Rectangle detection

Introduction to the experimental results of the routine

The routine code of this section is located at: [Source code summary / 04.Detecting / 02.find_rects.py]

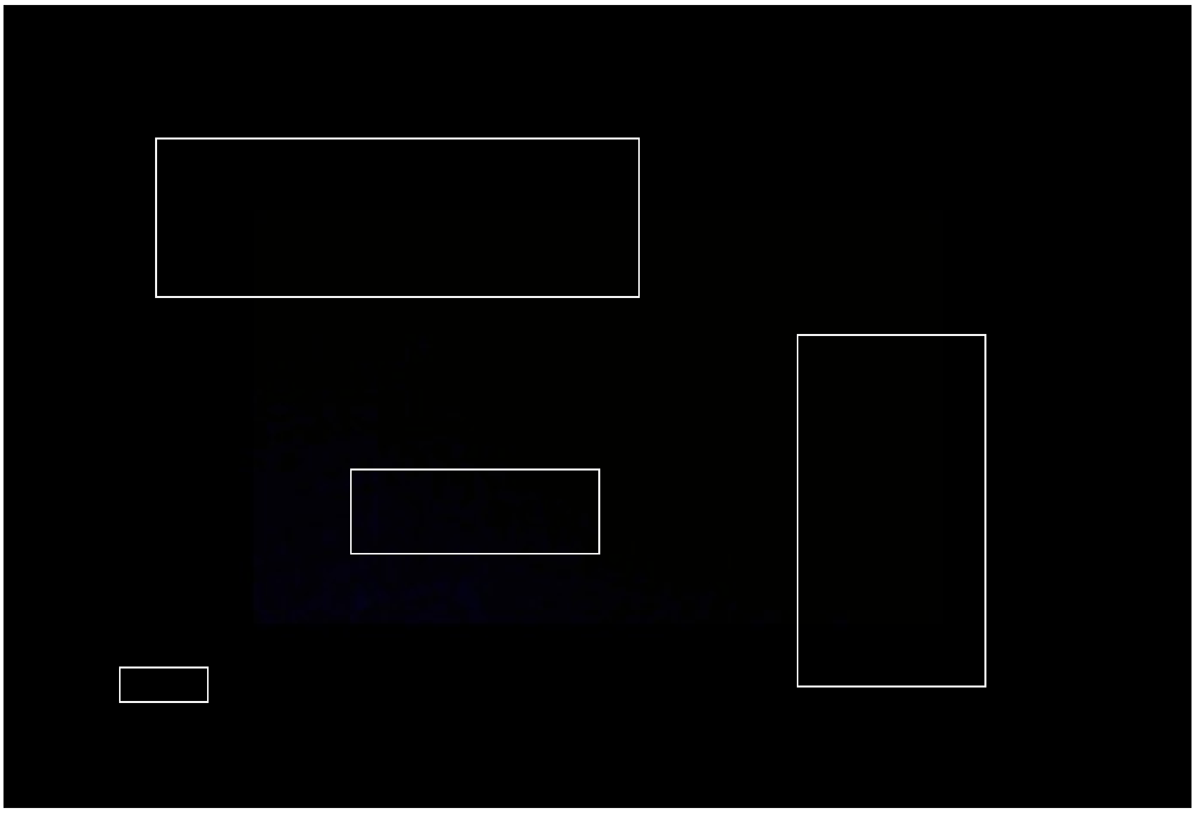
We use CanMV IDE to open the routine code and connect K230 to the computer via USB

Click the run button in the lower left corner of CanMV IDE,

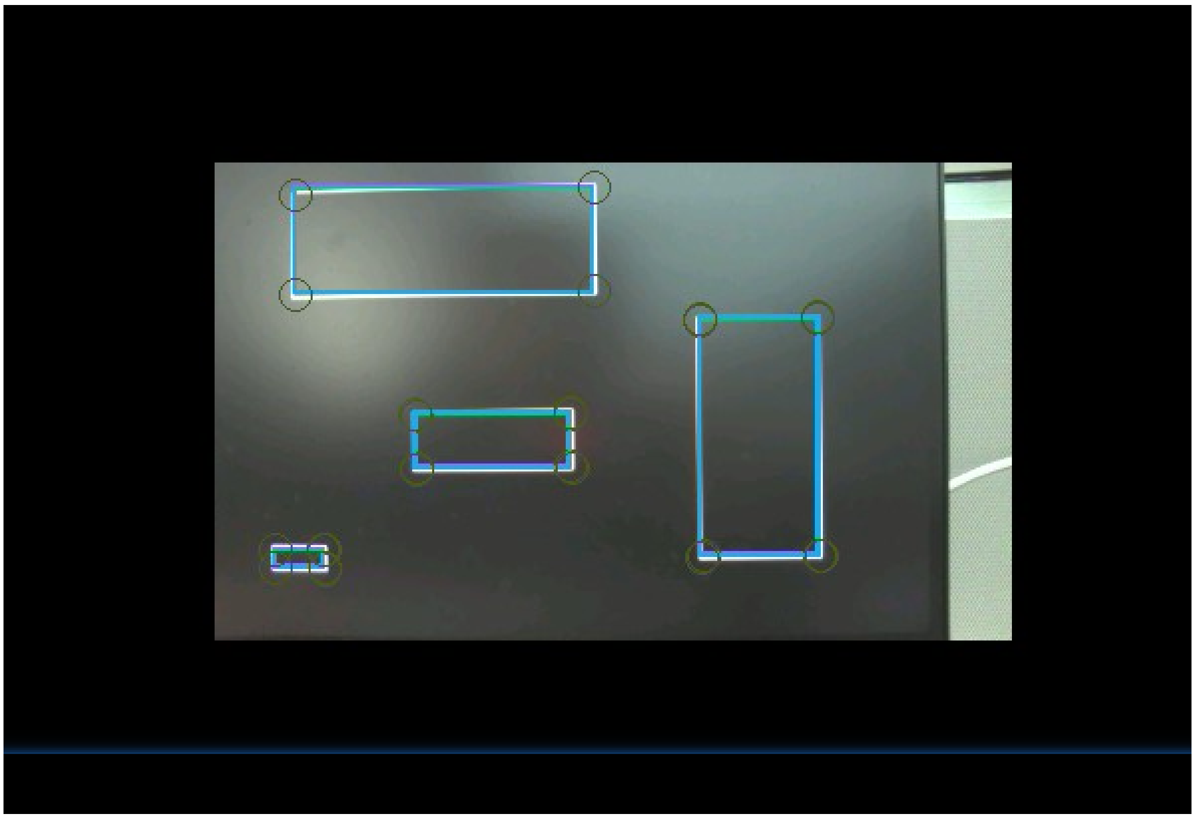
Aim the camera of K230 at the rectangular frame

You can see that the **rectangle** in the picture will be marked on the screen (if there is no screen, look at the frame buffer)

[Original image]



【K230 Identification】



Code explanation

The peripherals we will use in this section are mainly camera modules

The code in this section is logically consistent with the line segment detection in the previous section. The difference is that the find_rects method is used to detect the rectangular border

Rectangular detection is implemented by the find_rects() method in K230, which belongs to the image module

Complete code

```
# 导入必要的模块 Import required modules
import time, os, sys
from media.sensor import *
from media.display import *
from media.media import *

# 图像分辨率设置 Image resolution settings
PICTURE_WIDTH = 400
PICTURE_HEIGHT = 240

# 摄像头配置 Camera configuration
sensor = None

# 显示模式选择 Display mode selection
# 可选: "VIRT", "LCD"
# Options: "VIRT"(Virtual Display), "LCD"
DISPLAY_MODE = "LCD"

# 根据显示模式设置分辨率 Set resolution based on display mode
if DISPLAY_MODE == "VIRT":
```

```python
    DISPLAY_WIDTH = ALIGN_UP(1920, 16)
    DISPLAY_HEIGHT = 1080
elif DISPLAY_MODE == "LCD":
    DISPLAY_WIDTH = 800
    DISPLAY_HEIGHT = 480
else:
    raise ValueError("Unknown DISPLAY_MODE, please select 'VIRT', 'LCD'")

# 创建时钟对象用于FPS计算 Create clock object for FPS calculation
clock = time.clock()

try:
    # 初始化摄像头 Initialize camera
    sensor = Sensor(id=2)
    sensor.reset()

    # 设置图像分辨率和格式 Set image resolution and format
    sensor.set_framesize(width=PICTURE_WIDTH, height=PICTURE_HEIGHT,
chn=CAM_CHN_ID_0)
    sensor.set_pixformat(Sensor.RGB565, chn=CAM_CHN_ID_0)

    # 初始化显示器 Initialize display
    if DISPLAY_MODE == "VIRT":
        Display.init(Display.VIRT, width=DISPLAY_WIDTH, height=DISPLAY_HEIGHT,
fps=60)
    elif DISPLAY_MODE == "LCD":
        Display.init(Display.ST7701, width=DISPLAY_WIDTH, height=DISPLAY_HEIGHT,
to_ide=True)

    # 初始化媒体管理器 Initialize media manager
    MediaManager.init()
    sensor.run()

    while True:
        os.exitpoint()
        clock.tick() # 开始计时 Start timing

        # 捕获图像 Capture image
        img = sensor.snapshot(chn=CAM_CHN_ID_0)

        print("【矩形信息 Line Statistics Start】")


        for r in img.find_rects(threshold = 8000):
            img.draw_rectangle(r.rect(), color = (40, 167, 225),thickness=2)
            for p in r.corners(): img.draw_circle(p[0], p[1], 8, color = (78,
90, 34))
            print(r)
        print("【==============================】")

        # 显示FPS Display FPS
        print(f"FPS: {clock.fps()}")

        # 居中显示图像 Display image centered
        x = int((DISPLAY_WIDTH - PICTURE_WIDTH) / 2)
        y = int((DISPLAY_HEIGHT - PICTURE_HEIGHT) / 2)
        Display.show_image(img, x=x, y=y)
```

```
except KeyboardInterrupt as e:
    print("User Stop: ", e)
except BaseException as e:
    print(f"Exception: {e}")
finally:
    # 清理资源 Cleanup resources
    if isinstance(sensor, Sensor):
        sensor.stop()
    Display.deinit()
    os.exitpoint(os.EXITPOINT_ENABLE_SLEEP)
    time.sleep_ms(100)
    MediaManager.deinit()
```

Code structure

Basic settings:

- Set image resolution (400x240)
- Support two display modes: virtual display (VIRT, 1920x1080) and LCD display (800x480)

Main function flow:

- Initialize the camera, set resolution and RGB565 color format
- Initialize the display
- Enter the main loop:
    - Real-time image capture
    - Detect rectangles in the image
    - Draw borders (blue) and corners (dark green) on the detected rectangle
    - Display FPS (frame rate)
    - Center the image on the screen

Exception handling:

- Capture keyboard interrupts and other exceptions
- Clean up resources at the end of the program: stop the camera, turn off the display, release media resources

Rectangle detection functions

find_rects()

```
image.find_rects([roi=Auto, threshold=10000])
```

Find rectangles in the image using the same quad detection algorithm used to find AprilTags. Works best with rectangles that contrast sharply with the background. AprilTag's quad detection can handle rectangles that are arbitrarily scaled/rotated/sheared. Returns a list of image.rect objects.

roi is the region of interest (x, y, w, h) of the rectangles to copy. If not specified, the ROI is the image rectangle. Only pixels within the roi region are operated on.

Rectangles with a border size (created by sliding the Sobel operator over all pixels on the rectangle edge and adding that value) less than threshold are filtered out of the returned list. The right value for threshold depends on your application/scenario.

Compressed and bayer images are not supported.