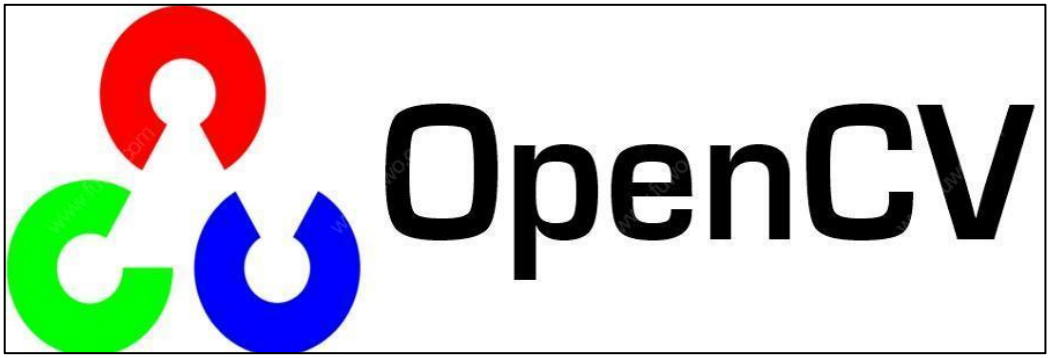
Lesson 2 OpenCV Introduction

1. Introduction

OpenCV (Open Source Capture Vision) is a free computer vision library that can process images and video is used to complete various tasks, such as displaying the signal input by the camera and allowing the robot to recognize objects in life.



OpenCV Icon

Although python has its own image processing library PIL, its function is weaker than OpenCV. OpenCV provides with a complete Python port, Python3.5 and python-opencv libraries have been integrated in the mirroring system we provide file. You can directly use this powerful computer vision library.

2. Storing Form of Image

After recognizing the image, how does computer store different images?

We know that an image is composed of pixels, so how does the computer store and recognize different images? Each pixels can be represented by the R, G, and B values of the three primary colors ranging from 0-255. OpenCV uses a ternary for each pixel array representation, and then save this array to record all the information of the image. But it should be noted that the array of

the three color channels of the RGB image recorded in OpenCV, the recording order will become BGR, so we often call the image is a BGR image.

For other images such as HSV and Lab standards, they are also stored in the form of a multivariate array. An OpenCV the image is a two-dimensional or three-dimensional array of type .array, and an 8-bit grayscale image (only black and white images) is a two-dimensional array, a 24-bit BGR image is a three-dimensional array. For example, for a BRG image, image[0,0,0] the first value represents the Y coordinate or row of the pixel, and 0 represents the top; the second value represents the x coordinate or column of the pixel, and 0 represents the far left; the third value represents the color channel.

These arrays of recorded images can be accessed separately like ordinary Python arrays to obtain a certain color separately the value of the channel, or the image of a certain area of the image.

3. Image Reading and Writing

Image read：cv2.imread(Location，Model)

Location——the location of image that you want read. The location can be an absolute path or a relative path, but note that the usage of path slashes in different operating systems.

Model —— image loading mode. cv2.IMREAD_COLOR specifies to load a color image, but does not load its own Alpha channel (record transparency); cv2.IMREAD_GRAYSCALE specifies to load an image in grayscale mode. cv2.IMREAD_UNCHANGED specifies to load an image as such including alpha channel.

Display image：cv2.imshow("Name", Pic)。

Name —— the window name that will be displayed on the window.

Pic——The image to be displayed (the image object that has been read in using cv2.imread() before).

```
import cv2
a = cv2.imread("camera.png")
cv2.imshow("test", a)
cv2.waitKey()
cv2.destroyAllWindows()
```

4. Video reading and writing

Video can be regarded as a fast switching image, so video reading and writing can actually be regarded as another form of image reading and writing.

Camera initialization：cv2.VideoCapture(Number)

“http://127.0.0.1:8080/?action=stream?dummy=param.mjpg”.

Number—— Camera number. It is 0 in general. Because the camera of Raspberry Pi transmit data through local server, “http://127.0.0.1:8080/?action=stream?dummy=param.mjpg” needs to be filled in.

Read camera frame：Capture.read()。

Capture—— refer to the camera that has been defined before.

For example: display the camera image and stop displaying when press “q” key.

```
import cv2
cap=cv2.VideoCapture('http://127.0.0.1:8080/?action=stream?dummy=param.mjpg')
while(cap.isOpened()):
    ret,frame=cap.read()
    cv2.imshow('test',frame)
    key=cv2.waitKey(1)
    if key &0x00FF==ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```