1. Camera driver tutorial

1.1 Common API functions in camera drivers

1.1.2. cv2.VideoCapture() function:

cap = cv2.VideoCapture(0)

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

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

cap = cv2.VideoCapture("…/1.avi")

VideoCapture("…/1.avi"), which means that the parameter is the video file path and the video is opened.

1.1.3. cap.set() function

Set camera parameters. Do not modify them at will. 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 rate

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); #Exposure 50

Parameter explanation:

CV_CAP_PROP_POS_MSEC - Current position of the video, in milliseconds or video acquisition timestamp

CV_CAP_PROP_POS_FRAMES - Frame index to be decompressed/acquired in the next step, starting from 0

CV_CAP_PROP_POS_AVI_RATIO - Relative position of the video file (0 - start of the video, 1 - end of the video)

CV_CAP_PROP_FRAME_WIDTH - Frame width in the video stream

CV_CAP_PROP_FRAME_HEIGHT - Frame height in the video stream

CV_CAP_PROP_FPS - Frame rate

CV_CAP_PROP_FOURCC - Four characters representing codec

CV_CAP_PROP_FRAME_COUNT - Total number of frames in the video file

The function cvGetCaptureProperty obtains the specified properties of the camera or video file.

The following are detailed parameters:

#define CV_CAP_PROP_POS_MSEC 0 //Current position calculated in milliseconds

#define CV_CAP_PROP_POS_FRAMES 1 //Current position calculated in frames

#define CV_CAP_PROP_POS_AVI_RATIO 2 //Relative position of the video, from 0 to 1 The first three parameters should be related to video playback and reading related dynamic information

#define CV_CAP_PROP_FRAME_WIDTH 3 //Frame width

#define CV_CAP_PROP_FRAME_HEIGHT 4 //Frame height

#define CV_CAP_PROP_FPS 5 //Frame rate

#define CV_CAP_PROP_FOURCC 6 //4 character encoding method

#define CV_CAP_PROP_FRAME_COUNT 7 //Number of 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 //Boolean flag 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 correction flag (Note: only supports DC1394 v2.x end cur-rently)

1.1.3. cap.isOpened() function:

Return true for success, false for failure

1.1.4. ret,frame = cap.read() function:

cap.read() reads the video frame by frame, ret,frame are the two return values of the cap.read() method. Among them, ret is a Boolean value. If the read frame is correct, it returns True. If the file is not read to the end, its return value is False.

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

1.1.5. cv2.waitKey() function:

The parameter is 1, which means a delay of 1ms to switch to the next frame image. If the parameter is too large, such as cv2.waitKey(1000), it will feel stuck due to the long delay.

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

1.1.6. cap.release() and destroyAllWindows() functions:

cap.release() releases the video, and calls destroyAllWindows() to close all image windows.

1.2 Implementation effect

Source code path:

/home/pi/project_demo/07.AI_Visual_Recognition/01.Camera_Driving/01_Camera_Driving.ipynb

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

image_widget = widgets.Image(format='jpeg', width=640, height=480)  #设置摄像头显示组件 Set up the camera display component
```

```
#bgr8转jpeg格式 bgr8 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)                                    #打开摄像头/dev/video0 Open the camera /dev/video0
width=640
height=480
image.set(cv2.CAP_PROP_FRAME_WIDTH,width)#设置图像宽度 Set the image width
image.set(cv2.CAP_PROP_FRAME_HEIGHT,height)#设置图像高度 Set the image height

# image.set(3,600)
# image.set(4,500)
# image.set(5, 30)  #设置帧率 Setting the frame rate
# image.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
# image.set(cv2.CAP_PROP_BRIGHTNESS, 40) #设置亮度 -64 - 64  0.0 Set Brightness -64 - 64 0.0
# image.set(cv2.CAP_PROP_CONTRAST, 50)   #设置对比度 -64 - 64  2.0 Set Contrast -64 - 64 2.0
```

```
# image.set(cv2.CAP_PROP_EXPOSURE, 156)  #设置曝光值 1.0 - 5000  156.0 Set the exposure value 1.0 - 5000 156.0

ret, frame = image.read()     #读取摄像头数据 Reading camera data
image_widget.value = bgr8_to_jpeg(frame)
```

```
try:
    display(image_widget)  #显示摄像头组件 Display camera components
    while 1:
        ret, frame = image.read()
        image_widget.value = bgr8_to_jpeg(frame)
        time.sleep(0.010)
except KeyboardInterrupt:
    print(" Program closed! ")
    pass
```

```
#使用完成对象记住释放掉对象，不然下一个程序使用这个对象模块会被占用，导致无法使用
#Remember to release the object after using it, otherwise the object module will be occupied by the next program, making it unusable.
image.release()
```