id out of range
8524	Unbound camera is selected	Camera select unbound
8525	Error of device opening	Camera open error
8526	Error of camera scanning	Camera scan error
8527	Unsupported camera acquisition image format	Camera pixel format error
8528	Camera acquisition buffer area is not enough	Camera cache size error
8529	Camera is not found	Camera not found
8530	Acquisition fails	Camera grab fail
8531	Camera trigger mode is not matched	Camera trigger mode

		unmatch
8532	Error of camera closing	Camera close error
8533	The number of cameras exceeds	Camera count exceeds limit
8534	Camera is removed	Camera removed
8535	Camera status is unknown	Camera unknown
8536	Error of camera resources releasing	Camera release error
8537	Error of camera command execution	Camera command execute error
8538	Error of camera acquisition state	Camera not started
8539	Camera can't be used	Camera not Accessible
8551	Camera scanned type is conflict	Camera scan type conflict
8560	Unsupported camera parameter configuration	Camera param undefined
8561	Error of camera parameters reading	Camera param read error
8562	Error of camera parameters writing	Camera param write error
8563	Camera parameter name length exceeds	Camera param name length error
8564	Unsupported camera parameter node types	Camera param node type error
8565	Camera parameter access mode is not matched	Camera param access error
8567	Camera parameter values exceed	Camera param value error
8568	Camera parameter value types error	Camera param type error
8580	Error of camera library loading	Camera lib load error
8581	Error of camera library function getting	Camera lib load function error
8582	Error of camera library format	Camera lib format error
8583	The version camera library can't process	Camera lib version error
8584	Error of camera library initialization	Camera lib init error

	loading	
8585	Error of camera library uninstalling	Camera lib uninit error
8586	Some functions of camera library are empty	Camera lib function not found
8587	Camera is being used, camera library can't be uninstalled	Camera lib cannot be removed
8600	Parameters are empty	Param null
8601	Parameter 1 is empty	Param 1 null
8602	Parameter 2 is empty	Param 2 null
8603	Parameter 3 is empty	Param 3 null
8604	Parameter 4 is empty	Param 4 null
8605	Parameter 5 is empty	Param 5 null
8606	Parameter 6 is empty	Param 6 null
8607	Parameter 7 is empty	Param 7 null
8608	Parameter 7 is empty	Param 8 null
8609	Parameter 9 is empty	Param 9 null
8620	Error of parameter type	Param type error
8621	Error of parameter 1 type	Param 1 type error
8622	Error of parameter 2 type	Param 2 type error
8623	Error of parameter 3 type	Param 3 type error
8624	Error of parameter 4 type	Param 4 type error
8625	Error of parameter 5 type	Param 5 type error
8626	Error of parameter 6 type	Param 6 type error
8627	Error of parameter 7 type	Param 7 type error
8628	Error of parameter 8 type	Param 8 type error
8629	Error of parameter 9 type	Param 9 type error
8640	Parameter out of range	Param out of range
8641	Parameter 1 out of range	Parameter 1 out of range
8642	Parameter 2 out of range	Parameter 2 out of range

8643	Parameter 3 out of range	Parameter 3 out of range
8644	Parameter 4 out of range	Parameter 4 out of range
8645	Parameter 5 out of range	Parameter 5 out of range
8646	Parameter 6 out of range	Parameter 6 out of range
8647	Parameter 7 out of range	Parameter 7 out of range
8648	Parameter 8 out of range	Parameter 8 out of range
8649	Parameter 9 out of range	Parameter 9 out of range
8650	Parameter 10 out of range	Parameter 10 out of range
8651	Parameter 11 out of range	Parameter 11 out of range
8652	Parameter 12 out of range	Parameter 12 out of range
8653	Parameter 13 out of range	Parameter 13 out of range
8654	Parameter 14 out of range	Parameter 14 out of range
8655	Parameter 15 out of range	Parameter 15 out of range
8656	Parameter 16 out of range	Parameter 16 out of range
8657	Parameter 17 out of range	Parameter 17 out of range
8658	Parameter 18 out of range	Parameter 18 out of range
8659	Parameter 19 out of range	Parameter 19 out of range
8700	Size can't meet requirement	Size error
8701	Invalid size	Size invalid
8702	Size out of range	Empty
8703	Data is empty	Format error
8704	Unsupported data format	Dim error
8705	Dimension out of range	Size unmatch
8706	Input image or matrix size is not matched	Only 8-bit gray image
8740	Error of image format, only 8-bit channel is supported	Image data type unknown
8741	Unsupported or undefined image data type	Image channel error
8744	Error of image channel numbers	Image channel unmatch

8745	The number of channels is not matched	Image channel unmatch
8746	Error of source image channel numbers	Image source channel error
8748	Only support single channel image	Only gray image
8749	Image data needs to be aligned	Image alignment error
8750	Error of ROI size	Image ROI size error
8751	Image data type is not matched	Image data type unmatch
8752	Invalid image	Image invalid
8753	ROI out of range	ROI out of range
8780	Matrix multiplication size is not matched	Matrix multiplication size mismatch
8781	Matrix is not phalanx	Matrix not square
8782	Invalid matrix	Matrix invalid
8800	Contours or lengths referenced from contour sequences are not supported	Contour fixed size
8801	Contour length is zero	Contour size error
8802	Contour doesn't belong to polygon type	Contour is not polygon
8803	Contour doesn't belong to sequence type	Contour is not seq
8804	Unsupported contour type	Contour type not supported
8810	Invalid element segment	Segment element invalid
8830	Region is empty	Region empty
8860	List element type is not matched	List element type error
8861	Unsupported operation in specialized list	Not supported in special list
8862	Variable doesn't support general list inserting	Insert not supported for special element
8863	Unsupported operations of general list	Not supported in common list
8864	List element is empty	List element is NULL
8866	List size can't be 0	List size error
8870	Element can't insert 2 lists	Inserting two lists is not

		supported
8900	Unsupported color name	Undefined color name
8901	Color value out of range	Color value error
8902	Unsupported Marker type	Unknown marker type
8903	Error of font structure creating	Create font error
8904	Error of font loading	Load font error
8940	Filter size exceeds	Filter size out of range
8941	Error of filter offset	Filter anchor error
8942	Error of filter structure	Filter struct error
8970	Error of morphologic type	Morph type error
8971	Error of structural element shape	Morph shape error
8972	Error of structural element generation	Morph kernel create error
8973	Error of structural element	Morph kernel error
8974	Error of structural element size	Morph kernel size error
9000	Error of feature type	Feature type error
9001	Error of feature value calculation	Feature value error
9002	Error of moment order number	Moment order error
9003	Error of moment type	Moment type error
9051	The number of sample points for matching template is not enough	Edge invalid
9052	Abnormal template data	Modul error
9053	Shape template edge extraction fails	Extract edge error
9054	Error of auto-threshold value calculation	Auto threshold calculate error
9055	Error of matrix row numbers matching	Finds parameter matrix rows error
9056	Error of matrix column numbers matching	Finds parameter matrix cols error
9057	Error of interpolation data operation	Data for interpolate error

9090	Template data has not been generated	Shape-model data no pregenerate
9100	Measurement area and measurement functions are not matched	Measure type unmatch
9101	Invalid measurement area	Measure invalid
9102	The number of measurement points can't meet the lowest requirement	Measure points num error
9103	The X coordinate of the fitting point exceeds the image range	Measure X out of image
9104	The Y coordinate of the fitting point exceeds the image range	Measure Y out of image
9105	Subregion width exceeds the range	Measure sub width error
9106	Error of subregion numbers	Measure sub num error
9140	Error of circle measurement	Measure circle error
9141	Error of line measurement	Measure line error
9150	ocr sample is empty	Ocr sample null
9151	Error of ocr sample matching, the number of samples, sample classification	Ocr classification num not match
9152	Incorrect feature type	Ocr feature type error
9153	Data is empty when extracting feature	Ocr feature empty
9154	ocr horizontal projection is empty	Ocr char segment error
9155	ocr identifier is empty	Ocr null
9156	ocr identifier doesn't exist	Ocr not existence
9157	Invalid ocr	Ocr invalid
9158	Error of ocr identifier type	Ocr type error
9163	ocr training sample type is not enough	Ocr training sample class num error
9164	ocr training fails	Ocr training fail
9165	ocr sample doesn't remark	Ocr sample not marked
9200	Unsupported barcode type	Barcode type unsupport

9201	Error of signal configuration, or configured value exceeds the range	Barcode param invalid
9210	Error of reader related image	Code reader scan image error
9211	Error of decoder creating	Code reader create error
9240	Calibration target point is too less	Calib too few points
9241	Error of target point extraction	Extract points error
9242	Error of base standard coordinate system calculation	Calculate Datum CSYS error
9243	Error of base standard coordinate system	Datum CSYS error
9244	Error of calibration type	Calib type error
9245	Calibration doesn't support correction	Calib unsupport correct
9270	Defect type is not matched	Defect type not match
9280	Error of measurement type defect	Defect detector type error
9281	Abnormal defect handle measurement parameters	Defect detector measure parameter error
9282	Subregion of defect handle measurement is too less	Defect detector measure regions error
9500	The number of fitting points is not enough	Fitting data error
9501	Error of interpolation value result	Interpolation error
9502	Two points of line coincide	Coincidence points
9503	Error of transformation matrix	Transform matrix error
9504	Error of fitting calculation	Fitting calc error
9505	Point set shared line	Collinear
9506	The number of valid points is not enough	Valid points num error
9507	Error of internal calculation	Internal calc error
9508	Point set coincide	Coincidence
9509	Line 1 endpoint coincides	Not line1
9510	Line 2 endpoint coincides	Not line2

9950	Error of interface expansion dynamical library loading	Extension load dll error
9951	Error of interface expansion function getting	Extension get dll functions error
9952	Error of interface expansion initialization	Extension execute dll init error
9953	Interface expansion function is prohibited	Extension interface disabled
9954	The interface dependent library version is incompatible	Extension dll version error