第十四周周志

本周学习了利用OpenMV进行巡线的方法。
具体思路是分区域找红色色块来寻中线和找边界转弯点，将数据传给stm32。

参考代码如下：

```
import pyb, sensor, image, math, time
from pyb import UART
import json
uart = UART(3,115200)

red_threshold = [(71, 25, 98, 19, 101, -14)]
black_threshold = [(0, 42, -84, 19, -104, 32)]
#QVGA 320*240
roi_1 = [(40, 0, 240, 40),         #  北
            (40, 200, 160, 40),     # 南
            (0, 0, 40, 240),        #  西
            (280, 0, 40, 240),      #  东
            (30,80,300,80)]          #   中

sensor.reset() # 初始化摄像头
sensor.set_pixformat(sensor.RGB565) # 格式为 RGB565.
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(20) # 跳过10帧，使新设置生效
sensor.set_auto_whitebal(False) # turn this off.
sensor.set_auto_gain(False) # must be turned off for color tracking
clock = time.clock()

buf =[0 for i in range(5)]
m = -1
zhong_x = 0
zhong_y = 0
bei_x = 0
bei_y = 0
nan_x = 0
nan_y = 0
dong_x = 0
dong_y = 0
xi_x = 0
xi_y = 0
black_num = 0
threshold_pixel = 280
uart.init(115200, bits=8, parity=None, stop=1)


def pack_dot_data():
    pack_data=bytearray([0xAA,0xFF,zhong_x>>8,zhong_x,dong_y>>8,dong_y,xi_y>>8,xi_y,
        black_num,0x00,0x00])

    lens = len(pack_data)      #数据包大小
    #pack_data[3] = lens-6;     #有效数据个数

    sc = 0
    ac = 0
    i = 0
    while i<(lens-1):
```

```
        sc = sc + pack_data[i]
        ac = ac + sc
        i=i+1
    pack_data[lens-2] = sc
    pack_data[lens-1] = ac;

    return pack_data




while(True):
        clock.tick()
        m = -1
        isten = 0
        black_num = 0
        img = sensor.snapshot().lens_corr(strength = 1.8, zoom = 1.0)
        for r in roi_1:
            m += 1
            blobs = img.find_blobs(red_threshold, roi=r[0:4],pixels_threshold=100, area_threshol
            #img.draw_rectangle(r[0:4], color=(255,0,0))
            if blobs:
                most_pixels = 0
                largest_blob = 0
                for i in range(len(blobs)):
                #目标区域找到的颜色块（线段块）可能不止一个，找到最大的一个，作为本区域内的目标直线
                    if blobs[i].pixels() > most_pixels:
                        most_pixels = blobs[i].pixels()
                        #merged_blobs[i][4]是这个颜色块的像素总数，如果此颜色块像素总数大于
                        largest_blob = i

                #print(blobs[largest_blob].cx(),blobs[largest_blob].cy())
                if m == 0:
                    bei_x = blobs[largest_blob].cx()
                    bei_y = blobs[largest_blob].cy()
                elif m == 1:
                    nan_x = blobs[largest_blob].cx()
                    nan_y = blobs[largest_blob].cy()
                elif m == 2:
                    dong_x = blobs[largest_blob].cx()
                    dong_y = blobs[largest_blob].cy()
                elif m == 3:
                    xi_x = blobs[largest_blob].cx()
                    xi_y = blobs[largest_blob].cy()
                elif m == 4:
                    zhong_x = blobs[largest_blob].cx()
                    zhong_y = blobs[largest_blob].cy()

                # Draw a rect around the blob.
                img.draw_rectangle(blobs[largest_blob].rect())
```

```
            #将此区域的像素数最大的颜色块画矩形和十字形标记出来
            img.draw_cross(blobs[largest_blob].cx(),
                           blobs[largest_blob].cy())
            buf[m] = 1
        else:
            buf[m] = 0
            if m == 0:
                bei_x = 0
                bei_y = 0
            elif m == 1:
                nan_x = 0
                nan_y = 0
            elif m == 2:
                dong_x = 0
                dong_y = 0
            elif m == 3:
                xi_x = 0
                xi_y = 0
            elif m == 4:
                zhong_x = 0
                zhong_y = 0

blobs = img.find_blobs(black_threshold, roi=r[0:4],pixels_threshold=100, area_threshold=
for b  in blobs:
    #记录大于threshold_pixel值的黑色色块的数量
    if b.pixels() > threshold_pixel:
        img.draw_rectangle(b.rect())
        print(b.pixels())
        black_num = black_num + 1;
print(zhong_x,zhong_y)
uart.write(pack_dot_data())
print(black_num)
print(clock.fps())
```