FINGER CNC

Open Designs Customize The Future

Visual Function

User Manual

Version No: F202409VU-EN

Guangzhou Finger Technology Co., Ltd

CONTENTS

1. Document Overview

1.1 Revision History

Modification Date	**Note**
20240507	Initial Draft

1.2 About FINGER CNC

Guangzhou Finger Technology Co., Ltd. aims to develop high-performance open CNC systems that simplify automation development and make machine tool automation easily accessible. As one of China's high-performance controller manufacturers, Finger Technology focuses on customer needs, continuously pushing the boundaries of technological R&D to gradually form a comprehensive ecosystem of key automation technologies, providing customers with complete solutions and convenient services. We have established complete, professional, and efficient sales and service channels across various regions in China.

Finger Technology is dedicated to the R&D and production of CNC systems, motion controllers, edge computing controllers, Open CNC development platforms, CAD/CAM technologies, machine vision technologies, robotic control technologies, and industrial IoT technologies. Our industry-leading Open CNC development platform simplifies the customization of machinery and creates unique product value for our customers. Finger Technology proposes the integration of six core embedded technologies (motion control, HMI, PLC, machine vision, CAD/CAM, IoT) as part of our integrated product solutions, providing customers with optimal automation solutions. We have accumulated extensive product experience and customer base in industries such as turning and milling centers, grinding machines, spring machines, tool machines, woodworking machinery, winding machines, spinning machines, pipe bending machines, and 3C electronics, continuously striving for excellence.

In the field of high-speed and high-precision, Finger Technology conducts in-depth research on various high-performance motion control algorithms, widely applicable to different industry needs. Especially in multi-axis linkage interpolation, RTCP five-axis linkage control, multi-axis multi-channel control, electronic cam, winding, and tension control technologies, we provide customers with a wider range of solutions.

Focusing on customer needs, emphasizing results, pursuing excellence in innovation, and respecting talent have been the core principles and values of Finger Technology since its inception. We remain dedicated to these principles, diligently advancing and consistently developing reliable, user-friendly automation products. By extending the Open CNC concept to customer endpoints, we create exclusive value for our clients.

1.3 Preface

The B series controllers and above are equipped with embedded vision functionality, offering two methods of implementation: a C++ open interface and Python scripting. This document primarily focuses on how to use the C++ open interface to build embedded vision functions that meet production requirements.

This document serves as an introductory training manual for users who are new to vision functionality and as a reference guide for those with questions about specific interfaces or features during use. Each interface in the document is accompanied by a usage example, which users can copy directly into the HMI software to run and observe the results. For users who are unsure about the functionality of an interface, it is recommended to experiment with different detection objects and parameters based on the provided examples to compare the outcomes.

1.4 Information to Confirm Before Use

The embedded vision function is primarily developed through the HMI software during the development of screen projects. Users can call the relevant vision interfaces according to actual production requirements to complete the development of vision functionality. To ensure smooth development and usage, please confirm the following information before proceeding:

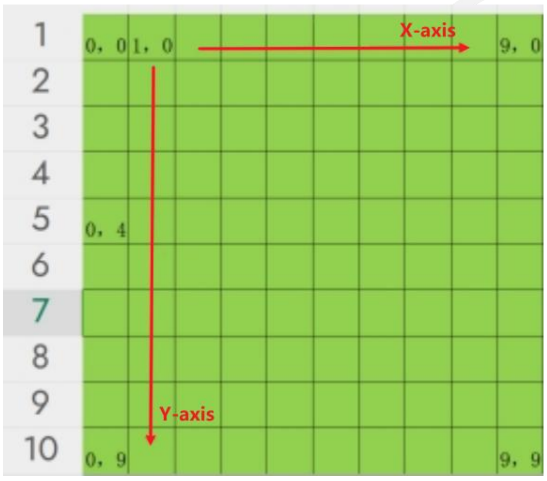
1. Ensure that the controller in use is a B series controller or higher.
2. Verify that the HMI software includes the HCamera module for camera operations and the HVisionModule for vision processing within the macro wizard.
3. Confirm that the vision function on the controller is enabled. The vision function must be activated through the Axis software. You can check variable com40783; when set to 1, the vision function is enabled, and when set to 0, it is disabled and cannot be used.
4. Verify whether you are using an Ethernet camera or a USB camera. If you are using an Ethernet camera, ensure that the Ethernet cable is directly connected between the camera and the controller's Ethernet port (connection via a switch is also acceptable). Using a USB-to-Ethernet adapter may lead to connection errors. If you are using a USB camera, make sure that the USB end of the camera is connected to the USB 3.0 port of the controller, not a USB 2.0 port, as this could cause instability in the connection.
5. When using an Ethernet camera, check if the camera's IP address and the controller's IP address are within the same subnet; otherwise, the connection will fail. You can use the HCamera.getIpList() and HCamera.setIp(n, ip) interfaces to retrieve and set the camera's IP address.

1.5 Basic Concepts of Vision

1. Cameras are divided into color cameras and monochrome cameras. Color cameras

capture images with three color channels (R, G, B), while monochrome cameras capture images with only one color channel (Gray), so grayscale images captured by monochrome cameras do not have color.

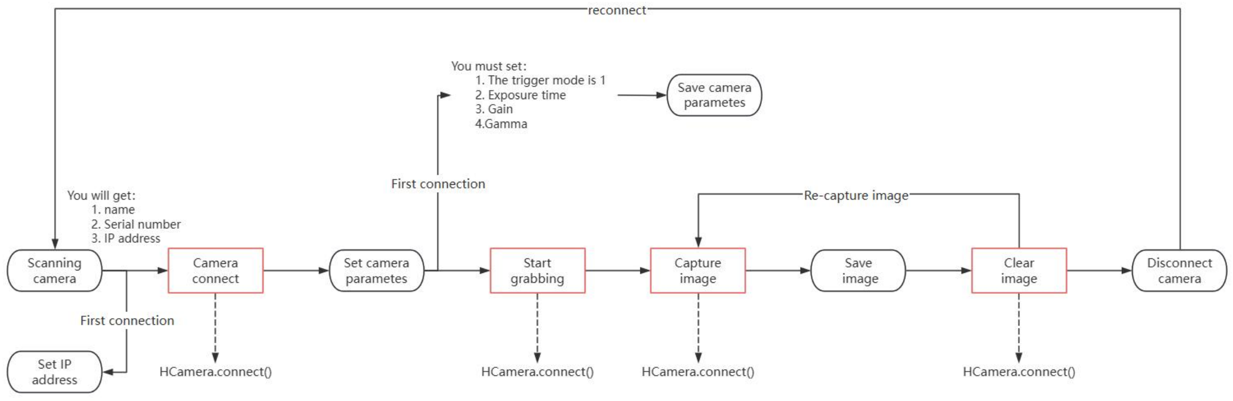
2. The most commonly used cameras are monochrome cameras. The grayscale images captured can be understood as a 2D array (or 2D matrix) of WH. For example, a 12-megapixel monochrome camera has a resolution of 4000 (W) * 3000 (H), determined by the camera's hardware (40003000 = 12 million, hence it is called a 12 MP camera).
3. In the field of vision, the top-left corner of the image is the origin, with coordinates (0, 0). The horizontal direction to the right is the positive X direction, and the vertical direction downward is the positive Y direction. The following image illustrates this with a resolution of 10*10:



4. In grayscale images, each pixel stores a "gray level," which is an integer between 0 and 255. When the gray level is 0, the pixel displays black; when the gray level is 255, the pixel displays white. Values between 0 and 255 represent varying shades of gray from black to white.
5. In the HCamera and HVisionModule modules, the term "image" refers to a variable as a whole. Whether reading an "image," capturing an "image," or receiving an "image" as a return type from an interface, memory usage will increase based on the size of the image. Therefore, memory management should be carefully handled when calling interfaces (using the clearImage function from the corresponding module).

6. Before using the HCamera module, please understand the following flow and functions in the diagram:

Note: The normal image acquisition flow is shown in the diagram. Red rectangles indicate required interfaces, while black rounded rectangles indicate optional interfaces. When capturing images, please confirm the trigger mode.



2. Camera Module Introduction

2.1 Camera Function Preview (Ctrl+Click to Navigate)

- Get Camera Name
- Get Camera Serial Number
- Get Ethernet Camera IP Address
- Set Ethernet Camera IP Address
- Connect to Camera
- Disconnect from Camera
- Get Camera Connection Status
- Get Camera Exposure Time
- Set Camera Exposure Time
- Get Camera Gain
- Set Camera Gain

2.2 Camera Properties Introduction

Camera	Explanation

Properties Introduction	
Exposure Time	Refers to the time interval from when the shutter opens to when it closes. A longer exposure time results in higher image brightness. You can adjust the exposure time based on the image's brightness level.
Camera Gain	Refers to the process of increasing the sensor signal strength to enhance image brightness during capture. However, increasing gain can also introduce image noise, affecting image quality. Therefore, gain should be chosen carefully to balance image brightness and noise levels for optimal results.
Trigger Mode	Include the following: Continuous Trigger: The camera captures image data continuously at a set frequency, providing real-time display. Software Trigger: The camera captures images based on software commands or signals. Hardware Trigger: The camera captures images based on external trigger signals or devices.
Camera Frame Rate	Refers to the number of continuous images the camera captures per second, measured in fps (frames per second).
Start/Stop Streaming	When streaming starts, the camera is in a standby state, waiting for the capture trigger command. Before modifying attributes such as pixel width, height, or offset values, streaming needs to be stopped. Streaming must be started before capturing images.
Pixel Width	Refers to the width of the image resolution captured by the camera. For example, in a 4000*3000 resolution image, the pixel width is 4000.
Pixel Height	Refers to the height of the image resolution captured by the camera.

	For example, in a 4000*3000 resolution image, the pixel height is 3000.
Pixel Offset X	When the image's pixel width is cropped, the pixel offset X can be adjusted to control the horizontal shift of the field of view.
Pixel Offset Y	When the image's pixel height is cropped, the pixel offset Y can be adjusted to control the vertical shift of the field of view.

2.3 Camera Interface Function Details, Usage Examples, and Effects

2.3.1 Get Camera Name

[Interface]

var list = HCamera.getDeviceList()

[Description]

Retrieves the list of camera model names. (Note: The scan list cannot be updated when a camera is connected.)

[Parameters]

None

[Return Value]

(array) A list of model names on the device.

[Usage Example]

```
var list    = HCamera.getDeviceList()
var number = list.length
print("Number of cameras： ", number)
for(i = 0; i < number; i++)
{
        str1 = "camera" + i + ":"
```

```
    print(str1 + list[i])
}
```

[Execution Effect]

If the camera is not found:

```
Number of cameras： 0
```

else：

```
Number of cameras： 1
camera0： MV-CS050-10UM
```

Note: This and the following 'print' statements are background prints for reference only.

2.3.2 Get Camera Serial Number

[Interface]

var list = HCamera.getSnList()

[Description]

Retrieve serial numbers from the camera list. (The scan list cannot be updated when a camera is connected.)

[Parameters]

None

[Return Value]

(array) Returns a list of serial numbers for the devices.

[Usage Example]

```
var list   = HCamera.getSnList()
var number = list.length
print("Number of cameras： ", number)
for(i = 0; i < number; i++)
{
    str1 = "camera" + i + ":"
    print(str1 + list[i])
}
```

[Execution Effect]

```
Number of cameras： 1
camera0： U3V:DA0888839
```

2.3.3 Get Ethernet Camera IP Address

[Interface]

var list = HCamera.getIpList()

[Description]

Retrieve IP addresses from the camera list. (The scan list cannot be updated when a camera is connected.)

[Parameters]

None

[Return Value]

(array) Returns the IP addresses of the device list. Applicable only to Ethernet cameras.

[Usage Example]

```
var list   = HCamera.getIpList()
var number = list.length
print("Number of cameras： ", number)
for(i = 0; i < number; i++)
{
    str1 = "camera" + i + ":"
    print(str1 + list[i])
}
```

[Execution Effect]

```
Number of cameras： 1
camera0： 192.168.2.97
```

2.3.4 Set Ethernet Camera IP Address

[Interface]

var flag = HCamera.setIp(n, ip)

[Description]

Set the IP address of the camera. This cannot be used after the camera is connected. Applicable only to Ethernet cameras.

[Parameters]、

(int)n: The index of the device (starting from 0).

(string)n: IP address, e.g., ip = "192.168.3.1".

[Return Value]

(bool) Returns true if successful, false if failed.

[Usage Example]

```
var list   = HCamera.getIpList()
var number = list.length
print("Number of cameras：", number)
for(i = 0; i < number; i++)
{
      str1 = "camera" + i + ":"
      print(str1 + list[i])
}
var flag = HCamera.setIp(0, "192.168.3.1")
print("IP set success flag = ", flag)
```

[Execution Effect]

```
Number of cameras： 1
camera0： 192.168.2.97
IP set success flag = true
```

Note: The Camera IP can be set to different subnets, but the controller and the camera need to be on the same subnet for the camera to connect.

2.3.5 Connect to Camera

[Interface]

var flag = HCamera.connect(n)

[Description]

Starts the connection to the camera.

[Parameters]

(int)n: The index of the device (starting from 0).

[Return Value]

(bool) Returns true if successful, false if failed.

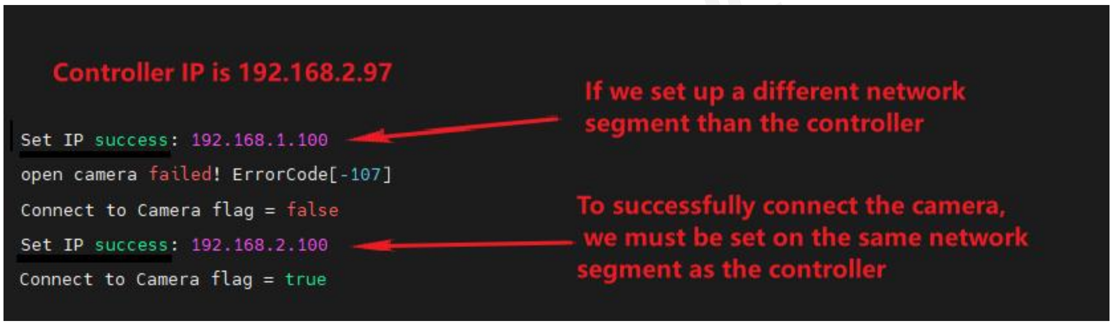
[Usage Example]

```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
```

[Execution Effect]

```
Connect to Camera flag = true
```

Note: When connecting to an Ethernet camera, ensure that the camera's IP and the controller's IP are on the same subnet.



```
Set IP success: 192.168.1.100
open camera failed! ErrorCode[-107]
Connect to Camera flag = false
Set IP success: 192.168.2.100
Connect to Camera flag = true
```

Controller IP is 192.168.2.97
If we set up a different network segment than the controller
To successfully connect the camera, we must be set on the same network segment as the controller

2.3.6 Disconnect from Camera

[Interface]

var flag = HCamera.disconnect(n)

[Description]

Disconnect from Camera

[Parameters]

(int)n: The index of the device (starting from 0).

[Return Value]

(bool) Returns true if successful, false if failed.

[Usage Example]

```
var flag = HCamera.connect(0)

print("Connect to Camera flag = ", flag)

flag = HCamera.disconnect(0)

print("Disconnect to Camera flag = ", flag)
```

[Execution Effect]

```
Connect to Camera flag = true
Disconnect to Camera flag = true
```

2.3.7 Get Camera Connection Status

[Interface]

var flag = HCamera.getCameraState(n)

[Description]

Get Camera Connection Status

[Parameters]

(int)n: The index of the device.

[Return Value]

(bool) Returns true if online, false if offline.

[Usage Example]

```
var flag = HCamera.getCameraState(0)

print("Get Camera Status flag = ", flag)


flag = HCamera.connect(0)

print("Connect to Camera flag = ", flag)


flag = HCamera.getCameraState(0)

print("Get Camera Status flag = ", flag)


flag = HCamera.disconnect(0)

print("Disconnect Camera flag = ", flag)


flag = HCamera.getCameraState(0)
```

[Execution Effect]

```
Get Camera Status flag = false
Connect to Camera flag = true
Get Camera Status flag = true
Disconnect to Camera flag = true
Get Camera Status flag = false
```

2.3.8 Get Camera Exposure Time

[Interface]

var time = HCamera.getExposureTime(n)

[Description]

Retrieves the camera's exposure time, in microseconds (μs).

[Parameters]Camera Exposure Time is

(int)n: The index of the device (starting from 0).

[Return Value]

(double) Returns the camera's exposure time in microseconds (μs). Returns -1 if retrieval fails.

[Usage Example]

```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
var time = HCamera.getExposureTime(0)
print("Camera Exposure Time is：", time)
```

[Execution Effect]

```
Connect to Camera flag = true
Camera Exposure time is： 9000
```

2.3.9 Set Camera Exposure Time

[Interface]

var flag= HCamera.setExposureTime(n, dExposureTime)

[Description]

Set Camera Exposure Time(in μs)。

[Parameters]

(int)n: The index of the device (starting from 0).

(double)dExposureTime: Exposure time (in μs).

[Return Value]

(bool) Returns true if successful, false if failed.

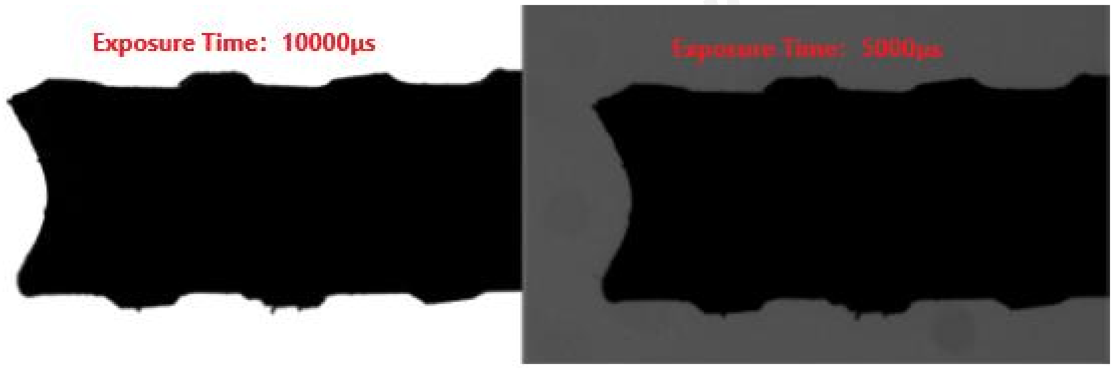
[Usage Example]

```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
var flag2 = HCamera.setExposureTime(0, 5000)
print("Modify Exposure Time flag2 = ", flag2)
```

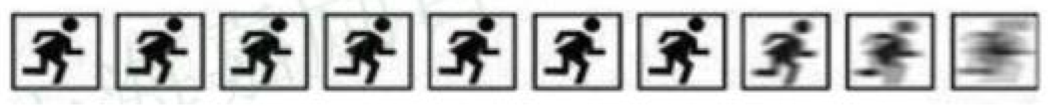
[Execution Effect]

```
Connect to Camera flag = true
Modify Exposure Time flag2 = true
```

Note: A longer exposure time results in higher image brightness. The effect is shown in the following image.



Additionally, increasing the exposure time will reduce the frame rate and decrease the capture speed, so it is generally not advisable to set it too high. Moreover, when photographing moving objects, a longer exposure time can cause the image to become blurred or show motion trails, as illustrated in the following image.



Supplementary Professional Knowledge:

Assuming the object's speed is 0.5 m/s, the camera has a resolution of 640x480, and the field of view is 4x3 mm:

1. 3 mm / 480 = 0.006 mm per pixel, so the pixel equivalent is 0.006 mm.
2. If the object's movement during the exposure time exceeds 1.5 pixels, motion blur can occur.

To avoid motion blur:t(exposure time)=0.006*1.5/0.5=0.018s = 18ms

2.3.10 Get Camera Gain

[Interface]

var gain = HCamera.getGain(n)

[Description]

Get Camera Gain

[Parameters]

(int)n: The index of the device (starting from 0).

[Return Value]

(double)The camera gain, -1 indicates retrieval failure.

[Usage Example]

```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
var gain = HCamera.getGain(0)
print("Camera Gain Value is：", gain)
```

[Execution Effect]

```
Connect to Camera flag = true
Camera Gain Value is：  2
```

2.3.11 Set Camera Gain

[Interface]

var flag = HCamera.setGain(n, dGain)

[Description]

Set Camera Gain

[Parameters]

(int)n: The index of the device (starting from 0).

(double)dGain: The camera gain.

[Return Value]

(bool) Returns true if successful, false if failed.

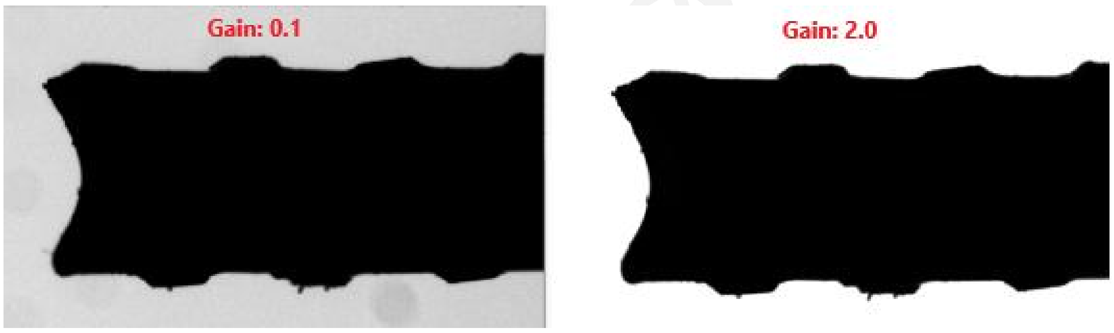
[Usage Example]

```
var flag = HCamera.connect(0)

print("Connect to Camera flag = ", flag)

var flag2 = HCamera.setGain(0, 1)

print("Modify Gain flag2 = ", flag2)
```

[Execution Effect]

```
Connect to Camera flag = true
Modify Gain flag2 = true
```

Gain is essentially an image effect amplifier within the camera that can multiply the image effect. It is used to increase the image brightness in low-light conditions, but increasing the gain also amplifies image noise. Therefore, gain is generally set to the default value of 1.



2.3.12 Get Camera Trigger Mode

[Interface]

var trig = HCamera.getTrigMode(n)

[Description]

Get Camera Trigger Mode

[Parameters]

(int)n: The index of the device (starting from 0).

[Return Value]

(int) Returns the camera trigger mode:

-1 indicates retrieval failure

0 indicates continuous trigger

1 indicates software trigger

2 indicates hardware trigger

[Usage Example]

```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
var trig = HCamera.getTrigMode(0)
print("Camera Trigger Mode is： ", trig )
```

[Execution Effect]

```
Connect to Camera flag = true
Camera Trigger Mode is： 1
```

2.3.13 Set Camera Trigger Mode

[Interface]

var flag = HCamera.setTrigMode (n, trig)

[Description]

Sets the camera's trigger mode. Options are 0 (continuous trigger), 1 (software trigger), or 2 (hardware trigger).

[Parameters]

(int)n: The index of the device (starting from 0).

(int)trigType: The camera trigger mode, where 0 = continuous trigger, 1 = software trigger, 2 = hardware trigger.

[Function]

Sets the camera's trigger mode.

[Return Value]

(bool) Returns true if successful, false if failed.

[Usage Example]

```
var flag = HCamera.connect(0)

print("Connect to Camera flag = ", flag)

var flag2 = HCamera.setTrigMode (0, 1)

print("Modify Trigger Mode flag2 = ", flag2)
```

[Execution Effect]

```
Connect to Camera flag = true
Modify Trigger Mode flag2 = true
```

1. Note:
2. The interface allows you to set three trigger modes: continuous trigger, software trigger (also known as single-shot trigger), and hardware trigger. Image acquisition is closely related to the trigger mode. The image capture interface only performs the "image acquisition," while the complete photo process includes "trigger command" + "exposure, transmission" + "image acquisition." Therefore, choosing the correct trigger mode is crucial for the project workflow.
3. In continuous trigger mode, the camera autonomously performs continuous "trigger command + exposure + transmission" actions at a frequency defined by the frame rate, leading to high CPU resource usage. The advantage is fast acquisition speed.
4. In software trigger mode, the camera waits for the capture command from the image acquisition interface HCamera.getImage(n) and then performs "trigger command + exposure + transmission + image acquisition." This results in lower CPU resource usage but longer time compared to continuous triggering.
5. In hardware trigger mode, an external IO signal is used to trigger the camera. Each IO trigger results in one "trigger command + exposure + transmission" action, followed by "image acquisition" using the HCamera.getImage(n) interface.

2.3.14 Capture Image

[Interface]

var image = HCamera.getImage (n)

[Description]

Single-frame capture, triggers the camera to capture one image (requires the camera trigger mode to be 0 or 1).

[Parameters]

(int)n: The index of the device (starting from 0).

[Return Value]

(array) QVariant containing the captured image; NULL if failed.

[Usage Example]

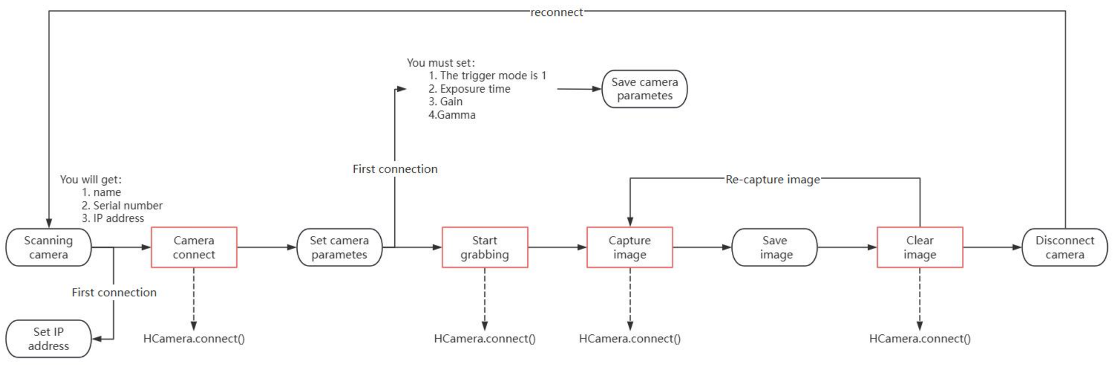
```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
flag = HCamera.startGrabbing(0)
print("Start Grabbing flag = ", flag )
var image = HCamera.getImage (0)
var flag2 = HCamera.saveImage(image , "/home/image1.bmp")
print("Save Image flag2 = ", flag2)
var flag3 = HCamera.clearImage(image)
print("Clear Image flag3 = ", flag3)
```

[Execution Effect]

```
Connect to Camera flag = true
Start Grabbing flag = true
Save Image flag2 = true
Clear Image flag3 = true
```

Note:

The normal image capture process is illustrated below: Red rectangles indicate required interfaces, while black rounded rectangles represent optional interfaces. Please ensure the trigger mode is correctly set when capturing images.



2.3.15 Save Image

[Interface]

var image = HCamera.saveImage(img, path, imageFormat = null, quality = -1)

[Description]

Saves an image returned by the getImage interface. The path can be created automatically.

[Parameters]

(array)img: The image variable returned by the getImage interface that needs to be saved.

(string)path: The path where the image will be saved, e.g., "/home/root/hust/usr/test/test.bmp".

(string)imageFormat="null": The format of the saved image, such as "BMP", "JPG", "PNG", etc. Defaults to null if the format is included in the path, and generally does not need to be specified.

(int)quality=-1: The quality of the saved image, ranging from 1 (lowest) to 100 (highest). Typically, set to -1, representing 100. Defaults to this value and generally does not need to be specified.

[Return Value]

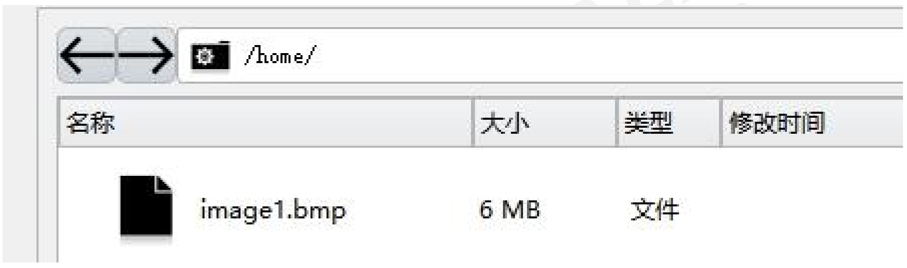
(bool) Returns true if successful, false if failed.

[Usage Example]

```
var flag = HCamera.connect(0)

print("Connect to Camera flag = ", flag)

flag = HCamera.startGrabbing(0)

print("Start Grabbing flag = ", flag )

var image = HCamera.getImage (0)

var flag2 = HCamera.saveImage(image , "/home/image1.bmp")

print("Save Image flag2 = ", flag2)

var flag3 = HCamera.clearImage(image)

print("Clear Image flag3 = ", flag3)
```

[Execution Effect]

```
Connect to Camera flag = true
Start Grabbing flag = true
Save Image flag2 = true
Clear Image flag3 = true
```


/home/

名称	大小	类型	修改时间
image1.bmp	6 MB	文件	

2.3.16 Get Frame Rate Enable Status

[Interface]

var flag = HCamera.getFrameRateEnable(n)

[Description]

Gets the frame rate enable status of the camera.

[Parameters]

(int)n: The index of the device (starting from 0).

[Return Value]

(bool) Returns the enable status.

[Usage Example]

```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
var flag2= HCamera.getFrameRateEnable(0)
print("Camera frame rate enable status flag2 = ", flag2)
```

[Execution Effect]

```
Connect to Camera flag = true
Camera frame rate enable status flag2 = false
```

2.3.17 Set Frame Rate Enable Status

[Interface]

var flag = HCamera.setFrameRateEnable(n, enable)

[Description]

Sets the frame rate enable status of the camera.

[Parameters]

(int)n: The index of the device (starting from 0).

(bool)enable: Enable status.

[Return Value]

(bool) Returns true if successful, false otherwise.

[Usage Example]

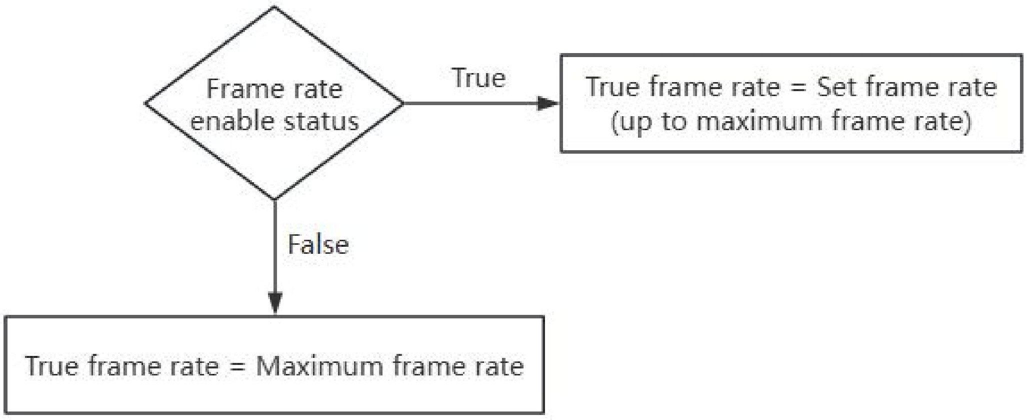
```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
var flag2 = HCamera.setFrameRateEnable(0, 1)
print("Set Camera frame rate enable status success flag2 = ", flag2)
```

[Execution Effect]

```
Connect to Camera flag = true
Set Camera frame rate enable status success flag2 = false
```

Note:

Setting the frame rate enable status determines whether the user-configured frame rate is effective. If the enable status is turned off, the camera will capture and transmit at the maximum frame rate by default. The frame rate refers to how many continuous frames the camera captures in one second, measured in fps (frames per second).


Frame rate enable status
True
True frame rate = Set frame rate (up to maximum frame rate)
False
True frame rate = Maximum frame rate

2.3.18 Get Frame Rate Value

[Interface]

var value = HCamera.getFrameRate(n)

[Description]

Get the camera's frame rate.

[Parameters]

(int)n: The device index (starting from 0).

[Return Value]

(double): The current frame rate value set for the camera.

[Usage Example]

```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
```

var value = HCamera.getFrameRate(0)

print("Get Camera frame rate value = ", value)

[Execution Effect]

```
Connect to Camera flag = true
Get Camera frame rate value = 60
```

2.3.19 Set Camera Frame Rate

[Interface]

var flag = HCamera.setFrameRate(n, value)

[Description]

Set the frame rate of the Camera.

[Parameters]

(int)n: The device index (starting from 0).

(double)value: The frame rate of the Camera.

[Return Value]

(bool) Returns true if successful, false if failed.

[Usage Example]

var flag = HCamera.connect(0)

print("Connect to Camera flag = ", flag)

var flag2 = HCamera.setFrameRate(0, 30)

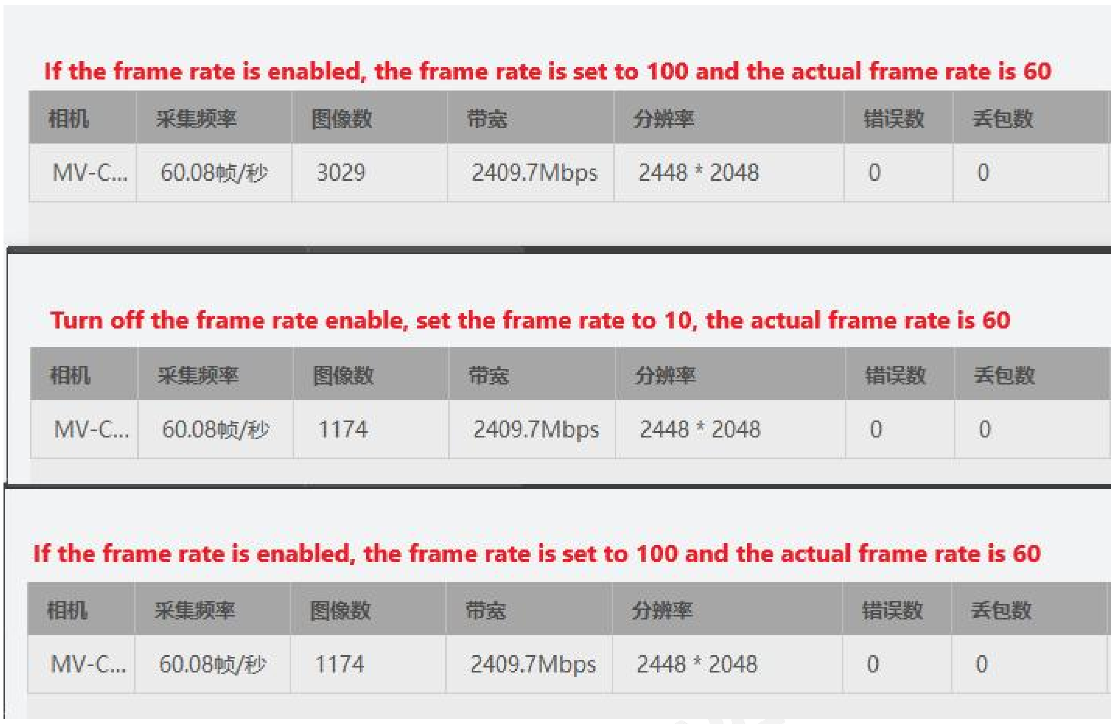
print("Set Camera frame rate flag2 = ", flag2)

[Execution Effect]

```
Connect to Camera flag = true
Set Camera frame rate flag2 = true
```

Note:

Frame rate settings can be set to any value, but the actual frame rate will not exceed the Camera's maximum frame rate, which is typically within 100 fps. Additionally, the user-set frame rate will only take effect if the frame rate enable status is active. You can check the frame rate enable status using the relevant interface for details.



If the frame rate is enabled, the frame rate is set to 100 and the actual frame rate is 60

相机	采集频率	图像数	带宽	分辨率	错误数	丢包数
MV-C...	60.08帧/秒	3029	2409.7Mbps	2448 * 2048	0	0

Turn off the frame rate enable, set the frame rate to 10, the actual frame rate is 60

相机	采集频率	图像数	带宽	分辨率	错误数	丢包数
MV-C...	60.08帧/秒	1174	2409.7Mbps	2448 * 2048	0	0

If the frame rate is enabled, the frame rate is set to 100 and the actual frame rate is 60

相机	采集频率	图像数	带宽	分辨率	错误数	丢包数
MV-C...	60.08帧/秒	1174	2409.7Mbps	2448 * 2048	0	0

2.3.20 Start Grabbing

[Interface]

var flag = HCamera.startGrabbing(n)

[Description]

Start grabbing. The Camera needs to start grabbing to take pictures. You need to stop grabbing before saving parameters.

[Parameters]

(int)n: The device number (starting from 0).

[Return Value]

(bool) Returns true if successful, false if failed.

[Usage Example]

var flag = HCamera.connect(0)

print("Connect to Camera flag = ", flag)

```
var flag2 = HCamera.startGrabbing(0)

print("Start grabbing flag2 = ", flag2)
```

[Execution Effect]

```
Connect to Camera flag = true
Start grabbing flag2 = true
```

Note:

Before executing the image capture interface, you must start grabbing first.

2.3.21 Stop Grabbing

[Interface]

var flag = HCamera.stopGrabbing(n)

[Description]

Stop grabbing. The Camera needs to start grabbing to take pictures. You need to stop grabbing to save parameters.

[Parameters]

(int)n: The index of the device (starting from 0)

[Return Value]

(bool) Returns true if successful, false if failed

[Example]

flag = HCamera.stopGrabbing(0)

[Usage Example]

```
var flag = HCamera.connect(0)

print("Connect to Camera flag = ", flag)

var flag2 = HCamera.startGrabbing(0)

print("Start grabbing flag2 = ", flag2)

var flag3 = HCamera.stopGrabbing(0)

print("Stop grabbing flag3 = ", flag3)
```

[Execution Effect]

```
Connect to Camera flag = true
Start grabbing flag2 = true
Stop grabbing flag3 = true
```

2.3.22 Get Pixel Width

[Interface]

var width = HCamera.getPixelWidth(n)

[Description]

Get the pixel width of the Camera.

[Parameters]

(int)n: The index of the device (starting from 0).

[Return Value]

(int) Returns the current pixel width of the Camera if successful, 0 if failed.

[Usage Example]

```
var flag = HCamera.connect(0)

print("Connect to Camera flag = ", flag)

var width = HCamera.getPixelWidth(0)

print("Camera pixel width = ", width )
```

[Execution Effect]

```
Connect to Camera flag = true
Camera pixel width = 3072
```

2.3.23 Set Pixel Width

[Interface]

var flag = HCamera.setPixelWidth(n, value)

[Description]

Sets the pixel width of the Camera.

[Parameters]

(int)n: The index of the device (starting from 0).

(int)value: The pixel width to set for the Camera.

[Return Value]

(bool) Returns true if successful, false if failed.

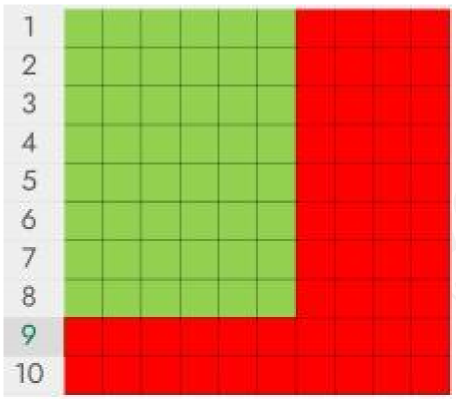
[Usage Example]

```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
var flag2 = HCamera.setPixelWidth(0, 1000)
print("Set Camera pixel width flag2   =   ", flag2 )
```

[Execution Effect]

```
Connect to Camera flag = true
Set Camera pixel width flag2 = true
```

Suppose the Camera's pixel resolution is 10x10, with a maximum pixel width of 10. This interface allows you to set the enabled pixel width. For example, setting the pixel width and height to 6x8 will result in an effective image area shown in the green region in the diagram, with a final image resolution of 6x8. This interface can be used to adjust the Camera's field of view and image size without affecting the precision.



2.3.24 Get Camera Pixel Height

[Interface]

var height = HCamera.getPixelHeight(n)

[Description]

\Retrieve the camera's pixel height.

[Parameters]

(int) n: The index of the device (starting from 0).

[Return Value]

(int) Returns the current pixel height of the camera if successful, otherwise returns 0.

[Usage Example]

```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
var height = HCamera.getPixelHeight(0)
print("Camera pixel height = ", height)
```

[Execution Effect]

```
Connect to Camera flag = true
Camera pixel height = 2048
```

2.3.25 Set Camera Pixel Height

[Interface]

var flag = HCamera.setPixelHeight(n, value)

[Description]

Set the pixel height of the camera.

[Parameters]

(int) n: The index of the device (starting from 0).

(int) value: The pixel height to set for the camera, used to crop the image size.

[Return Value]

(bool) Returns true if successful, false if failed.

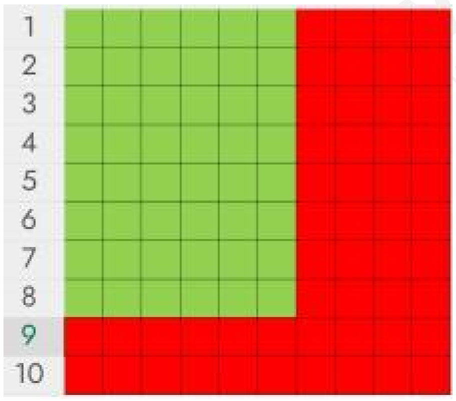
[Usage Example]

```
var flag = HCamera.connect(0)

print("Connect to Camera flag = ", flag)

var flag2 = HCamera.setPixelHeight(0, 1000)

print("Set Camera pixel height flag2 = ", flag2)
```

[Execution Effect]

```
Connect to Camera flag = true
Camera pixel height = 2048
```

Suppose the Camera has a pixel resolution of 10x10, the maximum pixel height is 10. You can use this interface to set the enabled pixel height. As shown in the image below, if you set the pixel width and height of a 10x10 Camera to 6x8, the effective area during image capture will be the green area in the image, resulting in an image resolution of 6x8. This interface can be used to change the Camera's field of view and image size without affecting accuracy.



2.3.26 Get Maximum Pixel Width

[Interface]

var maxWidth = HCamera.getPixelMaxWidth(n)

[Description]

Retrieve the maximum pixel width of the Camera.

[Parameters]

(int) n: The index of the device.

[Return Value]

(int) Returns the maximum pixel width of the Camera if successful; returns 0 if failed.

[Usage Example]

```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
var maxWidth = HCamera.getPixelMaxWidth(0)
print("Maximum Pixel Width of the Camera maxWidth = ", maxWidth )
```

[Execution Effect]

```
Connect to Camera flag = true
Maximum Pixel Width of the Camera maxWidth =3072
```

Note: The maximum pixel dimensions of each Camera are fixed and determined by the hardware.

2.3.27 Get Maximum Pixel Height

[Interface]

var maxHeight = HCamera.getPixelMaxHeight(n)

[Description]

Retrieve the maximum pixel height of the Camera.

[Parameters]

(int) n: The index of the device.

[Return Value]

(int) Returns the maximum pixel height of the Camera if successful; returns 0 if failed.

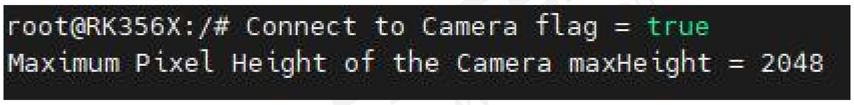
[Usage Example]

```
var flag = HCamera.connect(0)
```

```
print("Connect to Camera flag = ", flag)
var maxHeight = HCamera.getPixelMaxHeight(0)
print("Maximum Pixel Height of the Camera maxHeight = ", maxHeight )
```

[Execution Effect]

```
Connect to Camera flag = true
Maximum Pixel Height of the Camera maxHeight = 2048
```



```
root@RK356X:/# Connect to Camera flag = true
Maximum Pixel Height of the Camera maxHeight = 2048
```

Note: The maximum pixel dimensions of each Camera are fixed and determined by the hardware.

2.3.28 Get Pixel Offset X

[Interface]

var offsetX = HCamera.getOffsetX(n)

[Description]

Retrieve the pixel offset in the X direction of the Camera.

[Parameters]

(int) n: The index of the device.

[Return Value]

(int) Returns the current pixel offset in the X direction of the Camera if successful; returns -1 if failed.

[Usage Example]

```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
var offsetX = HCamera.getOffsetX(0)
print("Pixel Offset X of the Camera offsetX = ", offsetX )
```

[Execution Effect]

```
Connect to Camera flag = true
Pixel Offset X of the Camera offsetX = 0
```

2.3.29 Get Pixel Offset Y

[Interface]

var offsetY = HCamera.getOffsetY(n)

[Description]

Retrieve the pixel offset in the Y direction of the Camera.

[Parameters]

(int)n: The index of the device.

[Return Value]

(int) Returns the current pixel offset in the Y direction of the Camera if successful; returns -1 if failed.

[Usage Example]

```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
var offsetY = HCamera.getOffsetY(0)
print("Pixel Offset Y of the Camera offsetY   = ", offsetY )
```

[Execution Effect]

```
Connect to Camera flag = true
Pixel Offset Y of the Camera offsetY = 0
```

2.3.30 Set Pixel Offset in X Direction

[Interface]

var flag = HCamera.setOffsetX(n, value)

[Description]

Set the pixel offset in the X direction of the Camera. Due to hardware limitations, some offset values cannot be set and will be automatically adjusted to the nearest value (rounded down). For example, setting the value to 9 may automatically adjust it to 8. It is recommended to check the current value after setting it, as shown in the example.

[Parameters]

(int)n: The index of the device.

(int) value: The pixel offset value in the X direction of the Camera.

[Return Value]

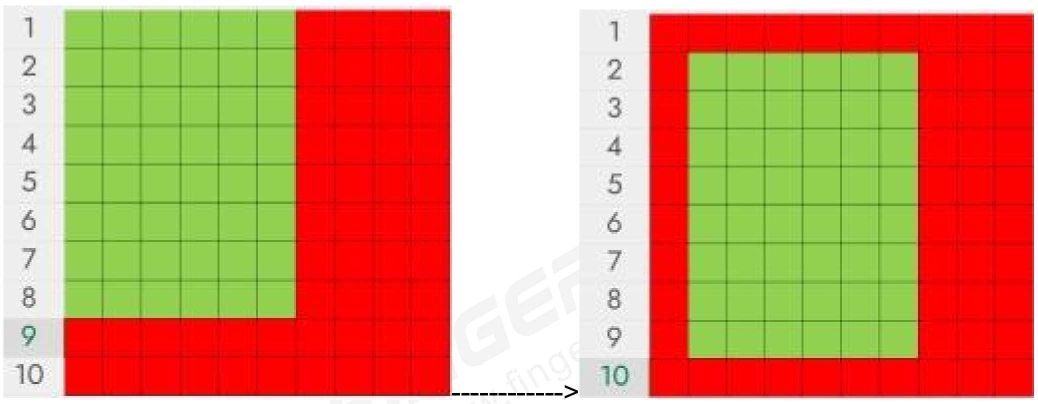
(bool) Returns true if successful; returns false if failed.

[Usage Example]

```
var offsetX = 30

flag = HCamera.setOffsetX(0, offsetX)

var currentValue = HCamera.getOffsetX(0)
```

[Execution Effect]

Assuming the Camera's pixel resolution is 10x10, the maximum pixel size is 10x10. If the pixel width and height are set to 6x8 using the corresponding interface, you can set offsets. For example, setting both X and Y offsets to 1 will result in a shifted effective area. The sum of X offset and pixel width should not exceed the maximum pixel width.



2.3.31 Set Pixel Offset in Y Direction

[Interface]

var flag = HCamera.setOffsetY(n, value)

[Description]

Set the pixel offset in the Y direction of the Camera. Due to hardware limitations, some offset values cannot be set and will be automatically adjusted to the nearest value (rounded down). For example, setting the value to 9 may automatically adjust it to 8. It is recommended to check the current value after setting it, as shown in the example.

[Parameters]

(int) n: The index of the device.

(int) value: The pixel offset value in the Y direction of the Camera.

[Return Value]

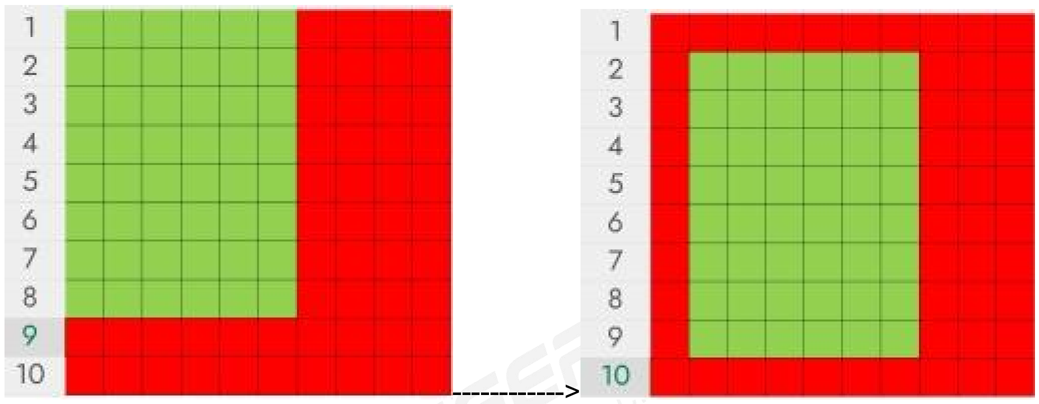
(bool) Returns true if successful; returns false if failed.

[Usage Example]

```
var offsetY = 30
flag = HCamera.setOffsetY(0, offsetY)
var currentValue = HCamera.getOffsetY(0)
```

[Execution Effect]

Assuming the Camera's pixel resolution is 10x10, the maximum pixel size is 10x10. If the pixel width and height are set to 6x8 using the corresponding interface, you can set offsets. For example, setting both X and Y offsets to 1 will result in a shifted effective area. The sum of Y offset and pixel height should not exceed the maximum pixel height.



2.3.32 Save Camera Parameters

[Interface]

var flag = HCamera.saveParameter(n)

[Description]

Save the Camera parameters. The parameters will be automatically read after the Camera is powered off and on again.

[Parameters]

(int)n: The index of the device.

[Return Value]

(bool) Returns true if successful; returns false if failed.

[Usage Example]

```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
var flag2 = HCamera.setExposureTime(0, 5000)
print("Modify Exposure Time flag2 = ", flag2)
var flag3 = HCamera.saveParameter(n)
print("Save Parameter flag3 = ", flag3 )
```

[Execution Effect]

```
Connect to Camera flag = true
Modify Exposure Time flag2 = true
Save Parameter flag3 = true
```

2.3.33 Clear Image Memory

[Interface]

var flag = HCamera.clearImage(image)

[Description]

Manually clear the image memory. For all interfaces of the Camera that return image types, you need to manually clear the image memory after use.

[Parameters]

(array) image: Image variable, typically obtained from the getImage() interface.

[Return Value]

(bool) Returns true if successful; returns false if failed.

[Usage Example]

```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
flag = HCamera.startGrabbing(0)
print("Start Grabbing flag = ", flag )
var image = HCamera.getImage (0)
var flag2 = HCamera.saveImage(image , "/home/image1.bmp")
print("Save Image flag2 = ", flag2)
var flag3 = HCamera.clearImage(image)
print("Clear Image flag3 = ", flag3)
```

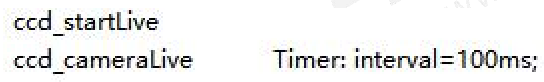
[Execution Effect]

```
Connect to Camera flag = true
Start Grabbing flag = true
Save Image flag2 = true
Clear Image flag3 = true
```

2.3.34 Real-Time Display (Video Stream)

[Description]

To implement real-time display, you need to define two macros (custom naming), as shown in the figure below. The ccd_startLive macro acts as a switch when bound to the control. Define a global variable liveFlag with a default value of false (default off); true to turn on, and false to turn off. The ccd_cameraLive macro sets up a timer to periodically capture and display images, achieving real-time display.



ccd_startLive
ccd_cameraLive Timer: interval=100ms;

[Usage Example]

(1) Implementation of ccd_startLive:

```
var flag = HCamera.connect(0)

print("Connect to Camera flag = ", flag)

If (!liveFlag )

{

HCamera.startGrabbing(0)

        liveFlag = true                        //liveFlag should be defined as a global variable

} else

{

        liveFlag = false

        HCamera.stopGrabbing(0)

}
```

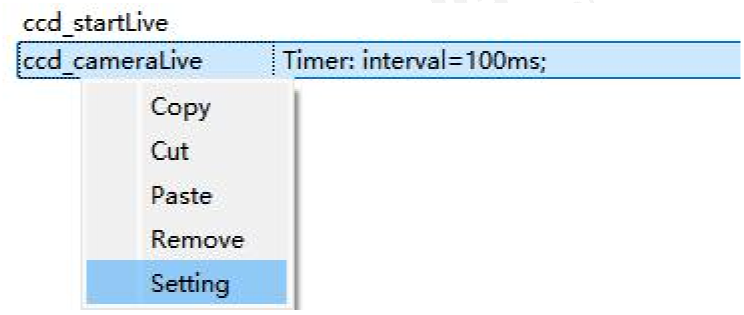
(2) Implementation of ccd_cameraLive:

```
//Form_3.hVisionView is the handle for the currently displayed interface plugin

if(liveFlag )

{

        var image = HCamera.getImage(0)

        Form_3.hVisionView.showImagheByData(image)
```

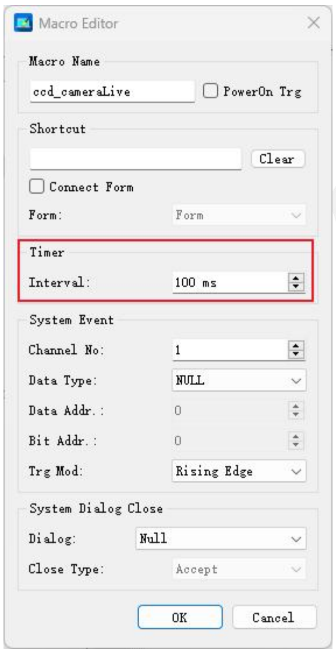
```
    HVisionModule.clearImage(image)
}
```

2.3.35 Setting Timer Method

1. Right-click on the macro that needs the timer set, and click "Settings."


ccd_startLive
ccd_cameraLive Timer: interval=100ms;

2. In the settings interface, locate the Timer section and set the period.


Macro Editor

Macro Name: ccd_cameraLive ☐ PowerOn Trg

Shortcut:
☐ Connect Form
Form: Form

Timer
Interval: 100 ms

System Event
Channel No: 1
Data Type: NULL
Data Addr.: 0
Bit Addr.: 0
Trg Mod: Rising Edge

System Dialog Close
Dialog: Null
Close Type: Accept

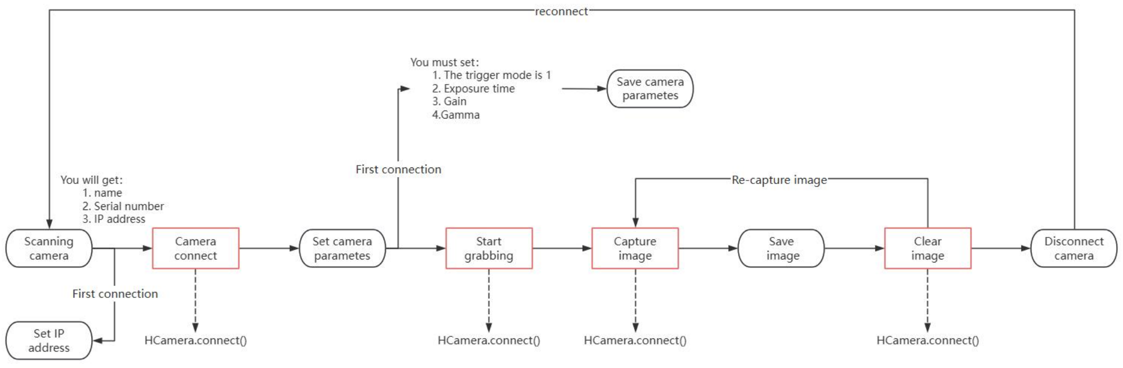
Note: The smaller the timer period, the higher the real-time display frame rate, but this also increases the CPU load, which may limit the exposure time.

Timer Period (ms)	Suggested Exposure Time (us)
100	1000~50000
150	1000~50000

200	1000~100000
250	1000~120000
300	1000~140000

2.4 Camera Module Notes

Please understand the following interface usage process. When capturing images, the steps marked in red are mandatory, while those in black are optional.



3. Image Processing Module Overview

3.1 Image Processing Module Features Overview (Ctrl+Click to Jump)

3.2 Image Processing Module Interface Details, Usage Examples, and Effects

3.2.1 Read Image

[Interface]

var image = HVisionModule.readImage(path, type)

[Description]

Reads an image from the local file system.

[Parameters]

(string)path: The path to the image file, e.g., path = "/home/hust/root/usr/image.bmp".

(int)type: Read type. type=0 for grayscale, type=1 for color.

[Return Value]

(array): Returns the image variable read from the path if successful; returns null if failed.

【Usage Example】

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var flag = HVisionModule.clearImage(image)
print("Clear image flag = ", flag)
```

[Execution Effect]

If the picture path is incorrect:

```
Clear image flag = false
```

else:

```
Clear image flag = true
```

Note: The read image interface returns an image variable in memory. Therefore, after use, remember to call the HVisionModule.clearImage(image) interface to clear it. Failing to do so might cause memory leaks due to continuous accumulation.

3.2.2 Convert Camera Image Variable to Vision Processing Module Image Variable

[Interface]

var visionImage = HVisionModule.cameraImageToVisionImage(cameraImage)

[Description]

The vision processing module cannot directly use the Camera module's image variable; this interface is used for conversion.

[Parameters]

(var)cameraImage: The image variable captured by the Camera module, typically returned by HCamera.getImage().

[Return Value]

(array): Returns an image variable that can be processed by the HVisionModule if successful; returns null if failed.

[Usage Example]

```
var flag = HCamera.connect(0)
print("Connect to Camera flag = ", flag)
var flag2 = HCamera.startGrabbing(0)
print("Start grabbing flag2 = ", flag2)
var src = HCamera.getImage(0)
var image = HVisionModule.cameraImageToVisionImage(src)
flag = HCamera.clearImage(src )
```

```
print("Clear image flag = ", flag)
flag2 = HVisionModule.clearImage(image)
print("Clear image flag2 = ", flag2)
```

[Execution Effect]

```
Connect to Camera flag = true
Start grabbing flag2 = true
Clear Image flag = true
Clear Image flag2 = true
```

Note: If the image variable is returned by an HCamera module interface, the system will automatically handle the cleanup. If the image variable is returned by an HVisionModule interface, you need to use the HVisionModule.clearImage() interface for cleanup.

3.2.3 Color Conversion

[Interface]

var image = HVisionModule.cvtColor(imageInput, type)

[Description]

Convert the image type.

[Parameters]

(array)image: Input image variable

(int)type: Conversion type. type = 0 represents color to grayscale, type = 1 represents grayscale to color

[Return Value]

(array) Returns the converted image if successful; returns null if failed. If the input image type is incorrect, the original image is returned.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)    //Read as grayscale image
var imageColor = HVisionModule.cvtColor(image , 1)            //Convert to color image
var flag = HVisionModule.clearImage(image)
```

```
print("Clear grayscale image flag = ", flag)
var flag2 = HVisionModule.clearImage(imageColor)
print("Clear color image flag2 = ", flag2)
```

[Execution Effect]

```
Clear grayscale Image flag = true
Clear color Image flag2 = true
```

3.2.4 Get Maximum Contours

[Interface]

var contour = HVisionModule.getMaxContours(image, thr, type)

[Description]

Get the largest contour points set from the image.

[Parameters]

(array)image: Input image variable

(int)thr: Threshold for contour segmentation, range is 0-255

(int)type: Contour extraction type. type = 0 represents white background with black objects, type = 1 represents black background with white objects

[Return Value]

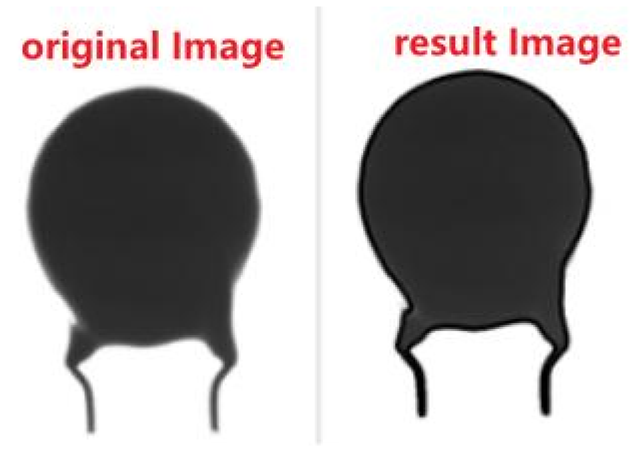
(array) Returns the largest contour found in the image if successful; returns null if failed.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var contour = HVisionModule.getMaxContours(image, 100, 0)
var flag = HVisionModule.drawContours (image, contour, -1)
print("Draw flag = ",flag)
flag = HVisionModule.saveImage(image, "/home/contour.bmp")
print("Save flag = ",flag)
var flag2 = HVisionModule.clearImage(image)
```

[Execution Effect]

```
Draw flag = true
Save flag2 = true
```



3.2.5 Get Bounding Rectangle of Contour

[Interface]

var list = HVisionModule.getBoundingRect(contour)

[Description]

Get the bounding rectangle of a contour point set. Returns the rectangle's center coordinates (centerX, centerY), width (w), height (h), and angle (0).

[Parameters]

(array)contour: Input contour point set

[Return Value]

(array) Returns a list containing [centerX, centerY, w, h, 0] if successful; returns null if failed.

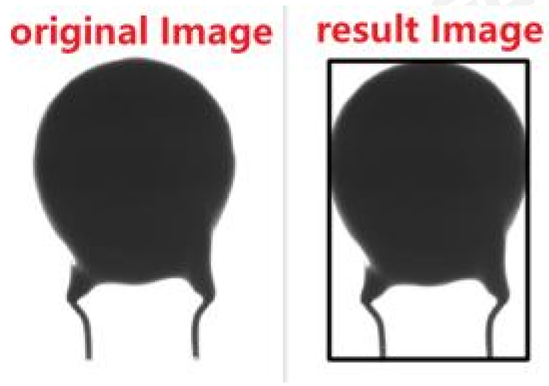
[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var contour = HVisionModule.getMaxContours(image, 100, 0)
var list = HVisionModule.getBoundingRect(contour)
print(list)
var flag = HVisionModule.drawRectangle2(image, list)
```

```
flag = HVisionModule.saveImage(image, "/home/contour.bmp")
print("Save flag = ",flag)
flag = HVisionModule.clearImage(image)
```

[Execution Effect]

```
list：474,503,297,444,0
Save flag = true
```



3.2.6 Get Minimum Bounding Rectangle of Contour (Smallest Area)

[Interface]

var list = HVisionModule.getMinAreaRect(contour)

[Description]

Get the position of the contour's minimum bounding rectangle (with the smallest area). The output includes center coordinates (centerX, centerY), rectangle length (w), width (h), and rotation angle (angle).

[Parameters]

(array)contour: Input contour point set

[Return Value]

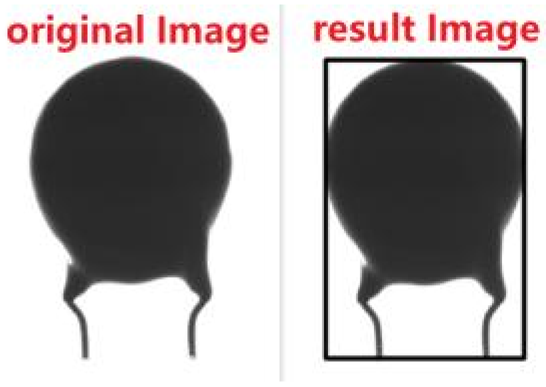
(array) Returns a list [centerX, centerY, w, h, angle] if successful; returns null if failed.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)

var contour = HVisionModule.getMaxContours(image, 100, 0)

var list = HVisionModule.getMinAreaRect (contour)

print(list)

var flag = HVisionModule.drawRectangle2(image, list)

flag = HVisionModule.saveImage(image, "/home/contour.bmp")

print("Save flag = ",flag)

flag = HVisionModule.clearImage(image)
```

[Execution Effect]

```
list：474,503,297,444,180
Save flag = true
```



3.2.7 Draw Contours

[Interface]

var flag = HVisionModule.drawContours(image, contour, num)

[Description]

Draw the contour points collection on the image.

[Parameters]

(array)image: The input image on which to draw.

(array)contour: The contour points collection to be drawn.

(int)num: The index of the contour to draw. Setting to -1 draws all contours.

[Return Value]

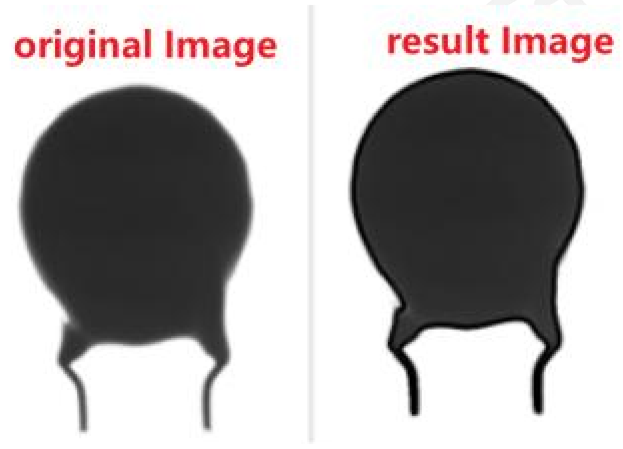
(bool) Returns true on success, false on failure.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var contour = HVisionModule.getMaxContours(image, 100, 0)
var flag = HVisionModule.drawContours (image, contour, -1)
print("Draw flag = ",flag)
flag = HVisionModule.saveImage(image, "/home/contour.bmp")
print("Save flag = ",flag)
var flag2 = HVisionModule.clearImage(image)
```

[Execution Effect]

```
Draw flag = true
Save flag = true
```



3.2.8 Draw Rectangle

[Interface]

var flag = HVisionModule.drawRectangle(image, pointX, pointY, w, h)

[Description]

Draw a rectangle on the image.

[Parameters]

(array)image: The input image on which to draw.

(int)pointX: The X coordinate of the starting point.

(int)pointY: The Y coordinate of the starting point.

(int)w: The width of the rectangle.

(int)h: The height of the rectangle.

[Return Value]

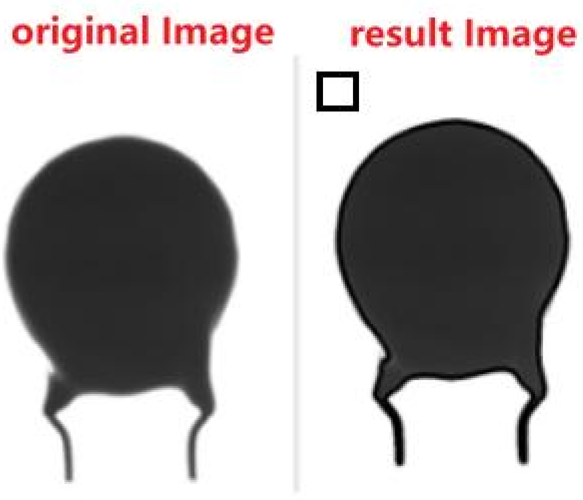
(bool) Returns true on success, false on failure.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)

var flag = HVisionModule.drawRectangle (image, 50, 50, 100, 100)

print("Draw flag = ",flag)

flag = HVisionModule.saveImage(image, "/home/rectImage.bmp")

print("Save flag = ",flag)

var flag2 = HVisionModule.clearImage(image)
```

[Execution Effect]

```
Draw flag = true
Save flag = true
```



3.2.9 Draw Rectangle2

[Interface]

var flag = HVisionModule.drawRectangle2(image, rectList)

[Description]

Draw rectangles on the image

[Parameters]

(array)image: The input image on which to draw.

(array)rectList: List of rectangles in the format [centerX, centerY, w, h, angle]. This format supports both axis-aligned and rotated rectangles.

[Return Value]

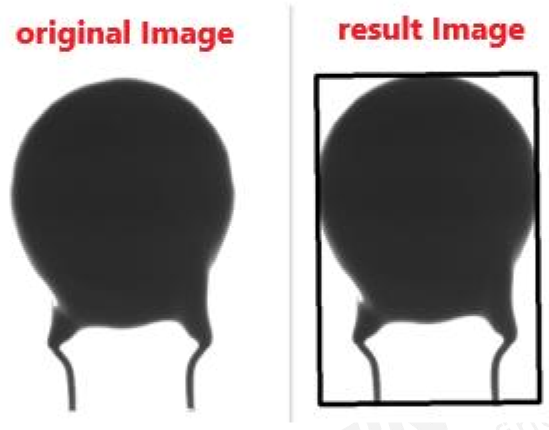
(bool) Returns true on success, false on failure.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var contour = HVisionModule.getMaxContours(image, 100, 0)
var list = HVisionModule.getMinAreaRect (contour)
var flag = HVisionModule.drawRectangle2(image, list)
flag = HVisionModule.saveImage(image, "/home/minRect.bmp")
print("Save flag = ",flag)
var flag2 = HVisionModule.clearImage(image)
```

[Execution Effect]

```
Save flag = true
```



3.2.10 Draw Line Segment

[Interface]

var flag = HVisionModule.drawLine(image, x1, y1, x2, y2, lineType = 0)

[Description]

Draw a line segment on the image

[Parameters]

(array)image: Input image to draw on

(int)x1: X coordinate of the line's starting point

(int)y1: Y coordinate of the line's starting point

(int)x2: X coordinate of the line's ending point

(int)y2: Y coordinate of the line's ending point

(int)lineType: Type of the line, default is solid line (0)

[Return Value]

(bool) Returns true if successful, false if failed.

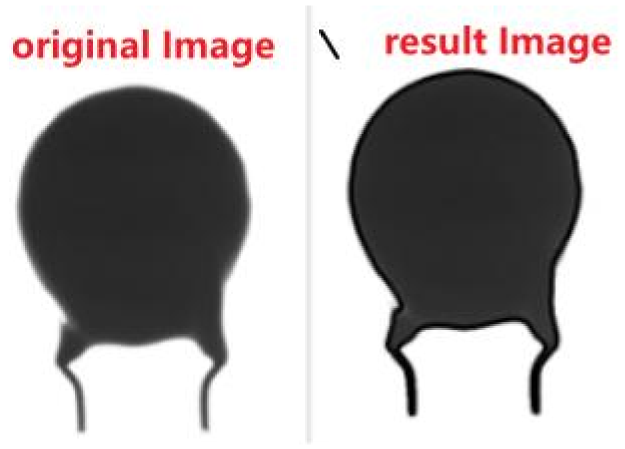
[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)

var flag = HVisionModule.drawLine (image, 50, 50, 100, 100)

flag = HVisionModule.saveImage(image, "/home/line.bmp")

print("Save flag = ",flag)
```

flag = HVisionModule.clearImage(image)

[Execution Effect]

Save flag = true



3.2.11 Draw Start Point

[Interface]

var flag = HVisionModule.moveTo(image, pointX, pointY)

[Description]

Draw a path on the image by setting the starting point. Use in combination with the lineTo() interface to draw a path.

[Parameters]

(array)image: Input image to draw on

(int)pointX: X coordinate of the starting point

(int)pointY: Y coordinate of the starting point

[Return Value]

(bool) Returns true if successful, false if failed.

[Usage Example]

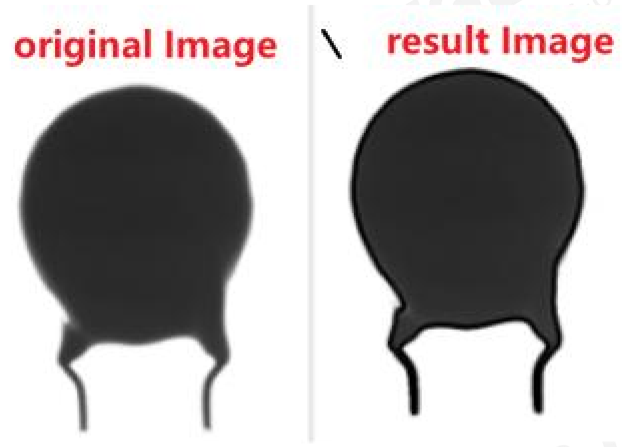
var image = HVisionModule.readImage("/home/test.bmp", 0)

var flag = HVisionModule.moveTo(image, 50, 50)

```
flag = HVisionModule.lineTo(image, 100, 100)

flag = HVisionModule.saveImage(image, "/home/moToLineTo.bmp")

print("Save flag = ",flag)

flag = HVisionModule.clearImage(image)
```

[Execution Effect]

```
Save flag = true
```



3.2.12 Draw Endpoint

[Interface]

var flag = HVisionModule.lineTo(image, pointX, pointY)

[Description]

Draws a line on the image, setting the endpoint at the specified coordinates. The starting point is set using the moveTo() interface or the previous lineTo() position.

[Parameters]

(array) image: The input image on which to draw.

(int) pointX: The X coordinate of the endpoint.

(int) pointY: The Y coordinate of the endpoint.

[Return Value]

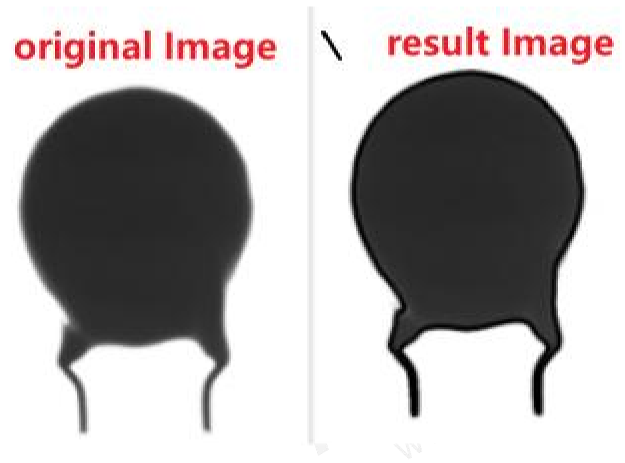
(bool) Returns true if successful, false otherwise.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)

var flag = HVisionModule.moveTo(image, 50, 50)

flag = HVisionModule.lineTo(image, 100, 100)

flag = HVisionModule.saveImage(image, "/home/moToLineTo.bmp")

print("Save flag = ",flag)

flag = HVisionModule.clearImage(image)
```

[Execution Effect]

```
Save flag = true
```



3.2.13 Draw Circle

[Interface]

var flag = HVisionModule.drawCircle(image, circleX, circleY, radius)

[Description]

Draws a circle on the image.

[Parameters]

(array) image: The input image on which to draw.

(int) circleX: The X coordinate of the circle's center.

(int) circleY: The Y coordinate of the circle's center.

(int) radius: The radius of the circle.

[Return Value]

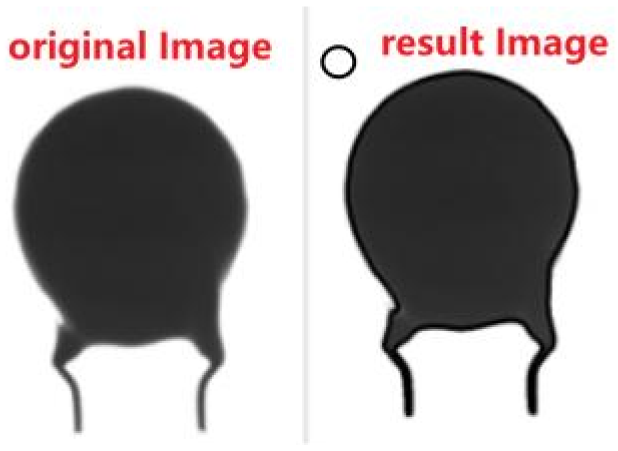
(bool) Returns true if successful, false otherwise.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var flag = HVisionModule.drawCircle(image, 200, 200, 100)
flag = HVisionModule.saveImage(image, "/home/cir.bmp")
print("Save flag = ",flag)
flag = HVisionModule.clearImage(image)
```

[Execution Effect]

```
Save flag = true
```



3.2.14 Draw Text

[Interface]

var flag = HVisionModule.drawText(image, text, pointX, pointY)

[Description]

Draws text on the image.

[Parameters]

(array) image: The input image on which to draw.

(string) text: The text to be drawn.

(int) pointX: The X coordinate of the text position.

(int) pointY: The Y coordinate of the text position.

[Return Value]

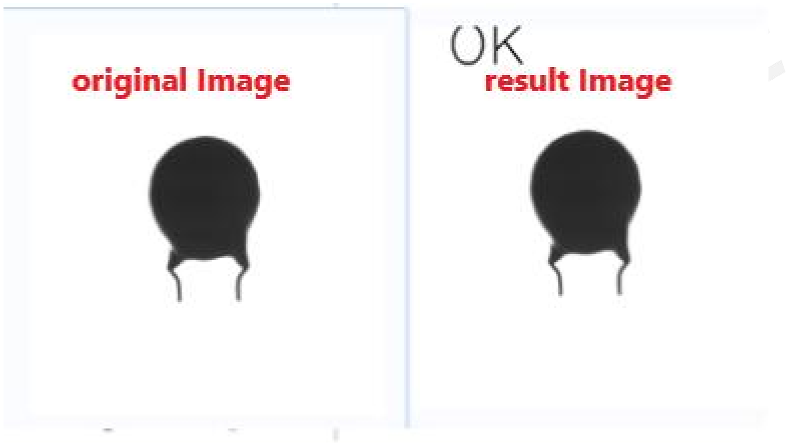
(bool) Returns true if successful, false otherwise.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var flag = HVisionModule.drawText(image, "OK", 100, 100)
flag = HVisionModule.saveImage(image, "/home/text.bmp")
print("Save flag = ",flag)
flag = HVisionModule.clearImage(image)
```

[Execution Effect]

```
Save flag = true
```



3.2.15 Crop Image

[Interface]

var dst = HVisionModule.cutImage(imageInput, x, y, w, h)

[Description]

Crops the image.

[Parameters]

(array) imageInput: The input image.

(int) x: The X coordinate of the cropping start point.

(int) y: The Y coordinate of the cropping start point.

(int) w: The width of the cropping area.

(int) h: The height of the cropping area.

[Return Value]

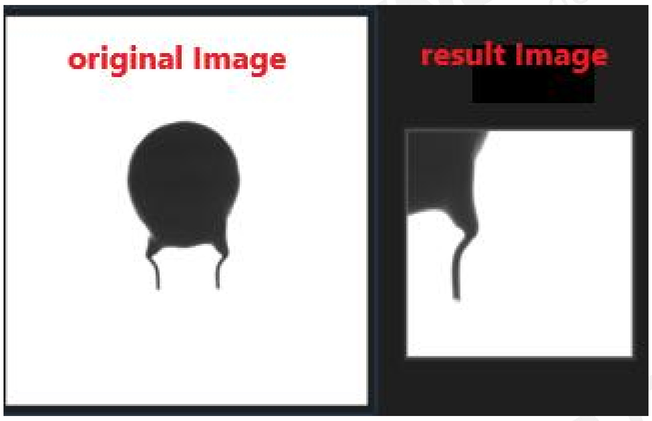
(array) Returns the cropped image if successful; returns null if failed.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var dst = HVisionModule.cutImage(image, 50, 50, 100, 100)
var flag = HVisionModule.saveImage(dst, "/home/cut.bmp")
print("Save flag = ",flag)
flag = HVisionModule.clearImage(image)
flag = HVisionModule.clearImage(dst)
```

[Execution Effect]

```
Save flag = true
```



3.2.16 Convert Rectangle to Four Vertex Coordinates

[Interface]

var pointList = HVisionModule.rectToPoints(rectList)

[Description]

Converts a rectangle to four vertex coordinates, returning them in a QVariantList in a counterclockwise order starting from the top-left corner.

[Parameters]

(array) rectList: The list of rectangle positions [centerX, centerY, w, h, angle].

[Return Value]

(array) Returns the coordinates of the four vertices of the minimum bounding rectangle [x1, y1, x2, y2, x3, y3, x4, y4] if successful; returns null if failed.

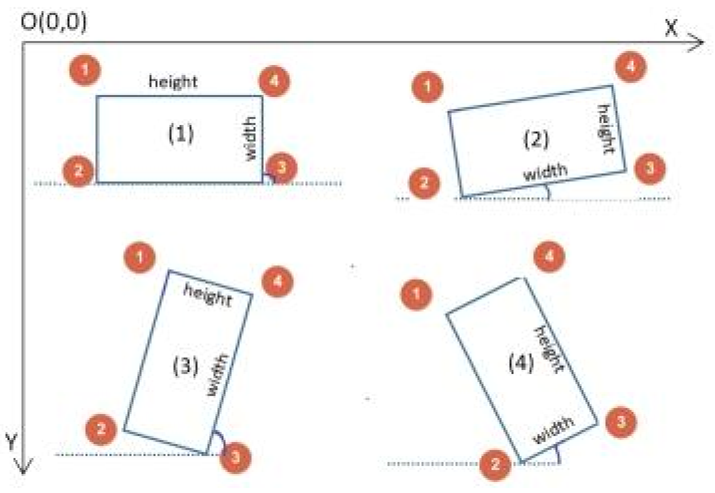
[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var contour = HVisionModule.getMaxContours(image, 100, 0)
var list = HVisionModule.getMinAreaRect (contour)
var pointList = HVisionModule.rectToPoints(list )
print("Vertex Coordinates:", pointList)
flag = HVisionModule.clearImage(image)
```

[Execution Effect]

```
Vertex Coordinates：625,721,620,279,324,282,329,724
```

The coordinates of the four vertices of the minimum bounding rectangle are in the following order, starting from the top-left corner (①) and moving counterclockwise.



3.2.17 Filter Contour Points Within a Range

[Interface]

var newContour = HVisionModule.getRangeContour(contour, type, min, max)

[Description]

Filters contour points. Given a set of contour points, you can use type to select whether to filter by x-direction or y-direction to get the contour points within the specified range.

[Parameters]

(array)contour: The original set of contour points

(int)type: Calculation type; 0 for x-direction filtering, 1 for y-direction filtering

(int)min: Minimum value for x-coordinate or y-coordinate of contour points

(int)max: Maximum value for x-coordinate or y-coordinate of contour points

[Return Value]

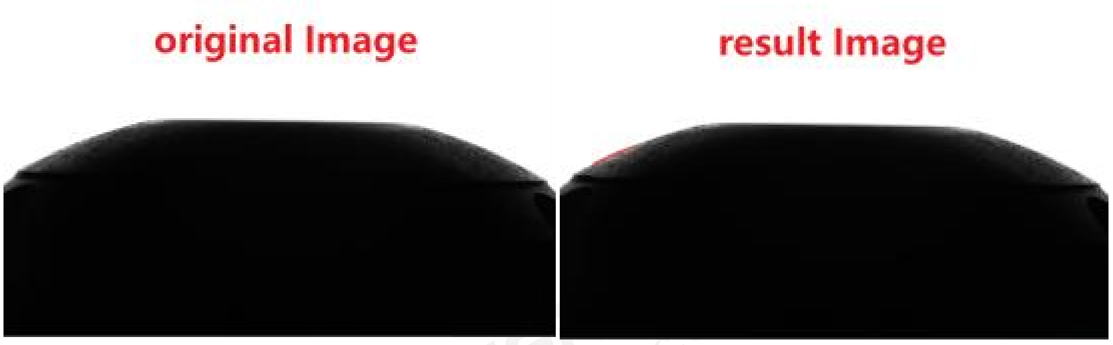
(array) Returns the set of contour points within the specified range if successful, otherwise returns null.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var contour = HVisionModule.getMaxContours(image, 100, 0)
var newContour = HVisionModule.getRangeContour(contour, 0, 100, 500)
var flag = HVisionModule.drawContours (image, newContour, -1)
print("Draw flag = ",flag)
flag = HVisionModule.saveImage(image, "/home/contour.bmp")
print("Save flag = ",flag)
flag = HVisionModule.clearImage(image)
```

[Execution Effect]

```
Draw flag = true
Save flag = true
```



3.2.18 Merge Contour Points

[Interface]

var newContour = HVisionModule.mergeContour(contour1, contour12)

[Description]

Merges two sets of contour points into one.

[Parameters]

(array)contour1: Contour points set 1

(array)contour2: Contour points set 2

[Return Value]

(array) Returns the merged contour points set if successful, otherwise returns null.

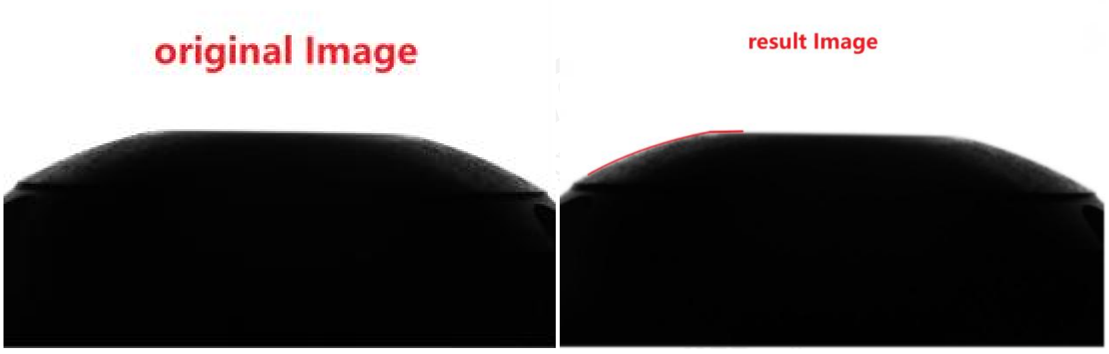
[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)

var contour = HVisionModule.getMaxContours(image, 100, 0)

var newContour = HVisionModule.getRangeContour(contour, 0, 100, 500)

var newContour2 = HVisionModule.getRangeContour(contour, 0, 1000, 1500)

var resultContour = HVisionModule.mergeContour(newContour, newContour2)

var flag = HVisionModule.drawContours (image, resultContour , -1)

print("Drawing flag = ",flag)

flag = HVisionModule.saveImage(image, "/home/contour.bmp")

print("Save flag = ",flag)
```

```
flag = HVisionModule.clearImage(image)
```

[Execution Effect]

```
Draw flag = true
Save flag = true
```



3.2.19 Get Arc

[Interface]

var arcList = HVisionModule.getArc(contour, num)

[Description]

Finds arcs within a set of contour points.

[Parameters]

(array)contour: Input contour points set used to calculate arcs.

(int)num: Input parameter specifying the number of iterations for the calculation.

[Return Value]

(array) Returns a list of circle centers and radii for the arcs [x, y, r] if successful, otherwise returns null.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var contour = HVisionModule.getMaxContours(image, 100, 0)
var arcList = HVisionModule.getArc(contour, 20)
print("Arc center and radius:", arcList )
flag = HVisionModule.clearImage(image)
```

[Execution Effect]

```
Arc center and radius：503,495,106
```

3.2.20 Set Drawing Color

[Interface]

HVisionModule.setPenColor(image, r, g, b)

[Description]

Sets the drawing color for the image. Note that the image must be in color for the color setting to be applied.

[Parameters]

(array)image: Input color image. Specifies the image for which the drawing color is set.

(int)r: Red channel value for the drawing color, should be between 0-255.

(int)g: Green channel value for the drawing color, should be between 0-255.

(int)b: Blue channel value for the drawing color, should be between 0-255.

[Return Value]

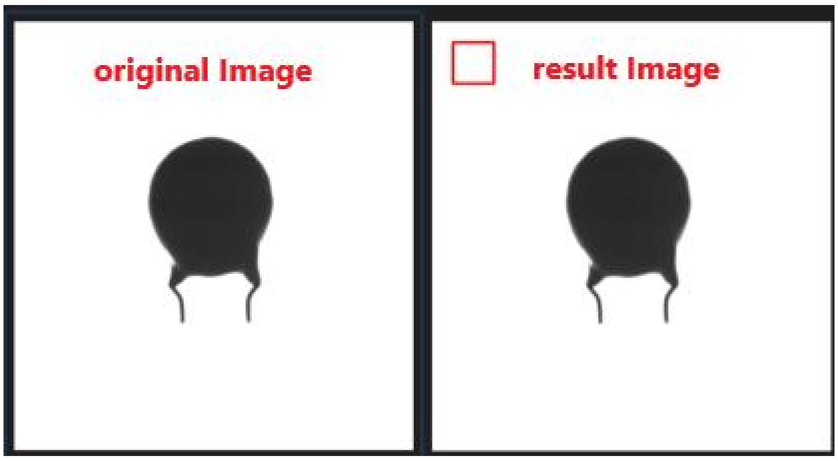
(bool) Returns true if successful, otherwise returns false.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var imageColor = HVisionModule.cvtColor(image , 1)
HVisionModule.setPenColor(imageColor , 255, 0, 0)
var flag = HVisionModule.drawRectangle (imageColor , 50, 50, 100, 100)
flag = HVisionModule.saveImage(imageColor , "/home/result.bmp")
print("Save flag = ",flag)
flag = HVisionModule.clearImage(image)
flag = HVisionModule.clearImage(imageColor)
```

[Execution Effect]

Save flag = true



3.2.21 Set Drawing Line Width

[Interface]

var flag = HVisionModule.setThickness(image, thickness)

[Description]

Sets the line width for all drawing operations.

[Parameters]

(array)image: Input image. Specifies the image for which the drawing line width is set.

(int)thickness: Line width for drawing, in pixels.

[Return Value]

(bool) Returns true if successful, otherwise returns false.

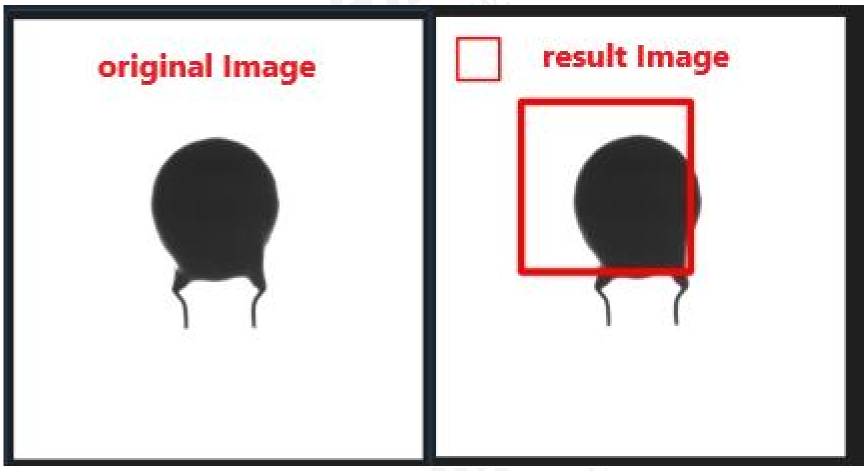
[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)

var imageColor = HVisionModule.cvtColor(image , 1)

HVisionModule.setPenColor(imageColor , 255, 0, 0)

var flag = HVisionModule.setThickness(imageColor , 5)

flag = HVisionModule.drawRectangle (imageColor , 50, 50, 100, 100)

var flag = HVisionModule.setThickness(imageColor , 15)
```

```
flag = HVisionModule.drawRectangle (imageColor , 200, 200, 400, 400)

flag = HVisionModule.saveImage(imageColor , "/home/result.bmp")

print("Save flag = ",flag)

flag = HVisionModule.clearImage(image)

flag = HVisionModule.clearImage(imageColor)
```

[Execution Effect]

```
Save flag = true
```



3.2.22 Threshold Segmentation

[Interface]

var thrImage = HVisionModule.thresholdProcessed(image, threshold, gray, type)

[Description]

Threshold segmentation process, converting a grayscale image into a binary image (with only 0 and 255).

[Parameters]

(array)image: Input grayscale image variable.

(int)threshold: Threshold value for segmentation, ranging from 0 to 255. For example, if threshold = 100, pixels with values less than 100 are set to 0, and those greater than or equal to 100 are set to the gray value.

(int)gray: Value ranging from 0 to 255, typically 255. When the grayscale value in the image is greater than the threshold, it is converted to the gray value.

(int)type: Segmentation type. After processing, type = 0 represents a white background with black objects, while type = 1 represents a black background with white objects.

[Return Value]

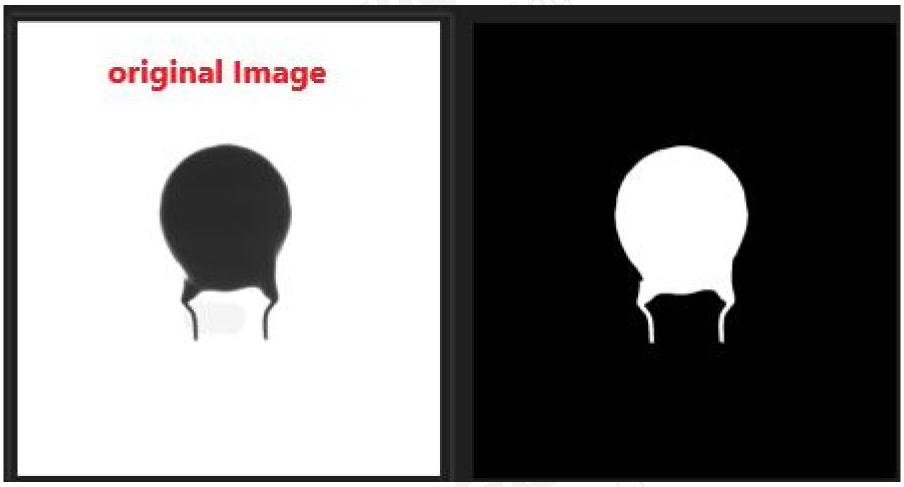
(array) Returns the processed binary image if successful, otherwise returns null.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var thrImage = HVisionModule.thresholdProcessed(image, 200, 255, 0)
var flag = HVisionModule.saveImage(thrImage , "/home/thr.bmp")
print("Save flag = ",flag)
flag = HVisionModule.clearImage(image)
flag = HVisionModule.clearImage(thrImage )
```

[Execution Effect]

```
Save flag = true
```



3.2.23 Opening Operation

[Interface]

var openImage = HVisionModule.openProcessed(image, xStep, yStep)

[Description]

Opening is a morphological operation in image processing, typically used to remove small objects or noise from an image. It first applies erosion, which reduces the size of objects in the image and may cause them to separate or disappear. Then, it applies dilation, which enlarges the objects in the image again. However, the small objects or noise removed during the erosion phase do not get enlarged again. Thus, by eroding first and then dilating, some small objects or noise can be eliminated.

[Parameters]

(array)image: Input image variable.

(int)xStep: Processing step size in the x direction. The larger the xStep, the stronger the processing effect in the x direction.

(int)yStep: Processing step size in the y direction. The larger the yStep, the stronger the processing effect in the y direction.

[Return Value]

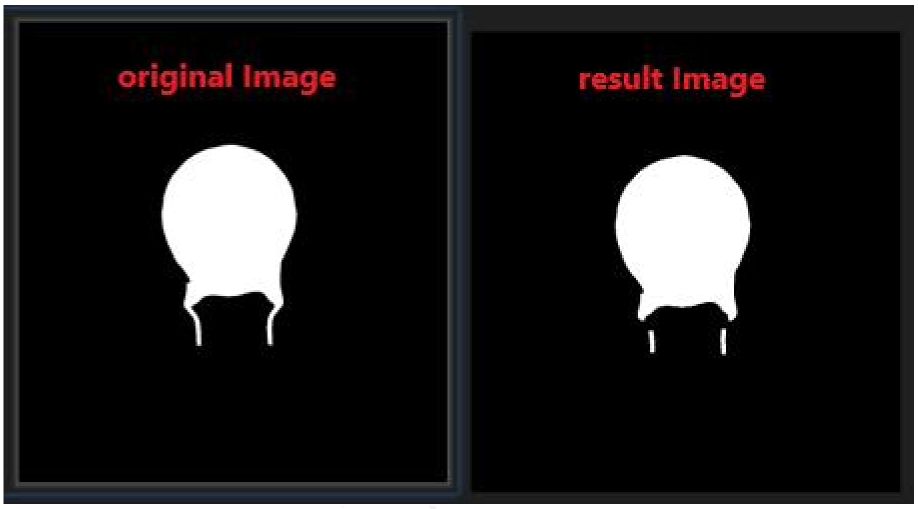
(array) Returns the processed image if successful, otherwise returns null.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var thrImage = HVisionModule.thresholdProcessed(image, 200, 255, 0)
var openImage = HVisionModule.openProcessed(thrImage, 10, 10)
var flag = HVisionModule.saveImage(openImage , "/home/open.bmp")
print("Save flag = ",flag)
flag = HVisionModule.clearImage(image)
flag = HVisionModule.clearImage(thrImage )
flag = HVisionModule.clearImage(openImage )
```

[Execution Effect]

```
Save flag = true
```



3.2.24 Image Subtraction

[Interface]

var subImage = HVisionModule.imageSubtraction(image1, image2)

[Description]

Subtracts one grayscale image from another.

[Parameters]

(array)image1: Input image variable 1.

(array)image2: Input image variable 2.

[Return Value]

(array) Returns the image after subtraction if successful, otherwise returns null.

[Usage Example]

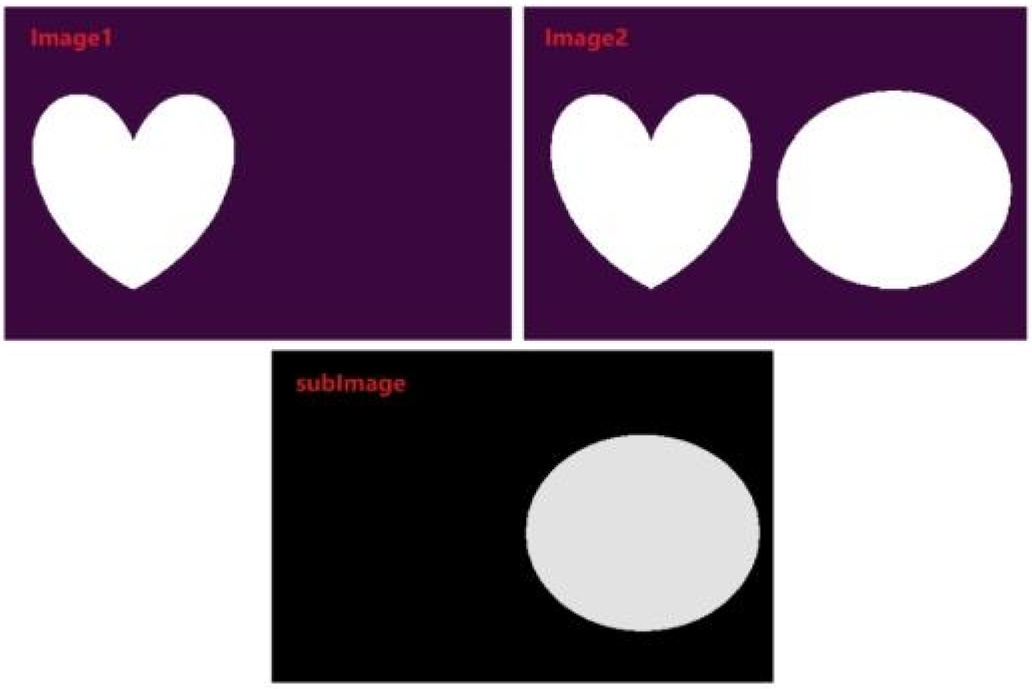
```
var image1 = HVisionModule.readImage("/home/Image1.bmp", 0)

var image2 = HVisionModule.readImage("/home/Image2.bmp", 0)

var subImage = HVisionModule.imageSubtraction(image1 , image2 )

var flag = HVisionModule.saveImage(subImage , "/home/sub.bmp")

print("Save flag = ",flag)

flag = HVisionModule.clearImage(Image2)
```

flag = HVisionModule.clearImage(Image1)

flag = HVisionModule.clearImage(subImage)

[Execution Effect]

```
Save flag = true
```



3.2.25 Get All Objects' Bounding Rectangles

[Interface]

var rectList = HVisionModule.getImageObjectRect(image)

[Description]

For a binary image, returns a list of bounding rectangles for all white regions in the image. Each rectangle is represented as [x, y, w, h].

[Parameters]

(array)image: Input image variable, preferably a binary image to ensure effective detection.

[Return Value]

(array) Returns a list of regions such as [[x1, y1, w1, h1], [x2, y2, w2, h2]] if successful; otherwise, returns null.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var thrImage = HVisionModule.thresholdProcessed(image, 100, 255, 0)
var rectList = HVisionModule.getImageObjectRect(thrImage)
flag = HVisionModule.clearImage(image)
flag = HVisionModule.clearImage(thrImage)
```

[Execution Effect]

To be supplemented

3.2.26 Get Average Hash Value of Image

[Interface]

var strHashList = HVisionModule.averageHashFun(image, listRect)

[Description]

Input the image and regions, use the average hash algorithm to compute the hash values of each region in the image, and return a list of strings for comparing the similarity of different regions in the image.

[Parameters]

(array)image: The input image variable

(array)listRect: The list of regions to be processed, typically obtained from the getImageObjectRect() function

[Return Value]

(array) Returns a list of hash values for each region in the image if successful; returns empty if failed.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var thrImage = HVisionModule.thresholdProcessed(image, 100, 255, 0)

var rectList = HVisionModule.getImageObjectRect(thrImage)

var strHashList = HVisionModule.averageHashFun(image, rectList )

flag = HVisionModule.clearImage(image)

flag = HVisionModule.clearImage(thrImage)
```

[Execution Effect]

```
hash value： 1111111111111011110101110110110110111000000000000000100000000000
00001000000000000......
```

3.2.27 Compare Image Hash Values

[Interface]

var HVisionModule.contrastHashFun(str1, str2)

[Description]

Compare two lists of hash values. The size of the lists must be equal. Typically, the comparison is done using hash lists computed from the same regions of two different images. The value ranges from 0 (minimum) to 400 * list size (maximum).

[Parameters]

(array)str1: Hash value list 1

(array)str2: Hash value list 2

[Return Value]

(int) Returns the comparison value if successful. A smaller value indicates greater similarity. Returns -1 if failed.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)

var image2 = HVisionModule.readImage("/home/test2.bmp", 0)

var thrImage = HVisionModule.thresholdProcessed(image, 100, 255, 0)

var rectList = HVisionModule.getImageObjectRect(thrImage)

var strHashList = HVisionModule.averageHashFun(image, rectList)
var strHashList2 = HVisionModule.averageHashFun(image2, rectList)
var value = HVisionModule.contrastHashFun(strHashList, strHashList2)
print("Contrast Value:", value)
flag = HVisionModule.clearImage(image)
flag = HVisionModule.clearImage(image2)
flag = HVisionModule.clearImage(thrImage)
```

[Execution Effect]

```
Contrast value：46
```

3.2.28 Save Image

[Interface]

var flag = HVisionModule.saveImage(image, path)

[Description]

Save the image to local storage.

[Parameters]

(array)image: The image variable to be saved.

(string)path: The path and filename where the image will be saved, e.g., path = "/home/usr/123.bmp"

[Return Value]

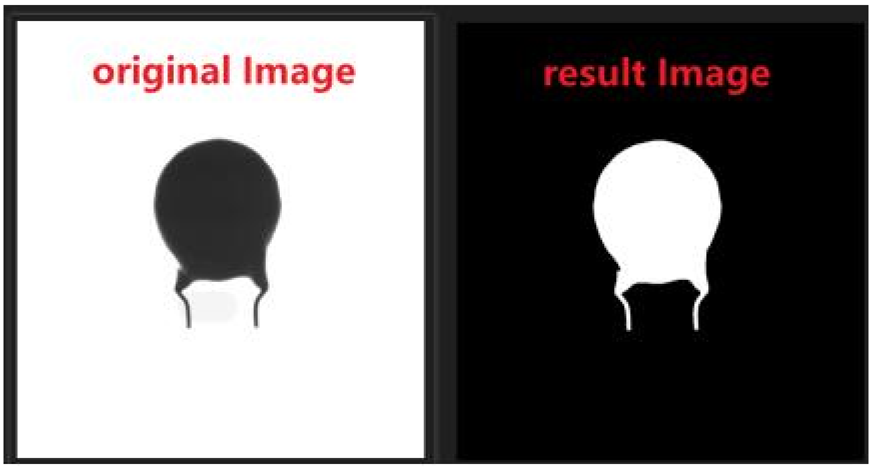
(bool) Returns true if successful, false if failed.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var thrImage = HVisionModule.thresholdProcessed(image, 200, 255, 0)
var flag = HVisionModule.saveImage(thrImage , "/home/thr.bmp")
print("Save flag = ",flag)
flag = HVisionModule.clearImage(image)
flag = HVisionModule.clearImage(thrImage )
```

[Execution Effect]

```
Save flag = true
```



3.2.29 Clear Image Memory

[Interface]

var flag = HVisionModule.clearImage(image)

[Description]

Manually clear image memory. For all interfaces that return image types, it is necessary to manually clear the image memory after use.

[Parameters]

(array)image: The image variable for which memory needs to be cleared.

[Return Value]

(bool) Returns true if successful, false if failed.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)    //Load as grayscale image
var imageColor = HVisionModule.cvtColor(image , 1)          //Convert to color image
var flag = HVisionModule.clearImage(image)
print("Clear grayscale image flag = ", flag)
var flag2 = HVisionModule.clearImage(imageColor)
```

```
print("Clear image flag2 = ", flag2)
```

[Execution Effect]

```
Clear grayscale image flag = true
Clear image flag2 = true
```

3.2.30 Set Drawing Text Size

[Interface]

var flag = HVisionModule.setTextSize(image, size)

[Description]

Set the text size for all drawing operations.

[Parameters]

(array)image: The input image where the text size for drawing will be set.

(int)size: The font size. The default value is 5. The unit is not specified; adjust this value through debugging.

[Return Value]

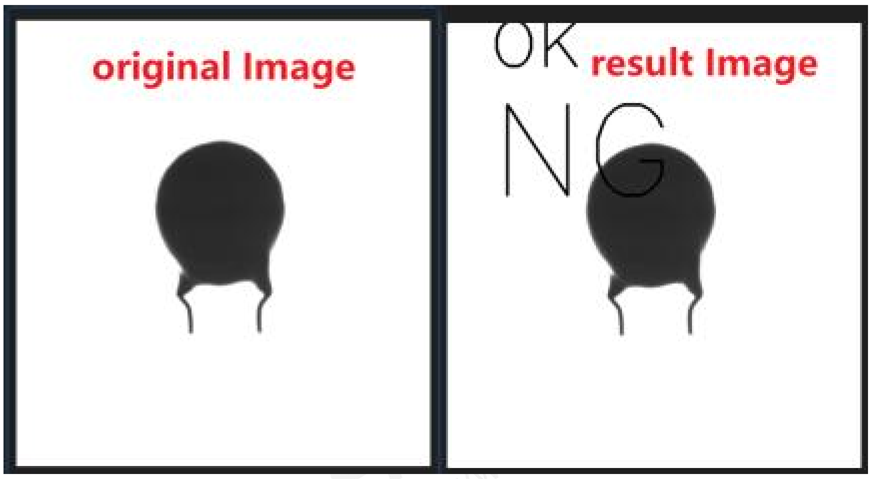
(bool) Returns true if successful, false if failed.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
flag = HVisionModule.setTextSize(image, 5)
var flag = HVisionModule.drawText(image, "OK", 100, 100)
flag = HVisionModule.setTextSize(image, 10)
var flag = HVisionModule.drawText(image, "NG", 100, 400)
flag = HVisionModule.saveImage(image, "/home/text.bmp")
print("Save flag = ",flag)
flag = HVisionModule.clearImage(image)
```

[Execution Effect]

```
Save flag = true
```



3.2.31 Edge Detection

[Interface]

var image = HVisionModule.edgeDetect(image, highThreshold, lowThreshold, coreSize)

[Description]

Performs edge detection. Input a grayscale image, adjust detection parameters, and return the edge-detected image.

[Parameters]

(array)image: The input image variable.

(int)highThreshold: High threshold. Values above this are considered edges, in the range of 0-255.

(int)lowThreshold: Low threshold. Values below this are not considered edges, in the range of 0-255. Typically about half of the highThreshold value.

(int)coreSize: Size of the edge detection kernel. Generally set to 3. Larger values require more pronounced edges.

[Return Value]

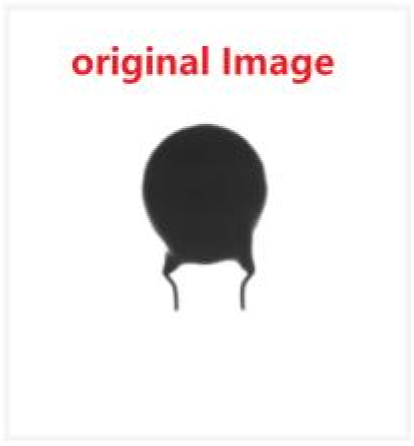
(array) Returns the edge-detected image if successful, or null if failed.

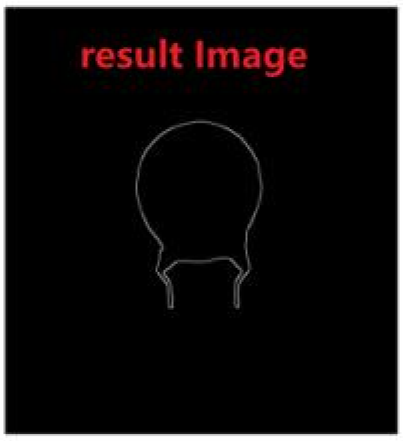
[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var edgeimage = HVisionModule.edgeDetect(image , 200, 150, 3)
flag = HVisionModule.saveImage(edgeimage , "/home/edge.bmp")
print("Save flag = ",flag)
var flag1 = HVisionModule.clearImage(edgeimage )
var flag2 = HVisionModule.clearImage(image)
```

[Execution Effect]

```
Save flag = true
```





3.2.32 Get Fitted Line

[Interface]

varlineList = HVisionModule.getFittedLine(image, thr, objectColor, polarityFlag, polarityDirection, edgeSize)

[Description]

Obtains the fitted lines in an image.

[Parameters]

(array)image: The input image variable.

(int)thr: Threshold for finding contour points in the image. Used for line fitting, range from -1 to 255, where -1 is for automatic threshold.

(bool)objectColor: Target color after thresholding. 0 for inverted black and white, 1 for unchanged black and white. Default is 0.

(bool)polarityFlag: Line polarity. 0 for white to black, 1 for black to white. Default is 0.

(int)polarityDirection: Line polarity direction. 0 for top-to-bottom, 1 for left-to-right. Default is 0 (top-to-bottom).

(int)edgeSize: Edge search size for line fitting. Only 3, 5, or 7 are allowed. Default is 5. Larger sizes are considered for high-resolution images or more precise edge detection.

[Return Value]

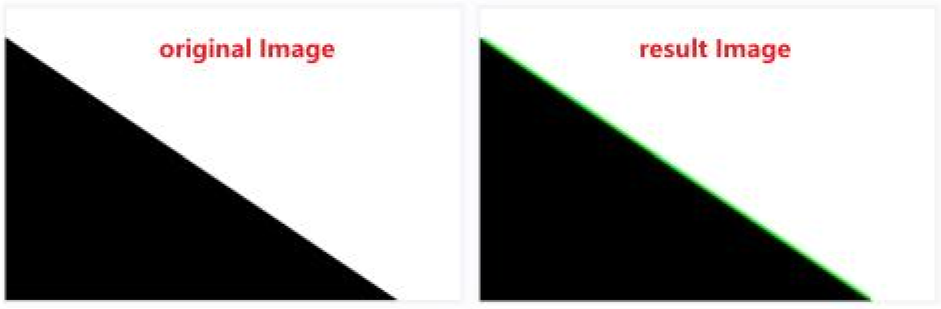
(array) Returns an array result[x1, y1, x2, y2, k, b], where x1, y1, x2, y2 are the start and end coordinates of the line (represented as a segment), and k, b are the coefficients in the line equation y = kx + b. For multiple lines, the result array includes data for each line, with 6 values per line.

[Usage Example]

```
var image = HVisionModule.readImage("/home/test.bmp", 0)
var lineList = HVisionModule.getFittedLine(image, 100)
print(lineList)
//Drawing Color Effect Imag
var colorimg = HVisionModule.cvtColor(image,1)
HVisionModule.setPenColor(colorimg,0,255,0)
var drawline =
HVisionModule.drawLine(colorimg,lineList[0],lineList[1],lineList[2],lineList[3])
var flag1 = HVisionModule.clearImage(colorimg )
var flag2 = HVisionModule.clearImage(image)
```

[Execution Effect]

```
[x1, y1, x2, y2, k, b] = [5, 51, 642, 477, 0.668056309, 48.5107421875]
```



3.2.33 Relative Coordinate Transformation for Fitted Lines

[Interface]

var axisList = HVisionModule.lineConvertByOrigin(lineList, offsetX, offsetY)

[Description]

Transforms the coordinates of fitted lines from a cropped image to the original image.

[Parameters]

(array)lineList: Array of fitted line coordinates relative to the cropped image.

(int)offsetX: X-direction offset, which is the X-coordinate of the top-left corner of the cropped image ROI.

(int)offsetY: Y-direction offset, which is the Y-coordinate of the top-left corner of the cropped image ROI.

[Return Value]

(array) Returns an array of fitted line coordinates relative to the original image. Returns an empty array if the operation fails.

[Usage Example]

var image = HVisionModule.readImage("/home/root/hust/05.bmp", 0);   // Read image

var reduceImg1 = HVisionModule.cutImage(image1, 1600, 520, 130, 120);   // Crop area for line fitting

var reduceImg2 = HVisionModule.cutImage(image1, 1390, 1240, 130, 120);

var lineList1 = HVisionModule.getFittedLine(reduceImg1, 150);   // Fit lines var lineList2 = HVisionModule.getFittedLine(reduceImg2, 150);

var originList1 = HVisionModule.lineConvertByOrigin(lineList1, 1600, 520); // Transform relative coordinates

var originList2 = HVisionModule.lineConvertByOrigin(lineList2,1390,1240)

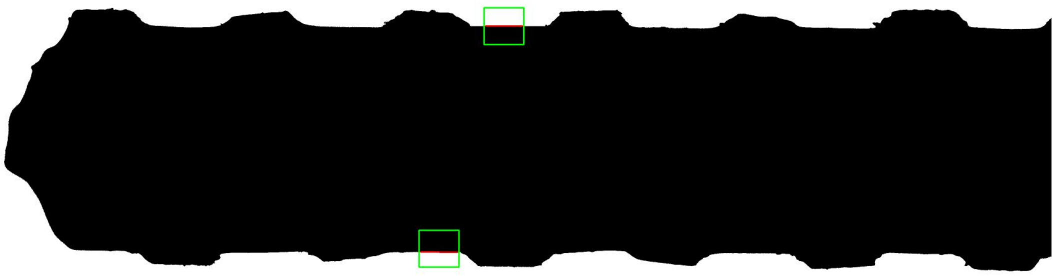
HVisionModule.clearImage(reduceImg1)

HVisionModule.clearImage(reduceImg2)

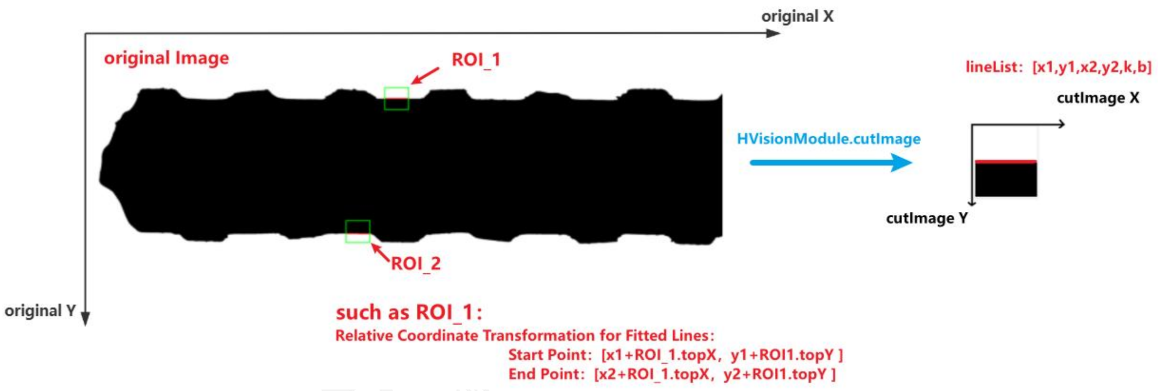
HVisionModule.clearImage(image)

[Execution Effect]

```
lineList1:[x1, y1, x2, y2, k, b]= [5, 59, 125, 59, 0, 59]
originList1:[x1, y1, x2, y2, k, b]=[1605, 579, 1725, 579, 0, 579]

lineList2:[x1, y1, x2, y2, k, b]=[5, 70, 125, 71, 0.00040645, 70.96531677]
originList2:[x1, y1, x2, y2, k, b]=[1395, 1310, 1515, 1311, 0 00040645, 1309 .90861282]
```



Relative Coordinate Transformation Explanation:


original X
original Image
ROI_1
ROI_2
original Y
HVisionModule.cutImage
lineList: [x1,y1,x2,y2,k,b]
cutImage X
cutImage Y
such as ROI_1:
Relative Coordinate Transformation for Fitted Lines:
Start Point: [x1+ROI_1.topX, y1+ROI1.topY]
End Point: [x2+ROI_1.topX, y2+ROI1.topY]

3.2.34 Get Line Distance

[Interface]

var distance = HVisionModule.getLineDistance(lineList1, lineList2, error)

[Description]

Calculates the distance between two parallel lines.

[Parameters]

(array) lineList1: Array for Line 1, format [x1, y1, x2, y2, k, b], where k and b are the slope and intercept of the line equation y = kx + b.

(array) lineList2: Array for Line 2, format [x1, y1, x2, y2, k, b].

(double) error: Allowed tolerance for the difference in slopes of the two lines. If (k1 - k2) is within [-error, error], the lines are considered parallel. Default value is 0.

[Return Value]

(double): Returns the distance between the two parallel lines if successful; returns -1 if failed.

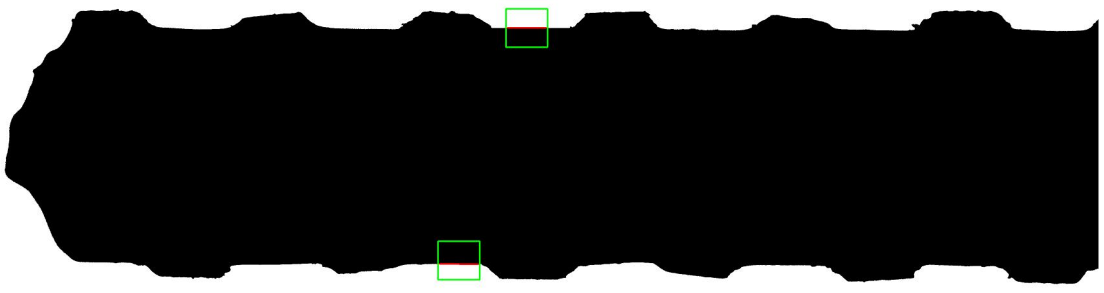
[Usage Example]

```
var image = HVisionModule.readImage("/home/root/hust/05.bmp", 0)      //读取图片
var image = HVisionModule.readImage("/home/root/hust/05.bmp", 0)    // Read image
var reduceImg1 = HVisionModule.cutImage(image, 1600, 520, 130, 120)    // Crop image
```

```
for line fitting
var reduceImg2 = HVisionModule.cutImage(image, 1390, 1240, 130, 120)
var lineList1 = HVisionModule.getFittedLine(reduceImg1, 150)    // Fit lines
var lineList2 = HVisionModule.getFittedLine(reduceImg2, 150)
var originList1 = HVisionModule.lineConvertByOrigin(lineList1, 1600, 520)       // Convert
coordinates to original image
var originList2 = HVisionModule.lineConvertByOrigin(lineList2, 1390, 1240)
var distance = HVisionModule.getLineDistance(originList1, originList2, 0.2); // Calculate
distance between lines
HVisionModule.clearImage(reduceImg1)
HVisionModule.clearImage(reduceImg2)
HVisionModule.clearImage(image)
```

[Execution Effect]

```
The distance between the two lines is:  730.9086128218623
```



3.2.35 Get Line Angle

[Interface]

var distance = HVisionModule.getLineAngle(slope1, slope2, error)

[Description]

Calculates the angle between two lines.

[Parameters]

(array) slope1: Slope of Line 1.

(array) slope2: Slope of Line 2.

(double) error: Allowed tolerance for the difference in slopes. If (slope1 - slope2) is within [-error, error], the lines are considered parallel. Default value is 0.

[Return Value]

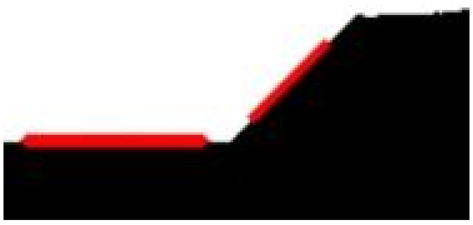
(double): Returns the angle between the two lines if successful; returns 0 if failed.

[Usage Example]

```
var image = HVisionModule.readImage("/home/root/hust/05.bmp", 0)      //Read image
var reduceImg1 = HVisionModule.cutImage(image1,1725,520,75,120)
var reduceImg2 = HVisionModule.cutImage(image1,1800,530,45,45)
var lineList1 = HVisionModule.getFittedLine(reduceImg1, 150)
var lineList2 = HVisionModule.getFittedLine(reduceImg2, 150)
HVisionModule.getLineAngle(lineList1[4],lineList2[4],0);
HVisionModule.clearImage(reduceImg1)
HVisionModule.clearImage(reduceImg2)
HVisionModule.clearImage(image1)
```

[Execution Effect]

```
The angle between the two lines is: 45°
```



3.2.36 Image Data Conversion

[Interface]

var imageData = HVisionModule.imageToData(image)

[Description]

Converts an image variable into data that can be directly displayed. Note that the data needs to be cleaned up.

[Parameters]

(array) image: Input image variable.

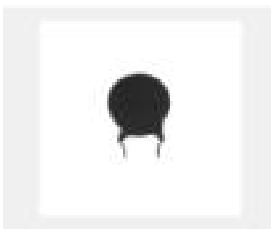
[Return Value]

(array): Returns a data variable that can be directly used for displaying with dynamic image plugins.

[Usage Example]

```
var image = HVisionModule.readImage("/home/root/hust/05.bmp", 0)    // Read image
var data = HVisionModule.imageToData(image)    // Convert image to data format
Form.dynamicImage.loadFromData(data)      // Load and display image with dynamic image plugin
HVisionModule.clearImage(image)
```

[Execution Effect]



3.2.37 Get White Area Size

[Interface]

var areaList= HVisionModule.getObjectArea(image)

[Description]

Calculates the area of white regions in an image. Note: The input image should be a threshold-segmented image.

[Parameters]

(array)image：Input image variable.

[Return Value]

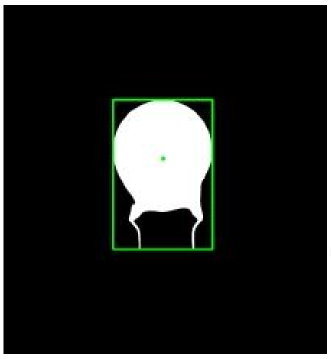
(array): Returns an array result[area, centerX, centerY], where area is the size of the white region, and centerX, centerY are the X and Y coordinates of the white region's center point.

[Usage Example]

```
var image = HVisionModule.readImage("/home/root/hust/05.bmp", 0)    // Read image
var thrImg = HVisionModule.thresholdProcessed(image, 150, 255, 1)    // Threshold segmentation
var areaList = HVisionModule.getObjectArea(thrImg)    // Calculate white area size
print("White area size: " + areaList[0])
var contours = HVisionModule.getMaxContours(thrImg,150,1)
var rectList = HVisionModule.getBoundingRect(contours)
```

[Execution Effect]

```
The white area is：81311
```



3.2.38 Calculate Tangent Points and Distances Between a Line Segment and Contours

[Interface]

var resultList= HVisionModule.lineToContours(x1, y1, x2, y2, contours);

[Description]

Calculates the slope of a line segment and determines the tangent points of this slope with the contours. Also computes the distance from these tangent points to the line segment (the furthest tangent point).

[Parameters]

(int) x1: X-coordinate of the line segment's start point.

(int) y1: Y-coordinate of the line segment's start point.

(int) x2: X-coordinate of the line segment's end point.

(int) y2: Y-coordinate of the line segment's end point.

(array) contours: The set of contour points.

[Return Value]

(array): Returns an array result[pointX, pointY, distance], where pointX and pointY are the coordinates of the tangent point, and distance is the distance from the tangent point to the line segment.

[Usage Example]

```
var image = HVisionModule.readImage("/home/root/hust/05.bmp", 0)    // Read image
var thrImg = HVisionModule.thresholdProcessed(image, 150, 255, 1)    // Threshold segmentation
var contours = HVisionModule.getMaxContours(thrImg, 100, 1)    // Get the maximum contours
// Perform opening operation to obtain target external rectangle or fitted line
var openimg = HVisionModule.openProcessed(thrImg, 210, 1)
var rectContours = HVisionModule.getMaxContours(openimg, 100, 1)    // Get the fitted
```

rectangle contours

// Obtain start and end points of the line segment from the external rectangle or fitted line

var list = HVisionModule.getMinAreaRect(rectContours)   // Fit external rectangle

var points = HVisionModule.rectToPoints(ContoursList)   // Convert rectangle to vertex coordinates

contours = HVisionModule.getRangeContour(contours, 1, 0, points[1])   // Filter contours by y-direction

// Calculate tangent points and distances based on the top side length of the tilted rectangle

var result1 = HVisionModule.lineToContours(points[0], points[1], points[6], points[7], contours)

// Calculate tangent points and distances based on the right side length of the tilted rectangle

var result2 = HVisionModule.lineToContours(points[4], points[5], points[6], points[7], contours)

print("Distance from tangent point1 to the top side of the tilted rectangle:", result1[2])

print("Distance from tangent point2 to the right side of the tilted rectangle:", result2[2])

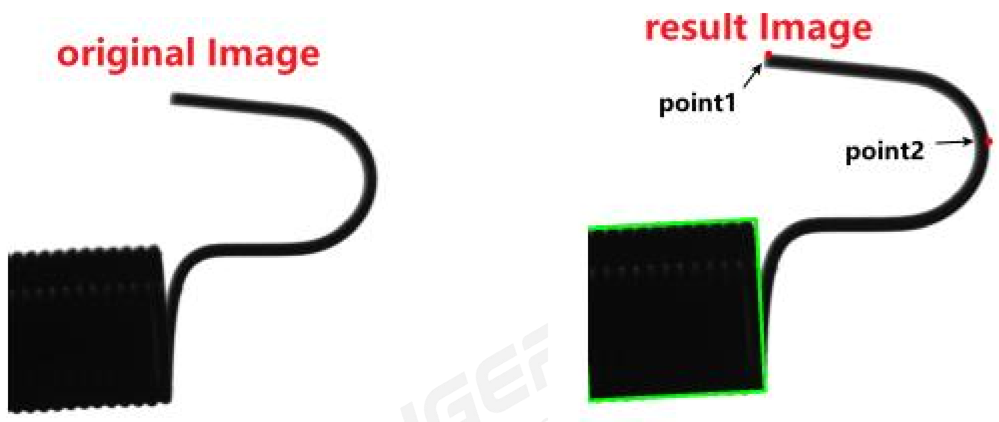
HVisionModule.clearImage(thrImg)

HVisionModule.clearImage(openimg)

HVisionModule.clearImage(image)

[Execution Effect]

```
Distance from tangent point1 to the top side of the tilted rectangle：288.48663798
Distance from tangent point2 to the right side of the tilted rectangle：415.69551668
```



3.2.39 Template Matching

[Interface]

var resultList= HVisionModule.getMatchTemplate(image,templImage,type,score)

[Description]

Performs template matching on the image and returns the matching results.

[Parameters]

(array) image: The image variable to be used for template matching.

(array) templImage: The standard template image variable.

(int) type: The template matching calculation type. Currently, only three types are allowed:

1: Standard squared difference matching.

3: Standard correlation matching, using multiplication between the template and the image.

5: Correlation coefficient matching, matching the relative values of the template's mean with the image's mean.

(float) score: The template matching score standard, ranging from 0 to 100. A higher score indicates that the matched area is more similar to the standard template.

[Return Value]

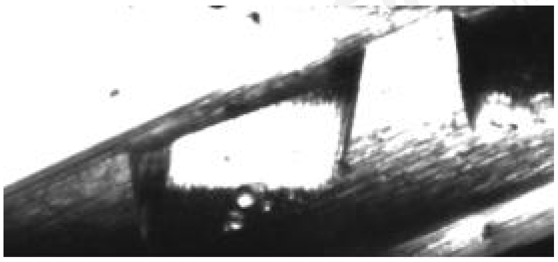
(array): Returns an array result[topX, topY, maxValue], where topX and topY are the coordinates of the top-left corner of the matching area, and maxValue is the highest matching score. If maxValue is less than score, the array [0, 0, 0] is returned.

[Usage Example]

```
var image = HVisionModule.readImage("/home/root/usr/image.bmp", 0)  //Read image
var rectinfo = Form.hVisionView.getItemInfo(-1)   // Get rectangle info after editing in the vision plugin
var xTop = rectinfo[1] - rectinfo[3] / 2
var yTop = rectinfo[2] - rectinfo[4] / 2
var w = rectinfo[3]
var h = rectinfo[4]
var reduceImg = HVisionModule.cutImage(image, xTop, yTop, w, h)   // Crop image for template use
// Alternatively, directly read the template image
// var tempImage = HVisionModule.readImage("/home/root/usr/image.bmp", 0)
var resultList = HVisionModule.getMatchTemplate(image, reduceImg, 3, 90)   // Perform template matching
var colorImg = HVisionModule.cvtColor(image, 1)
HVisionModule.setPenColor(colorImg, 0, 255, 0)
HVisionModule.drawRectangle(colorImg, xTop, yTop, w, h)   // Draw rectangle to display

HVisionModule.clearImage(colorImg)
HVisionModule.clearImage(reduceImg)
HVisionModule.clearImage(iamge)
```

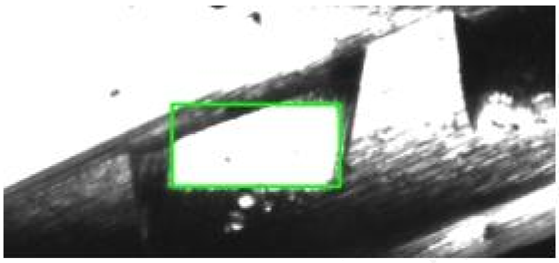
[Execution Effect]

Original Image:



Template:

Result Image:



3.2.40 Image Resizing

[Interface]

var resizeImg= HVisionModule.resizeImage(image,sizeX, sizeY)

[Description]

Rescale the image proportionally by enlarging or reducing it.

[Parameters]

(array)image: Input image variable

(float)sizeX: Scaling factor in the x direction

(float)sizeY: Scaling factor in the y direction

[Return Value]

(array) On success, returns the resized image variable; on failure, returns null.

[Usage Example]

var image = HVisionModule.readImg("/home/root/hust/05.bmp", 0) //Read the image

var resizeImg1 = HVisionModule.resizeImage(image,0.25,0.25)

var resizeImg2 = HVisionModule.resizeImage(image,1.25,1.25)

HVisionModule.saveImage(resizeImg,"/home/root/hust/resizeImg1.bmp") //Save image to check scaling effect

HVisionModule.saveImage(resizeImg,"/home/root/hust/resizeImg2.bmp")

HVisionModule.clearImage(resizeImg1)

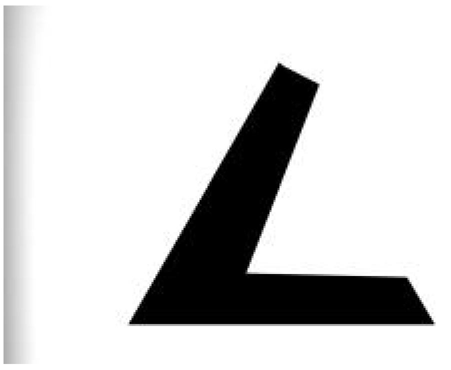
HVisionModule.clearImage(resizeImg2)

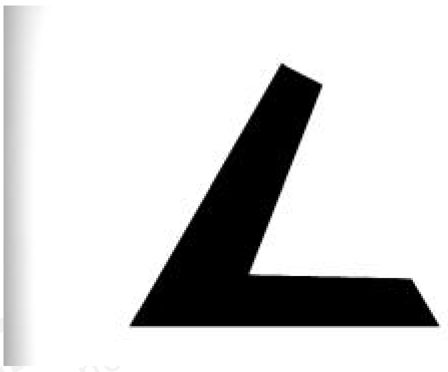
HVisionModule.clearImage(image)

[Execution Effect]

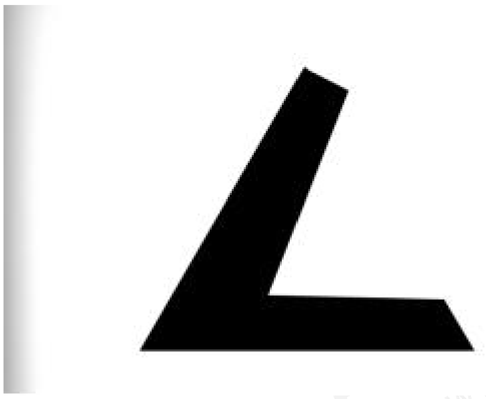
Original Image:
Resolution：3072×2048
Size：6.00 MB

Scaling factor(0.25,0.25)：
Resolution：768×385
Size：385 KB





Scaling factor(1.25,1.25)：
Resolution：3840×2560
Size：9.37 MB



3.2.41 Image Rotation

[Interface]

var rotateImg= HVisionModule.rotateImage(image,rotateAngle)

[Description]

Rotate the image counterclockwise.

[Parameters]

(array)image: Input image variable

(int)rotateAngle: Rotation angle of the image, rotates the image counterclockwise. Only 90, 180, or 270 degrees are allowed. If the rotation angle is 90 or 270, the image width and height will also change.

[Return Value]

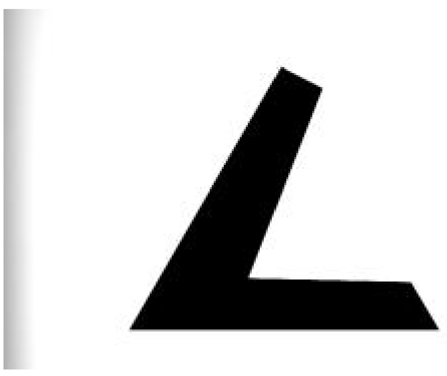
(array)Returns the rotated image variable if successful, otherwise returns null.

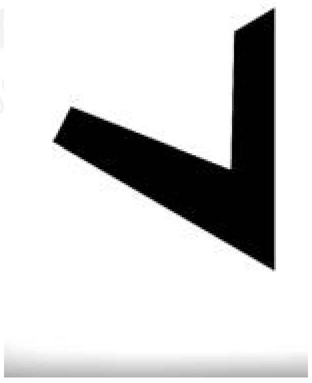
[Usage Example]

```
var image = HVisionModule.readImage("/home/root/hust/05.bmp", 0)      //Read image
var rotateImg = HVisionModule.rotateImage(image,90)
HVisionModule.saveImage(rotateImg,"/home/root/hust/rotateImg.bmp")  //Save image to
view rotation effect
HVisionModule.clearImage(rotateImg)
HVisionModule.clearImage(image)
```

[Execution Effect]

Original Image:: Resolution：768×512 Size：385 KB	Counterclockwise rotation 90 degrees: Resolution：512×768 Size：385 KB





4. Notes and Usage Recommendations

1. BMP Image Size Calculation (assuming a resolution of 4000x3000):

- Grayscale Image:4000*3000/1024/1024 = 11.4MB
- Color Image: 4000*3000*3/1024/1024 = 34.2MB

2. Theoretical Image Transfer Time Calculation (assuming a 11.4 MB image from Camera to Device):

- 100 Mbps Ethernet/100 Mbps Cable (Transfer speed: 12.5 MB/s):11.4/12.5 = 912ms
- Gigabit Ethernet (Transfer speed: 125 MB/s):11.4/125 = 91.2ms
- USB 3.0 Interface (Transfer speed: 375 MB/s):11.4/375 = 30.4ms

3. For High-Speed Projects: It is recommended to use cameras with lower to medium resolution, high frame rates, black-and-white sensors, USB 3.0 interfaces, short exposure times, and hardware triggering to minimize the time spent on image acquisition and transfer.
4. In Controller Systems: You can use variables, IO, interrupts, etc., to control HMI macro execution and perform vision-related functions. Among these, interrupt-based triggering is the most efficient.

5. Disclaimer

This document provides information regarding FINGER CNC series products. No intellectual property rights are granted by this document, either expressly or implicitly, or by estoppel or any other means. Except for the responsibilities stated in the terms and conditions of product sales, our company assumes no other liability. Furthermore, our company makes no express or implied warranties regarding the sale and/or use of this product, including but not limited to warranties of suitability for a particular purpose, merchantability, or non-infringement of any patent, copyright, or other intellectual property rights. Any unauthorized use without our company's written permission constitutes an infringement, and upon discovery, our company will pursue legal action against the infringer in accordance with the law. Our company reserves the right to modify product specifications and descriptions at any time without prior notice.

FINGER CNC

Guangzhou Finger Technology Co.,Ltd

Hotline：020-39389901

Repair Helpline：18127931302

Fax：020-39389903

Postal Code：511495

E-mail：finger@fingercnc.com

Website：www.finger-cnc.com

Address：1F,No. 8, Chengding Street, Zhongcun Street, Panyu District, Guangzhou City, Guangdong Province

Official Website

Official Wechat