Chapter2: Basic operation of OpenCV

1.1 Read, display and save image

1. Image reading

```
img = cv2.imread('tankCar.jpg', 0)
```

// The first parameter is the path of the image, and the second parameter is how to read the image.

cv2.IMREAD_COLOR: Read in a color image. The transparency of the image is ignored, which is the default parameter.

cv2.IMREAD_GRAYSCALE: Read in images in grayscale mode

2.Image displaying

```
cv2.imshow('image',img)
```

// The first parameter is name of window, The window will automatically adjust to the size of the image

// The second parameter is the handle to display the image. However, the window will flash during the execution of the program. You need to add the following statement:

cv2.waitKey(0): Keyboard binding program, waiting for keyboard input.

cv2.destroyAllWindows(): delete any windows we created

cv2.destroyWindow('image'): delete the specified window name

3.Image save

```
cv2.imwrite('car.jpg', img)
```

// The first parameter is the name of the saved file.

// The second parameter is the saved image

4.Image read display save

After running this program, the system will load a picture and display it. Press 's' key on the keyboard to save the image and exit, or press 'ESC' to exit without saving. The source code of the program is located

/home/pi/yahboom/basic_writeAndRead/basic_writeAndRead.py

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
* @par Copyright (C): 2010-2019, Shenzhen Yahboom Tech
* @file         basic_writeAndrRead
* @version      V1.0
* @details
* @par History

@author: longfuSun
"""

import cv2
img=cv2.imread('tankCar.jpg',cv2.IMREAD_COLOR)

cv2.imshow('image',img)

k=cv2.waitKey(0)

#Save the image to the specified file when the keyboard enters "s"
#Exit when the keyboard enters "Esc"
if k==27:
    cv2.destroyAllWindows()
elif k==ord('s'):
    cv2.imwrite('car.jpg',img)
    print('save image successfully')
    cv2.destroyAllWindows()
```

1.2 Read, display and save video

Video is read, displayed and saved.

The functions are cv2.VideoCapture() and cv2.VideoWrite()

Get the video stream from the camera:

Turn on the camera:

cap = cv2.VideoCapture(0)

The parameter 0 indicates the default camera of the device. When the device has multiple cameras, you can change the parameter selection.

Read the video stream of the camera: ret, frame = cap.read() no parameters, but need to be placed in an infinite loop to continuously read to form video.

Freed camera resources: cap.release() no parameters. The camera must be turned off and resources released before the program closes.

Read the video file:

cap = cv2.VideoCapture('filename')

Save video:

Create a VideoWriter object and specify the output file name, specify the video encoding format.

Specify the encoding format:

```
fourcc = cv2.VideoWriter_fourcc(*'XVID')
```

Specify the output file:

```
out = cv2.VideoWriter('output.avi',fourcc,20.0,(640,480))
```

The last parameter is the resolution of the video.

Video read display save : Get the video stream of the camera and save it in the current folder

1.3 Drawing lines, rectangles, circles

Learn to draw different geometry using OpenCV, the related functions: cv2.line(), cv2.circle(), cv2.rectangle(), cv2.putText(), etc.

Draw line**:**

cv2.line(img, startPoint, endPoint, color, thickness)

Draw circle:

cv2.circle(img, centerPoint, radius, color, thickness)

// when the thickness is negative, indicates that the circle is filled

Draw rectangle**:**

cv2.rectangle(img, point1, point2, color, thickness)

Point1 is the upper left vertex and point2 is the other vertex on point1 diagonal

Write text:

cv2.putText(img, text, point, font, size, color, thickness)

text is the text to be written,

point is the lower left coordinate of the first character,

font is the font type,

size is the font size.

Comprehensive: draw lines, draw circles, draw rectangles and text on one picture.

The source code of the program is located **/home/pi/yahboom/draw/draw.py**

The program is shown below:

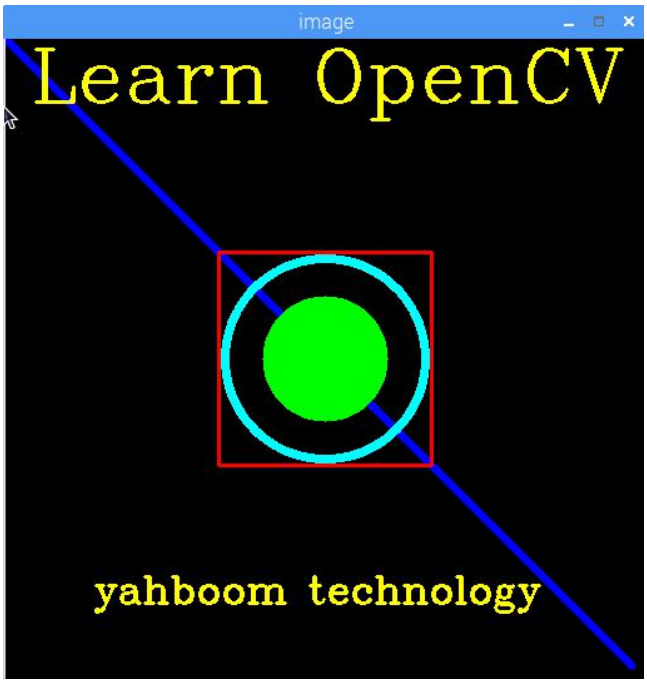
```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
* @par Copyright (C): 2010-2019, Shenzhen Yahboom Tech
* @file         draw
* @version      V1.0
* @details
* @par History
* @author       LongfuSun
"""

import cv2
import numpy as np
img=np.zeros((512,512,3),dtype=np.uint8)
#The position,perimeter,radius and other parameters of the graph are controlled by parameters in
#Draw
cv2.line(img,(0,0),(500,500),(255,0,0),5)
#Drwa circle ,fill circle, last parameter is -1
cv2.circle(img,(255,255),50,(0,255,0),-1)
#Profile of circle
cv2.circle(img,(255,255),80,(255,255,0),5)
#矩形
cv2.rectangle(img,(170,170),(340,340),(0,0,255),2)

#Text
cv2.putText(img,'Learn OpenCV yahboom',(20,50),cv2.FONT_HERSHEY_COMPLEX,2,(0,255,255),2)
cv2.putText(img,'yahboom technology',(70,450),cv2.FONT_HERSHEY_COMPLEX,1,(0,255,255),2)

cv2.imshow('image',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

The phenomenon is shown below:



1.4 OpenCV uses the mouse as a brush/palette

Learn two function: cv2.getTrackbarPos(),cv2.createTrackbar()

Create a slider:cv2.createTrackbar(name,window, min, max, callback)

// The first parameter is the name of the slider, the second parameter is the window in which the slider exists, the third and fourth parameters are the range of the slider, and the fifth is the callback function.

Get the value of the slider: cv2.getTrackbarPos(name, window)

//The first parameter is the name of the slider, and the second parameter is the window in which the slider exists.

The function of the routine we provide: Change the RGB value to implement a palette.

The source code of the program is located **/home/pi/yahboom/trackbar/trackbar.py**

The program is shown below:

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
    * @par Copyright (C): 2010-2019, Shenzhen Yahboom Tech
    * @file        socket server
    * @version      V1.0
    * @details
    * @par History

    @author: longfuSun
"""
import cv2
import numpy as np

def nothing(x):

    pass
img=np.zeros((320,512,3),dtype=np.uint8)
cv2.namedWindow('image')
#Create three trackcar,
#Parameter:"name","target window","initialization threshold","Scale","callback"
cv2.createTrackbar('R','image',0,255,nothing)
cv2.createTrackbar('G','image',0,255,nothing)
cv2.createTrackbar('B','image',0,255,nothing)

while True:
    cv2.imshow('image',img)
    r=cv2.getTrackbarPos('R','image')
    g=cv2.getTrackbarPos('G','image')
    b=cv2.getTrackbarPos('B','image')

    #Drag the trackbar to change the color of the window
    img[:]=[b,g,r]
    if cv2.waitKey(1)&0xFF==27:
        break
cv2.destroyAllWindows()
```

The phenomenon is shown below: