*Path:*
*/home/pi/Yahboom_project/Raspbot/3.AI Vision course/01.Drive camera/Drive camera.ipynb*

Common API functions used by OpenCV:

**1. cv2.VideoCapture()**

cap = cv2.VideoCapture(0)

The parameter in VideoCapture () is 0, which means Raspberry Pi video0.

(Note: You can view the current camera through the command ls/dev/ )

```
pi@raspberrypi:~ $ ls /dev/
argon-h264mem     media1              shm         tty38      vcio
argon-hevcmem     mem                 snd         tty39      vc-mem
argon-intcmem     memory_bandwidth    spidev0.0   tty4       vcs
argon-vp9mem      mmcblk0             spidev0.1   tty40      vcs1
autofs            mmcblk0p1           stderr      tty41      vcs2
block             mmcblk0p2           stdin       tty42      vcs3
btrfs-control     mqueue              stdout      tty43      vcs4
bus               net                 tty         tty44      vcs5
cachefiles        network_latency     tty0        tty45      vcs6
char              network_throughput  tty1        tty46      vcs7
console           null                tty10       tty47      vcsa
cpu_dma_latency   port                tty11       tty48      vcsa1
cuse              ppp                 tty12       tty49      vcsa2
disk              ptmx                tty13       tty5       vcsa3
dri               pts                 tty14       tty50      vcsa4
fb0               ram0                tty15       tty51      vcsa5
fd                ram1                tty16       tty52      vcsa6
full              ram10               tty17       tty53      vcsa7
fuse              ram11               tty18       tty54      vcsm
gpiochip0         ram12               tty19       tty55      vcsm-cma
gpiochip1         ram13               tty2        tty56      vcsu
gpiochip2         ram14               tty20       tty57      vcsu1
gpiomem           ram15               tty21       tty58      vcsu2
hwrng             ram2                tty22       tty59      vcsu3
i2c-1             ram3                tty23       tty6       vcsu4
initctl           ram4                tty24       tty60      vcsu5
input             ram5                tty25       tty61      vcsu6
kmsg              ram6                tty26       tty62      vcsu7
log               ram7                tty27       tty63      vga_arbiter
loop0             ram8                tty28       tty7       vhci
loop1             ram9                tty29       tty8       video0
loop2             random              tty3        tty9       video1
loop3             raw                 tty30       ttyAMA0    video10
loop4             rfkill              tty31       ttyprintk  video11
loop5             rpivid-h264mem      tty32       ttyS0      video12
loop6             rpivid-hevcmem      tty33       uhid       watchdog
loop7             rpivid-intcmem      tty34       uinput     watchdog0
loop-control      rpivid-vp9mem       tty35       urandom    zero
mapper            serial0             tty36       v4l
media0            serial1             tty37       vchiq
```

cap = cv2.VideoCapture("…/1.avi")

VideoCapture("…/1.avi"), This parameter indicates that if the path of the video file is entered, the video is opened.

**2.cap.set()**

Camera parameters common configuration methods:

capture.set(CV_CAP_PROP_FRAME_WIDTH, 1920);  # Width

```
capture.set(CV_CAP_PROP_FRAME_HEIGHT, 1080);    # Height
capture.set(CV_CAP_PROP_FPS, 30);                # Frame
capture.set(CV_CAP_PROP_BRIGHTNESS, 1);          # Brightness 1
capture.set(CV_CAP_PROP_CONTRAST,40);            # Contrast 40
capture.set(CV_CAP_PROP_SATURATION, 50);         # Saturation 50
capture.set(CV_CAP_PROP_HUE, 50);                # Hue 50
capture.set(CV_CAP_PROP_EXPOSURE, 50);           # Visibility 50
```

**Parameter explanation:**

```
#define CV_CAP_PROP_POS_MSEC  0
// Calculate the current position in milliseconds
#define CV_CAP_PROP_POS_FRAMES     1
// Calculate the current position in frame
#define CV_CAP_PROP_POS_AVI_RATIO  2   // Relative position of the video
#define CV_CAP_PROP_FRAME_WIDTH    3    // Width
#define CV_CAP_PROP_FRAME_HEIGHT   4    // Height
#define CV_CAP_PROP_FPS            5     // Frame rate
#define CV_CAP_PROP_FOURCC         6    // 4 Character encoding
#define CV_CAP_PROP_FRAME_COUNT    7    // Video frames
#define CV_CAP_PROP_FORMAT         8    // Video format
#define CV_CAP_PROP_MODE           9
// Backend specific value indicating the current capture mode.
#define CV_CAP_PROP_BRIGHTNESS     10    // Brightness
#define CV_CAP_PROP_CONTRAST       11   // Contrast
#define CV_CAP_PROP_SATURATION     12 // Saturation
#define CV_CAP_PROP_HUE            13   // Hue
#define CV_CAP_PROP_GAIN           14 // Gain
#define CV_CAP_PROP_EXPOSURE       15 // Exposure
#define CV_CAP_PROP_CONVERT_RGB    16
// Mark whether the image should be converted to RGB.
#define CV_CAP_PROP_WHITE_BALANCE 17 // White balance
#define CV_CAP_PROP_RECTIFICATION 18 // Stereo camera calibration mark (note: only support DC1394 v2)
```

**3.cap.isOpened()**

Return true indicates open camera successful and false indicates open camera failure

**4.ret,frame = cap.read()**

**cap.read ()** reads the video frame by frame. ret and frame are the two return values of the cap.read () function.

ret is a Boolean value, if the read frame is correct, it will return true, If the file has not been read to the end, it returns False.

Frame is the image of each frame, which is a three-dimensional matrix.

**5.cv2.waitKey(n)**

n represents the delay time, if the parameter is 1, it means a delay of 1ms to switch to the next frame of image.

If the parameter is too large, such as cv2.waitKey (1000), it will freeze because of the long delay.

The parameter is 0, such as, cv2.waitKey (0) only displays the current frame image, which is equivalent to video pause.

**6.cap.release() and destroyAllWindows()**

Call cap.release () to release the video.

Call destroyAllWindows () to close all image windows.

**About Code**

Since our entire tutorial runs in JupyterLab, we must understand the various components inside.

Here we need to use the image display component.

**1.Import library**

```
import ipywidgets.widgets as widgets
```

**2.Set Image component**

```
image_widget = widgets.Image(format='jpeg', width=600, height=500)
```

**3.Display Image component**

```
display(image_widget)
```

**4.Open camera and read image**

```
image = cv2.VideoCapture(0)          # Open camera
ret, frame = image.read()            # Read camera data
```

**5.Assignment to components**

#Convert the image to jpeg and assign it to the video display component

```
image_widget.value = bgr8_to_jpeg(frame)
```

```
import cv2
import ipywidgets.widgets as widgets
import threading
import time

#Set camera display component
image_widget = widgets.Image(format='jpeg', width=600, height=500)
display(image_widget)          # display camera component
```

```
#bgr 8 to jpeg format
import enum
```

```
import cv2

def bgr8_to_jpeg(value, quality=75):
    return bytes(cv2.imencode('.jpg', value)[1])
```

```
image = cv2.VideoCapture(0)             # Open camera

# width=1280
# height=960
# cap.set(cv2.CAP_PROP_FRAME_WIDTH,width)    # set width of image
# cap.set(cv2.CAP_PROP_FRAME_HEIGHT,height)    # set height of image

image.set(3,600)
image.set(4,500)
image.set(5, 30)    # set frame
image.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
image.set(cv2.CAP_PROP_BRIGHTNESS, 40)    #set brightness -64 - 64    0.0
image.set(cv2.CAP_PROP_CONTRAST, 50)      #set contrast -64 - 64    2.0
image.set(cv2.CAP_PROP_EXPOSURE, 156)    #set exposure value 1.0 - 5000 156.0

ret, frame = image.read()          # read camera data
image_widget.value = bgr8_to_jpeg(frame)
```

```
while 1:
    ret, frame = image.read()
    image_widget.value = bgr8_to_jpeg(frame)
    time.sleep(0.010)
```

```
image.release()     #After using the object, we need to release the object, otherwise when we use the object again, the system will prompt that the object be occupied, making it unusable.
```

The camera screen is shown below：