

图形识别与打靶系统设计



汇报人：陈鲲龙、支喆为、刘翔宇、钱思畅



指导教师：王晓俊



学院：自动化学院





- 1** 设计背景
- 2** 实验目标
- 3** 软件设计流程图
- 4** 设计过程
- 5** 问题及解决方法
- 6** 实验结果展示
- 7** 创新点及总结

1

设计背景

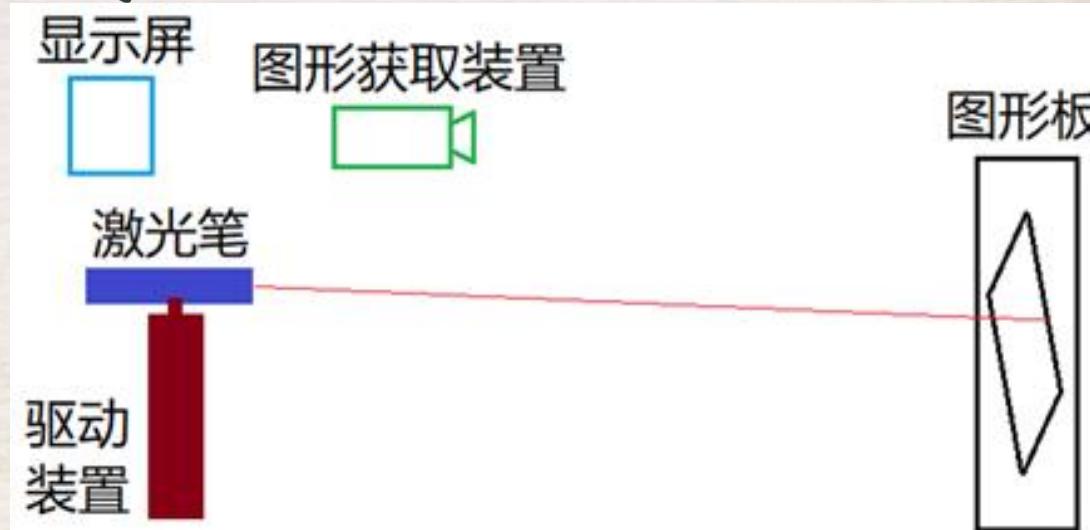
设计背景

激光打点技术是一种广泛应用于工业、科研和医疗等领域的精密测量工具。它通过利用激光束来进行高精度的点位标定和测量。以下是激光打点技术设计背景的详细描述：

1. 精密测量的需求在现代工业制造和科学的研究中，精确的空间定位和尺寸测量至关重要。例如，在制造业中，产品的尺寸和位置精度直接影响到最终产品的质量和性能。在建筑领域，精确的点位标定对于施工和装配也极为重要。
2. 激光打点的优势激光打点技术通过发射高精度的激光束来确定目标点的位置，其主要优势包括：高精度：激光点的位置可以达到亚毫米级的精度。长距离测量：激光束可以在较长距离上保持较高的测量精度。非接触测量：激光打点技术不需要接触被测物体，这对于测量软质或易损物体尤为重要。实时反馈：激光打点设备可以提供实时的测量数据，便于及时调整和校正。
3. 应用领域的广泛性激光打点技术在多个领域都有重要的应用：建筑和施工：用于测量和标定建筑物的结构、设备安装位置等。制造业：在生产线上用于检测和校正机械零部件的位置。科研：用于精密实验中的数据收集和空间定位。医疗：在一些医疗设备中，用于精确定位和导引。

2

实验目标



设计制作图形识别与打靶系统。该系统由图形板、图形获取装置、激光笔及其驱动装置、显示屏等构成（如图所示），能够识别图形板上的图形，控制激光笔光斑在图形板上运动并显示有关信息。

- 识别图形板上默认放置的单个数字图形1、2、3，并发送信息给单片机。
- 当摄像头识别出数字1或2时，进入找点打靶功能，摄像头识别视野范围内的靶点与激光点，向单片机发送信息，单片机控制激光打向目标点，打中目标点则蜂鸣器响。
- 在找点打靶功能中，通过按键控制激光点上下左右运动，摄像头能自动寻找目标图形并识别，向单片机发送数据并控制激光点不脱离靶点。
- 当识别到数字3时，进入识别功能，通过摄像头识别出三角形、正方形、五边形、六边形等图形，并用LCD屏显示，并实时刷新识别的图像。

3

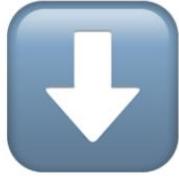
软件设计流程图

单片机

```
main () {
```

基础初始化: ? -Init ()

HAL库
SYS-CLOCK 系统时钟
LED灯
BEEP蜂鸣器
KEY键盘
LCD屏



定时器 TIM3-Init():串口超过 2s 没收到帧头则清零 openmv[] 数组

串口初始化: USART1-Init (): 定义波特率 115200
(并叫出串口中断 USART1_IRQHandler)

定时器 TIM9 的 ch1&2-Init():为两个舵机提供 PWM 信号

舵机初始化: servo-Init ():两个都舵机 90 度回中

```
while (1) {
```

键盘扫描 key-scan ():按键控制两舵机打角



LCD 屏幕打印

switch-case(openmv[5])

两个舵机输出给值

}

}



串口接收到内容进入中断 USART1_IRQHandler

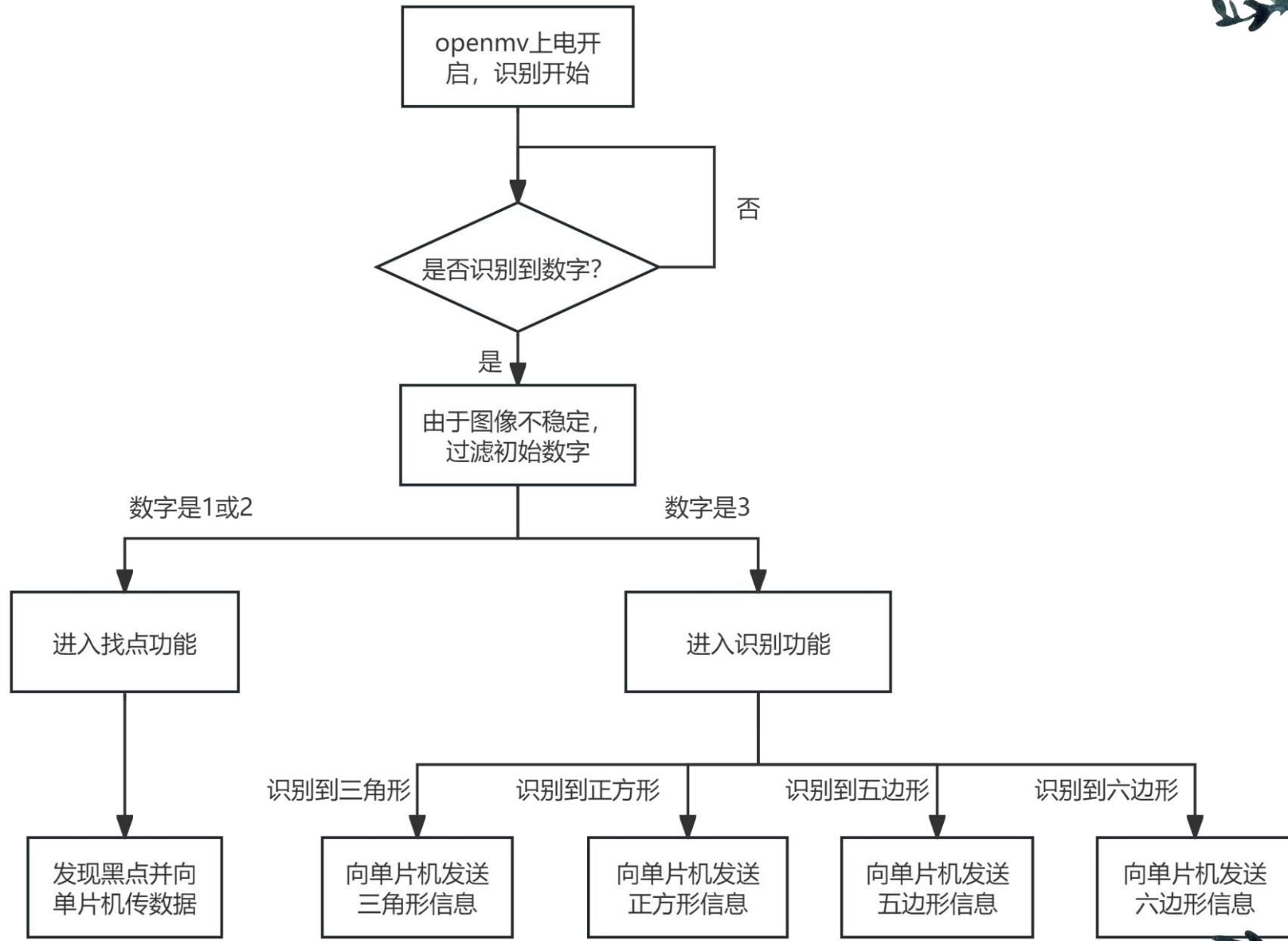


在回调函数 HAL_UART_RxCpltCallback 中
一位一位接收并存入 openmv[] 数组



中断结束回到主程序

摄像头



4

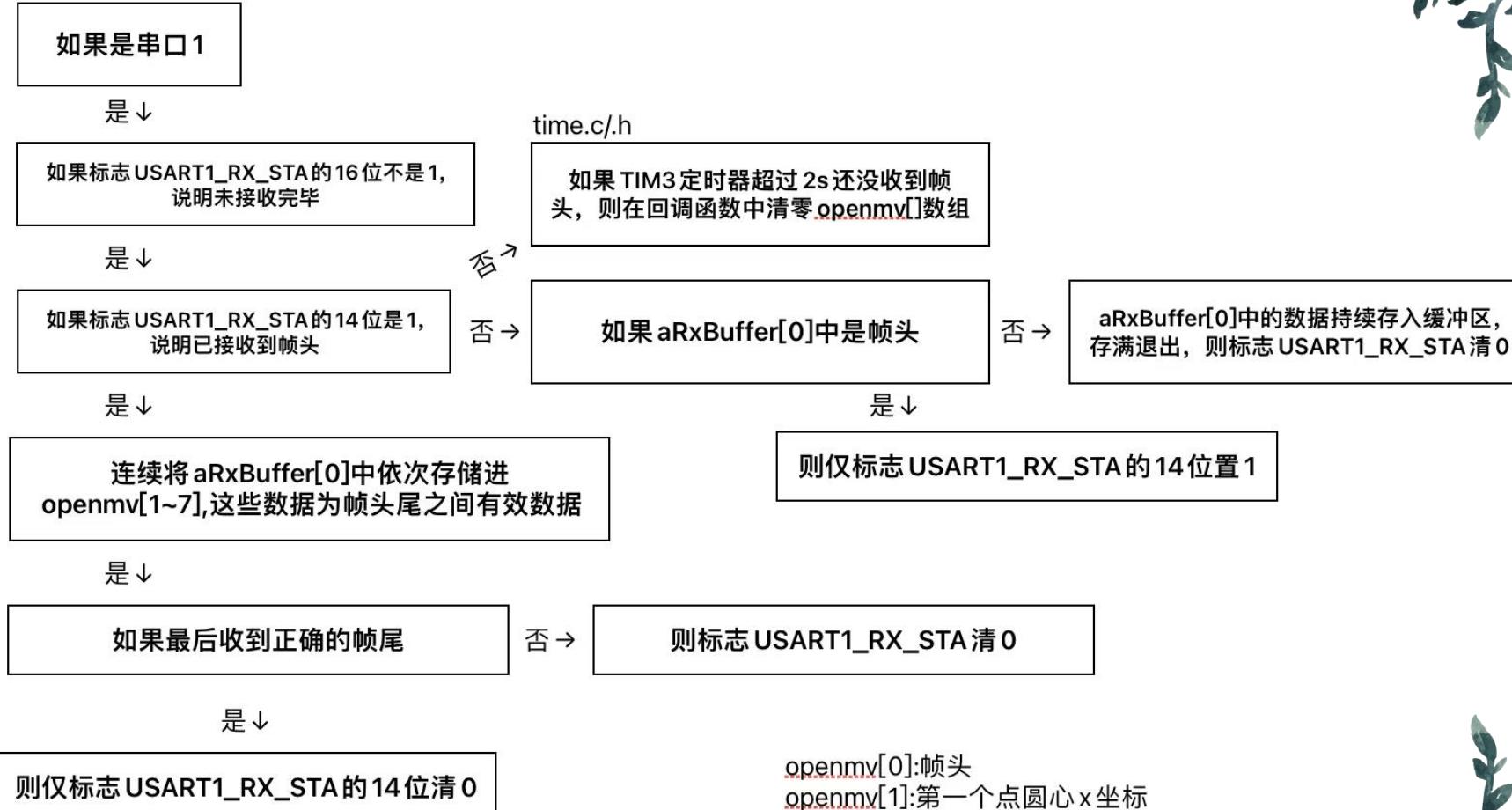
设计过程

二、编写串口接收控制代码

uart.c.h

中断 HAL_UART_Receive_IT(&UART1_Handler, (u8 *)aRxBuffer, RXBUFFERSIZE)
其中 RXBUFFERSIZE=1一位一位接收并存储在 aRxBuffer[0]中

串口接收逻辑核心：回调函数 HAL_UART_RxCpltCallback()



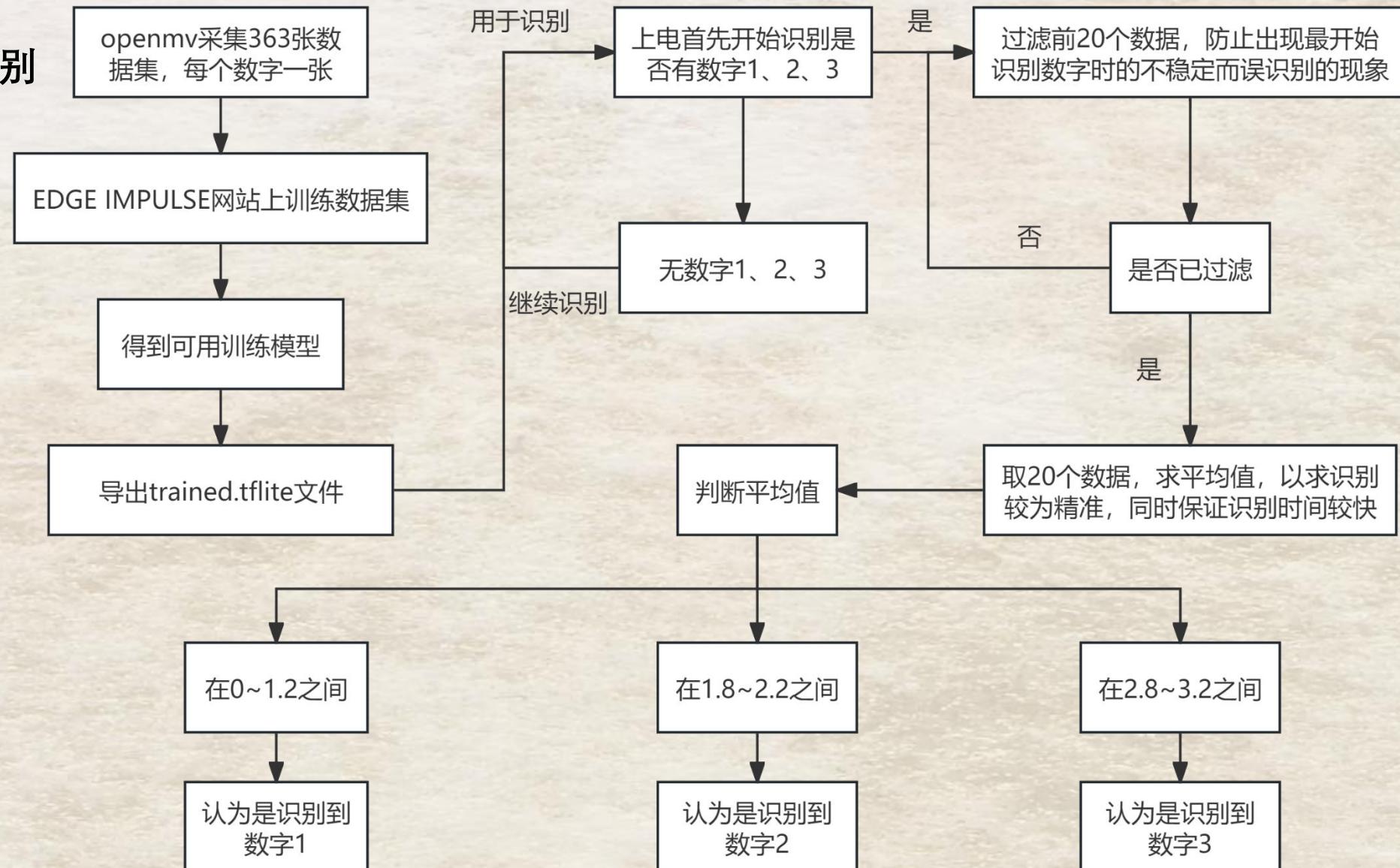
在一次接收 openmv 传来的数据后，openmv 数组中应呈现为：



openmv[0]:帧头
openmv[1]:第一个点圆心 x 坐标
openmv[2]:第一个点圆心 y 坐标
openmv[3]:第二个点圆心 x 坐标
openmv[4]:第二个点圆心 y 坐标
openmv[5]:判断找点模式 / 多边形边数
openmv[6]:激光点 x 坐标
openmv[7]:激光点 y 坐标
openmv[8]:帧尾

三、编写图像识别及激光打靶代码

1. 数字识别



三、编写图像识别及激光打靶代码

1. 数字识别

利用OpenMV采集数据集，然后在EDGE IMPULSE网站上训练数据集并得到用于识别的文件trained.tflite。

The screenshot illustrates the workflow for training a digital recognition model. On the left, the OpenMV IDE's dataset editor and script editor are visible. The dataset editor shows a folder structure for three classes (1.class, 2.class, 3.class) containing numerous image files (00000.jpg to 00019.jpg). The script editor displays a Python script for dataset capture, which includes sensor configuration and a loop for capturing images. To the right, the EDGE IMPULSE website interface shows the results of a trained model. The 'Test data' section lists several samples, each with an expected outcome of 1 and an accuracy of 100%, resulting in a total accuracy of 100.00%.

SAMPLE NAME	EXPECTED OUTC...	ACCURACY	RESULT
1.00059.574...	1	100%	1 1
1.00066.574...	1	100%	1 1
1.00076.574...	1	100%	1 1
1.00078.574...	1	100%	1 1
1.00091.574...	1	100%	1 1
1.00092.574...	1	100%	1 1

三、编写图像识别及激光打靶代码

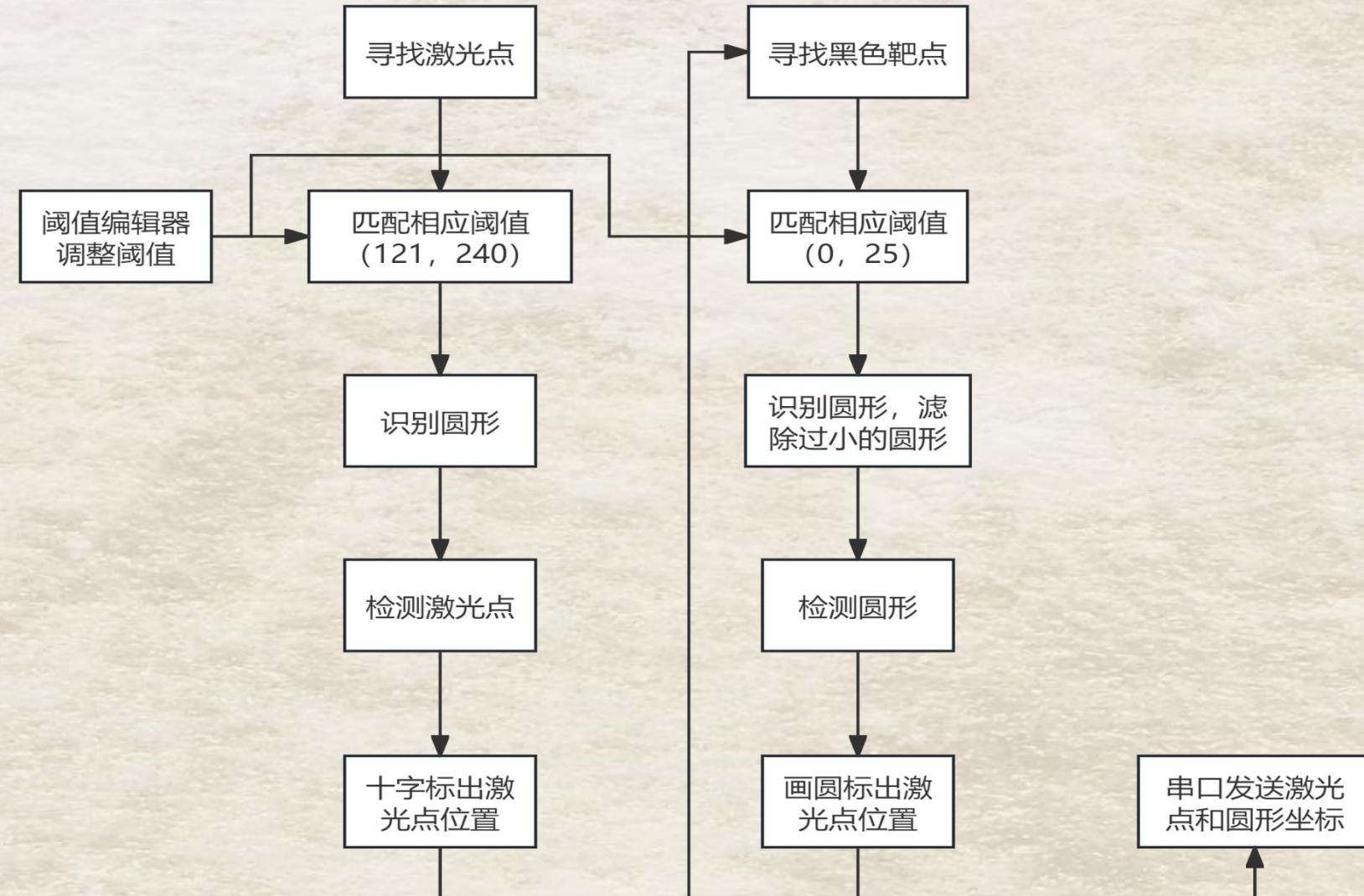
1. 数字识别

过滤初始图像不稳定时的图像，然后多次识别取平均值，作为判定进行找点功能还是识别功能的依据。

```
349  for obj in tf.classify("trained.tflite", img_copy, min_scale=1.0, scale_mul=0.5, x_overlap=0.0, y_overlap=0.0):
350      output = obj.output()
351      number = output.index(max(output))
352      # 在图像中框出识别区域
353      img.draw_rectangle(obj.rect(), color = (255,255,255))
354      # 在框的旁边显示识别到的数字
355      x, y, w, h = obj.rect()
356      img.draw_string(x, y - 10, str(number), color=(255, 255, 255)) # 在框的上方显示数字
357      lcd.write(img)
358      print(number)
359
360      if count_number != 40:
361          count_number += 1
362      if count_number>=20 and count_number < 40:
363          number_sum += number
364      if count_number == 40 and function == 0:
365          number_average = number_sum / 20
366          print('识别停止', number_average)
367
368          if number_average < 1.2 and number_average != 0:
369              number_c = 1
370          elif number_average < 2.2 and number_average > 1.8:
371              number_c = 2
372          elif number_average < 3.2 and number_average > 2.8:
373              number_c = 3
374          else:
375              number_sum = 0
376              number_average = 0
377              count_number = 0
```

三、编写图像识别及激光打靶代码

1. 找点打靶



三、编写图像识别及激光打靶代码

2. 找点打靶

要寻找激光点，以及小圆靶点，并将识别到的两个图形的信息串口发送到单片机。

```
# 寻找激光点
def find_laser(threshold):
    blobs = img.find_blobs([threshold], x_stride=1, y_stride=1)
    if blobs:
        b = blobs[0]
        cx = b.cx()
        cy = b.cy()
        if cx is not None and cy is not None:
            # 绘制十字来表示激光笔的位置
            img.draw_cross(cx, cy)
            print("找到了激光点")
        return cx, cy
    print("未找到激光点")
    return None, None
```

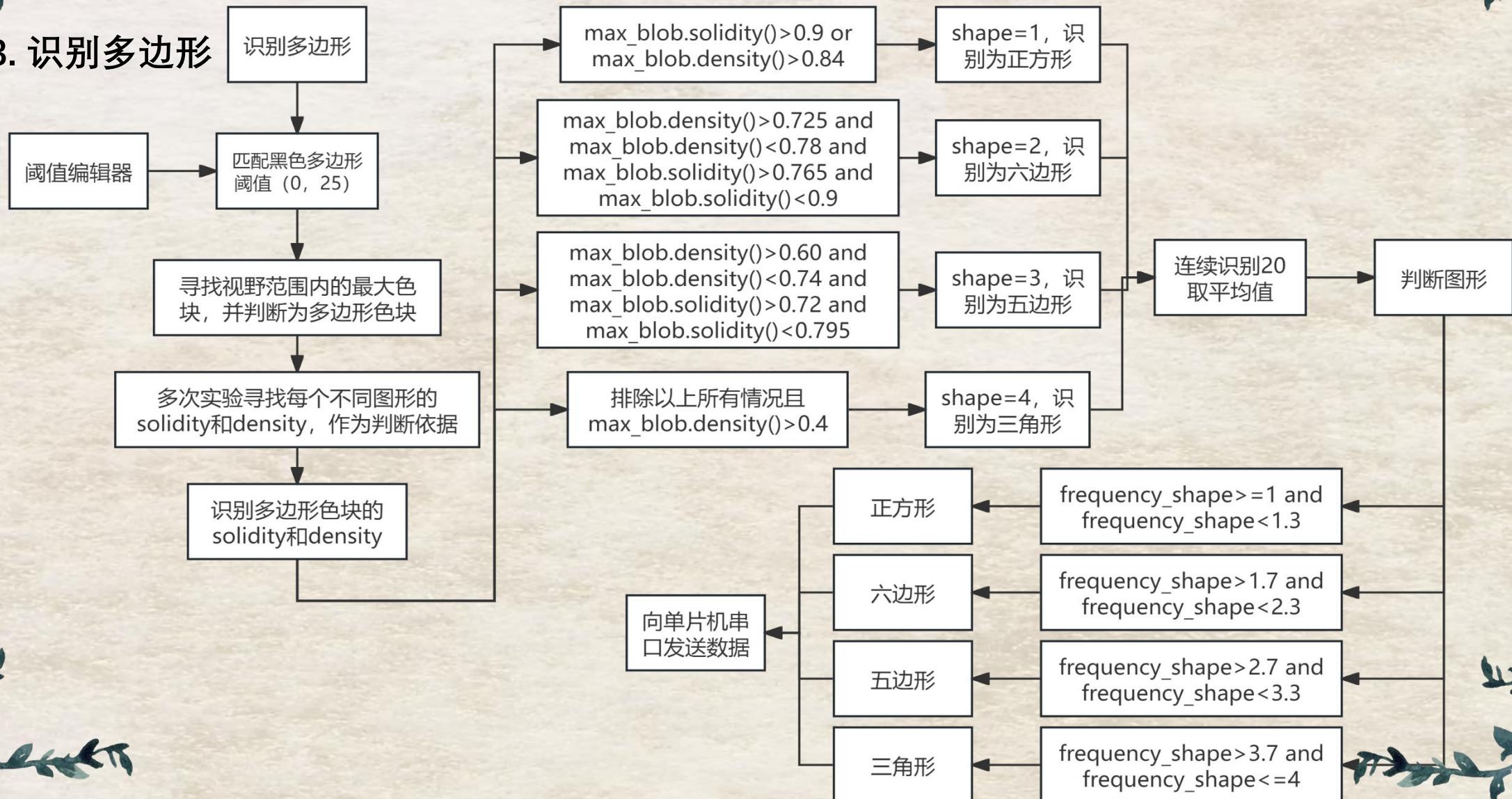
```
168 # 串口发送识别出的圆形信息
169 def send_circle(circles,laser_x,laser_y):
170     data = bytearray([0x0d]) # 起始标志
171
172 #     print(len(circles))
173
174     if laser_x!=None and laser_y!=None:
175         for c in circles:
176             data.extend(struct.pack(">BB", c[0], c[1]))
177 #                 data.extend(struct.pack("B",c[0]))
178 #                 data.extend(struct.pack("B",c[1]))
179 #                 data.append(c[0])
180 #                 data.append(c[1])
```

```
# 识别圆形靶点方法2
def detect_circles(img):
    detected_circles=[]
    blobs=img.find_blobs([threshold_black], x_stride=2, y_stride=2,merge=False)
    if blobs:
        for blob in blobs:
            cr = int((blob.w()+blob.h())/4)
            img.draw_circle(blob.cx(),blob.cy(),cr)
            detected_circles.append([blob.cx(),blob.cy()])
            print('圆形坐标')
            print(blob.cx(),',',blob.cy())
            coordinate_str = f"坐标: {blob.cx()}, {blob.cy()}"
            img.draw_string(blob.cx()-20, blob.cy()+5, coordinate_str)
```

```
182     if len(circles)==1:# 如果只识别到一个圆形
183         data.extend(struct.pack(">BB", 0xFF, 0xFF))
184     #
185     #
186     data.append(1)
187     #
188     data.append(laser_x)
189     data.append(laser_y)
190     data.extend(struct.pack(">BB", laser_x, laser_y))
191     elif len(circles)==2:# 如果识别到两个圆形
192     data.append(2)
193     data.append(laser_x)
194     data.append(laser_y)
195     data.extend(struct.pack(">BB", laser_x, laser_y))
```

三、编写图像识别及激光打靶代码

3. 识别多边形



三、编写图像识别及激光打靶代码

3. 识别多边形

识别白板上的图形，如果是根据边数来向单片机串口发送数字（对应着几边形）。多次识别取平均值，保证识别正确。

```
# 识别多边形
def detect_polygon(max_blob):#输入的是寻找到色块中的最大色块
    shape=-1 #保存形状
    # print(max_blob.solidity())

    if max_blob.solidity()>0.9 or max_blob.density()>0.84:
        img.draw_rectangle(max_blob.rect())
        shape=1#表示矩形

    elif max_blob.density()>0.725 and max_blob.density()<0.78 and max_blob.solidity()>0.765 :
        img.draw_rectangle(max_blob.rect())
        shape=2#表示六边形

    elif max_blob.density()>0.60 and max_blob.density()<0.74 and max_blob.solidity()>0.72 and max_blob.solidity()<0.84:
        img.draw_rectangle(max_blob.rect())
        shape=3#表示五边形

    elif max_blob.density()>0.4:
        img.draw_rectangle(max_blob.rect())
        shape=4#表示三角形

    elif max_blob.density()>0.6:
        img.draw_circle((max_blob.cx(), max_blob.cy(),int((max_blob.w()+max_blob.h())/4)))
        shape=5#表示圆形

    return shape #返回形状
```

```
if count==20:
    n=0
    frequency_shape=frequency_shape/count
    print(frequency_shape)
    if frequency_shape>=1 and frequency_shape<1.3:
        print('是正四边形')
        shape=1
        n=4
    elif frequency_shape>1.7 and frequency_shape<2.3:
        print('是正六边形')
        shape=2
        n=6
    elif frequency_shape>2.7 and frequency_shape<3.3:
        print('是正五边形')
        shape=3
        n=5
    elif frequency_shape>3.7 and frequency_shape<4:
        print('是正三边形')
        shape=4
        n=3
    if n!=0:
        send_polygon(n)
        break
frequency_shape=0
count=0
```

四、编写舵机工作PWM信号代码

servopwm.c/h

晶振 8M Hz 倍频 → 主频 168MHz

使用 APB1 (被分频为 84MHz) 下的 TIM9 的 ch1&2

选取 per=1000-1=999

选取 psc=1680-1=1679

} 在 main.c 中初始化时给定参数

使得 PWM 频率为 APB1/ (per+1)(psc+1)=50Hz

即舵机所需的工作在 20ms, 此时 0.5ms~ 2.5ms 对应 0°~ 180°

TIM_SetCompare () 接口函数: compare 参数 50 ~ 250 对应 0.5ms~ 2.5ms 即对应 0°~ 180°, 赋给 TIM9 的 CCR1/2 对应两个舵机, 具体接线引脚定义为 PE5/6

四、编写舵机打角代码

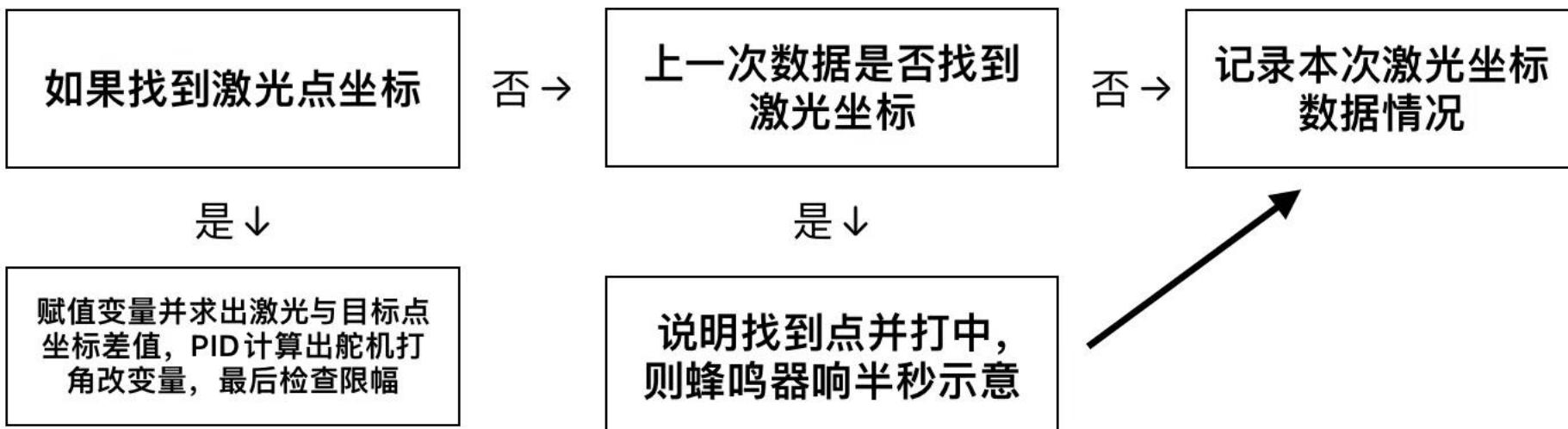
servo.c/.h

宏定义舵机的最大/最小/中值角度值

SetServoAngle 接口函数完成 $0^\circ \sim 180^\circ$ 角度值与 $50 \sim 250$ PWM compare 值之间的换算，并调用 TIM_SetCompare 接口函数

anglelimit 函数是舵机的角度限幅保护

激光找点核心函数：lasertrack ()



5

问题及解决方法

一、舵机控制遇到的问题

舵机需要工作在周期20ms的PWM波下，而PWM波又通过定时器实现，所以定时器重装载值per和分频值psc的选取尤其关键，在一开始随便给舵机出力值妄图直接试出出力值与舵机打角之间的数量对应关系，但结果发现舵机像电机一样连续转动并没有打到特定角度；这就是因为舵机没有工作在它所应有的频率，所以仔细梳理从外部晶振到舵机角度之间每一环的逻辑，发现原理图晶振为8MHz，经过倍频设置主频为168MHz，而所使用的TIM9是挂在APB1下的，所以被分频为84MHz，所以选择重装载值per为1000-1，分频值psc为1680-1，之所以减一是因为从0开始计数，所以参数次数减一，如此一来舵机才能工作在正确的频率，并且能准确计算出角度和出力的对应关系，并在程序中建立两层接口函数。

`servopwm.c/h`

晶振 8M Hz 倍频 → 主频 168MHz

使用 APB1 (被分频为 84MHz) 下的 TIM9 的 ch1&2

选取 per=1000-1=999

选取 psc=1680-1=1679 } 在 main.c 中初始化时给定参数

使得 PWM 频率为 APB1/ (per+1)(psc+1)=50Hz

即舵机所需的工作在 20ms，此时 0.5ms~ 2.5ms 对应 0°~ 180°

TIM_SetCompare () 接口函数：compare 参数 50 ~ 250 对应 0.5ms~ 2.5ms 即对应 0°~ 180°，赋给 TIM9 的 CCR1/2 对应两个舵机，具体接线引脚定义为 PE5/6

二、串口接收遇到的问题

受到了串口例程的指导的同时也受到了些许误导，所以在初期出现了串口有时能接受数据，有时无法接受数据的情况，所以仔细梳理HAL库串口通信的调用流程和逻辑，发现只要串口接收到数据则叫出统一中断接口函数USART1_IRQHandler，其中HAL_UART_Receive_IT(&UART1_Handler, (u8 *)aRxBuffer, RXBUFFERSIZE)的作用为将接收到的RXBUFFERSIZE位数据存入aRxBuffer数组，之后会回调HAL_UART_RxCpltCallback函数，我们对于接受到的数据的处理应当是写在此回调函数中，我在例程的基础（即一位一位接收）上，重新改写了回调函数的处理逻辑：→

uart.c/h

中断 HAL_UART_Receive_IT(&UART1_Handler, (u8 *)aRxBuffer, RXBUFFERSIZE)
其中 RXBUFFERSIZE=1一位一位接收并存储在aRxBuffer[0]中

串口接收逻辑核心：回调函数 HAL_UART_RxCpltCallback()

如果是串口1

是 ↓

如果标志 USART1_RX_STA 的 16 位不是 1，
说明未接收完毕

是 ↓

如果标志 USART1_RX_STA 的 14 位是 1，
说明已接收到帧头

是 ↓

连续将 aRxBuffer[0] 中依次存储进
openmv[1~7]，这些数据为帧头尾之间有效数据

是 ↓

如果最后收到正确的帧尾

是 ↓

则仅标志 USART1_RX_STA 的 14 位清 0

time.c/h

如果 TIM3 定时器超过 2s 还没收到帧头，则在回调函数中清零 openmv[] 数组

否 →

如果 aRxBuffer[0] 中是帧头

否 →

aRxBuffer[0] 中的数据持续存入缓冲区，
存满退出，则标志 USART1_RX_STA 清 0

是 ↓

则仅标志 USART1_RX_STA 的 14 位清 0

在一次接收 openmv 传来的数据后，openmv 数组中应呈现为：

{
openmv[0]:帧头
openmv[1]:第一个点圆心 x 坐标
openmv[2]:第一个点圆心 y 坐标
openmv[3]:第二个点圆心 x 坐标
openmv[4]:第二个点圆心 y 坐标
openmv[5]:判断找点模式 / 多边形边数
openmv[6]:激光点 x 坐标
openmv[7]:激光点 y 坐标
openmv[8]:帧尾

OpenMV识别红色激光

原创

不是笨小孩w 于 2023-08-02 11:19:24 修改 阅读量8.8k 收藏 179 点赞数 49

版权

文章标签: python 开发语言 算法



GitCode 开源社区 文章已被社区收录

加入社区

本篇讲述强光下用OpenMV识别激光点，并实时发送位置，这里介绍一下识别激光的方法，使用的是色块识别。但是激光点面积很小，而且在黑色区域容易被吞掉。因此对图像本身做一定的处理，比如调节曝光度等。主要就是采集，[二值化](#)，找块，显示。

sensor.set_auto_exposure(False, exposure_us=1400)改变openmv的曝光度

```
1 sensor.set_auto_gain(False)
2 sensor.skip_frames(20)
3 sensor.set_auto_exposure(False, exposure_us=1400)
4 sensor.set_auto_whitebal(False)
```

```
11 sensor.reset()
12 #sensor.set_pixformat(sensor.RGB565)
13 sensor.set_pixformat(sensor.GRAYSCALE)
14 #sensor.set_framesize(sensor.QVGA)
15 sensor.set_framesize(sensor.QQVGA)
16 #sensor.set_windowing((240, 240))
17 sensor.skip_frames(time = 2000)
18 sensor.set_auto_gain(False)
19 sensor.set_auto_exposure(False, exposure_us=20000)
20 sensor.set_auto_whitebal(False)
21 #sensor.set_vflip(True)
22 #sensor.set_hmirror(True)
```

激光点识别的困境

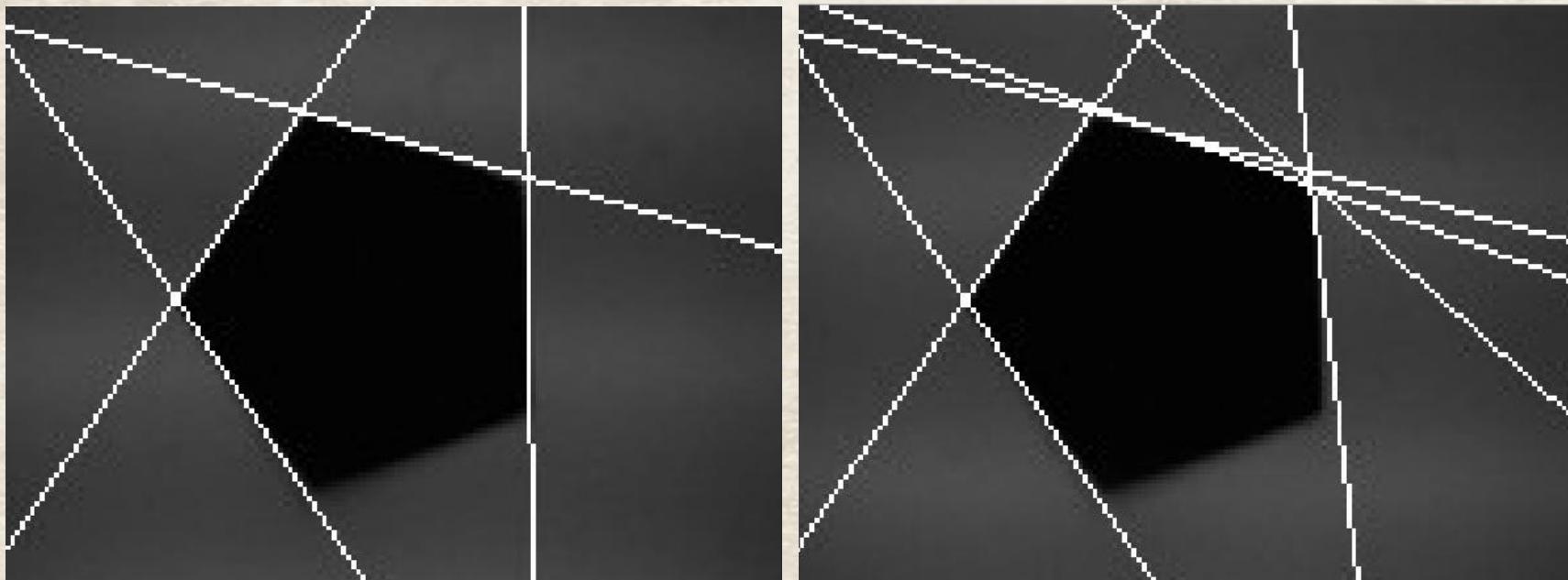
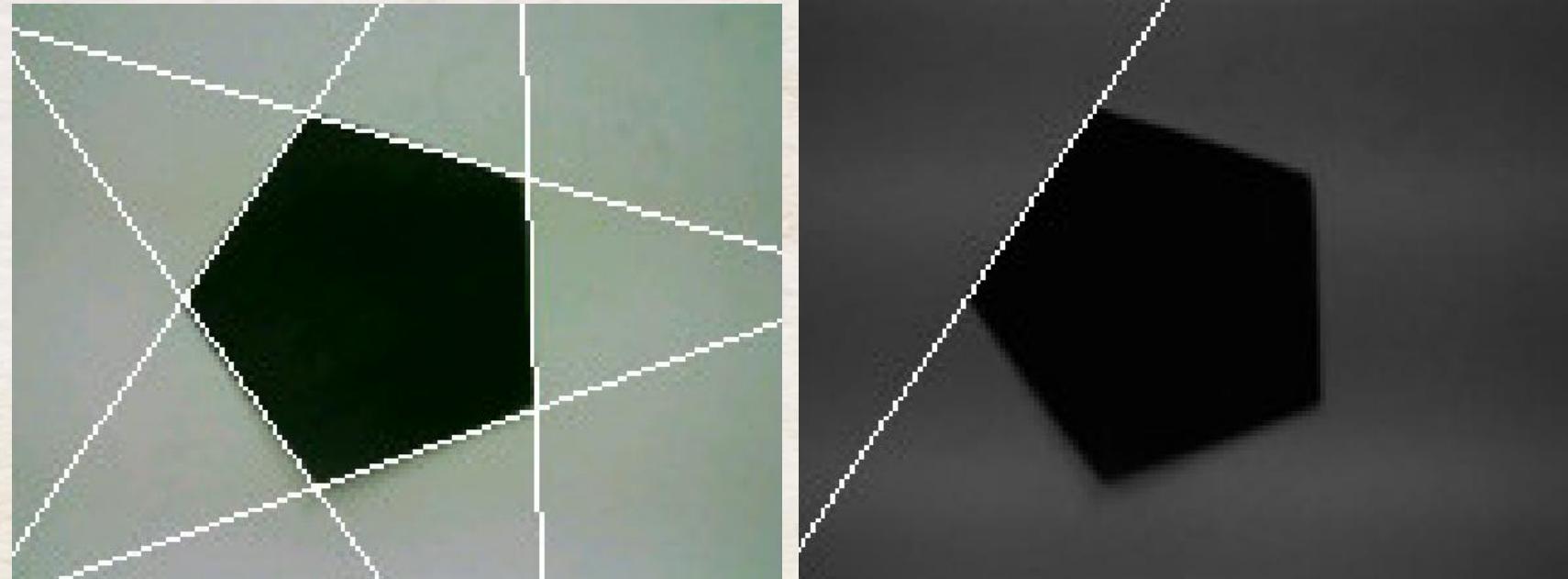
激光点自身的光点不够明亮，在rgb彩图的情况下通过指定较小的阈值范围可以识别但不稳定，但是在灰度图中难以识别。

采用继电器，提高激光亮度，并且设定曝光值，使得激光点与环境对比明显，提高了识别精度和稳定性。

多边形识别的困境

OpenMV官方教程自带的找线方法识别时，如果是RGB彩图并且无曝光调节加暗，识别效果很好

但如果是灰度图并且加了曝光调暗图像，识别情况变差，出现线条误识别



多边形识别的困境

借鉴CSDN上“万无一失的OpenMV识别矩形、圆形、三角形方法”，选择根据
max_blob.solidity()和
max_blob.density()来
判断形状

万无一失的OpenMV识别矩形、圆形、三角形方法

原创 Huiyeee 于 2022-08-20 21:06:15 发布 阅读量1.6w 收藏 355 点赞数 36

文章标签：单片机 视觉检测

版权



GitCode 开源社区 文章已被社区收录

加入社区

density()

返回 blob 的密度比率。这是 blob 中的像素数除以其边界框区域。低密度比率通常意味着锁定不太好。结果在 0 到 1 之间。

extent()

blob.density() 的别名。

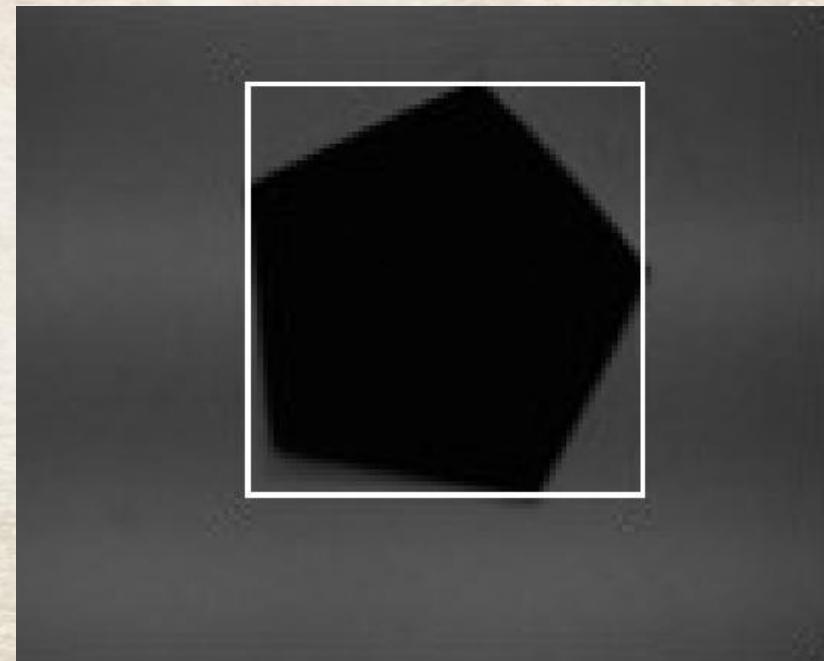
compactness()

类似于 blob.density()，但是使用 blob 的周长来测量对象的密度，因此更准确。结果在 1 之间。

solidity()

类似于 blob.density()，但是使用最小区域旋转矩形与边界矩形来测量密度。结果在 1 之间。

```
def detect_polygon(max_blob):#输入的是寻找到色块中的最大色块
    shape=-1 #保存形状
    # print(max_blob.solidity())
    if max_blob.solidity()>0.9 or max_blob.density()>0.84:
        img.draw_rectangle(max_blob.rect())
        shape=1#表示矩形
    elif max_blob.density()>0.725 and max_blob.density()<0.78 and max_blob.solidity()>0.7:
        img.draw_rectangle(max_blob.rect())
        shape=2#表示六边形
    elif max_blob.density()>0.60 and max_blob.density()<0.74 and max_blob.solidity()>0.72:
        img.draw_rectangle(max_blob.rect())
        shape=3#表示五边形
    elif max_blob.density()>0.4:
        img.draw_rectangle(max_blob.rect())
        shape=4#表示三角形
```



bytearray(b'\r\xff\xff\xff\xff\x05\xff\xff\n')

3.0

是正五边形

bytearray(b'\r\xff\xff\xff\xff\x05\xff\xff\n')

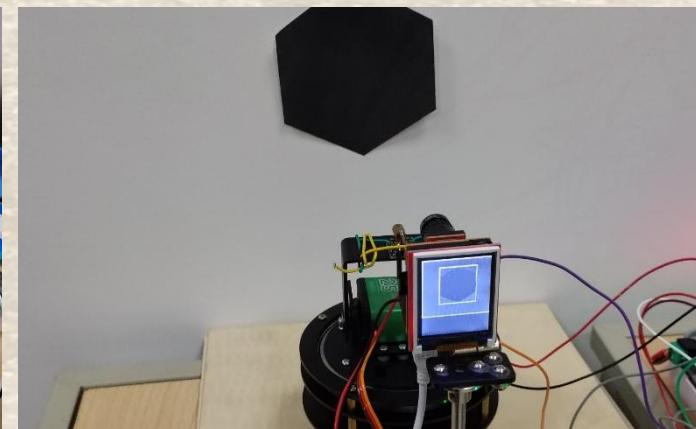
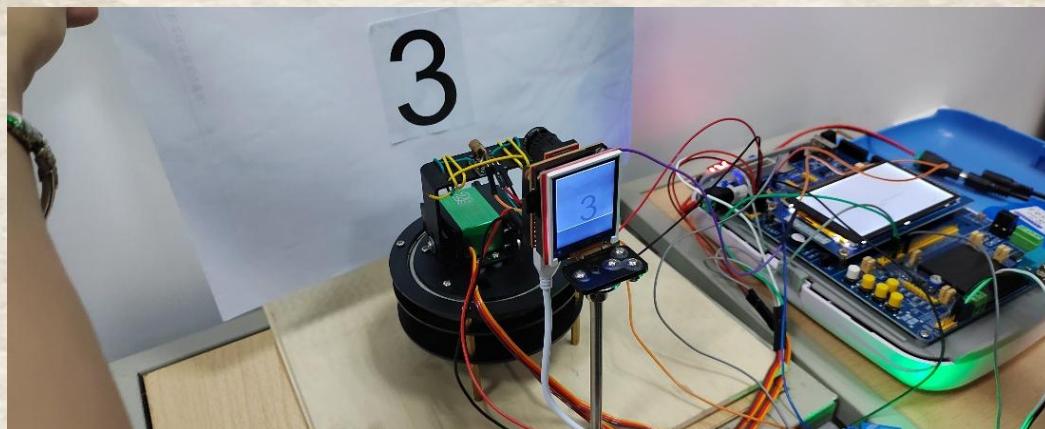
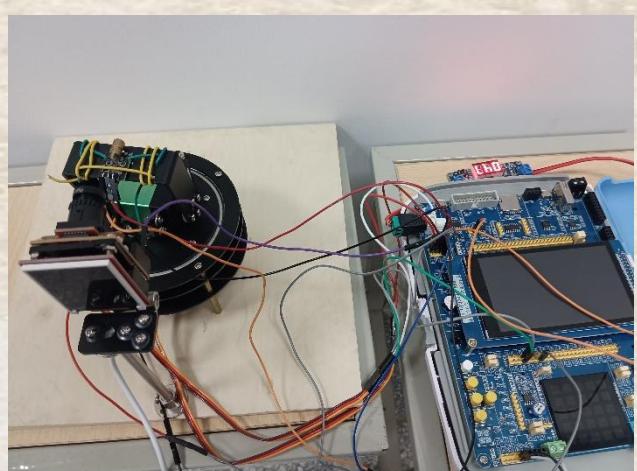
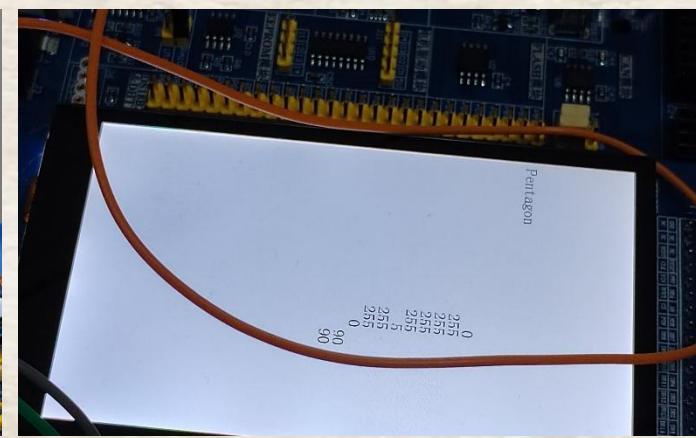
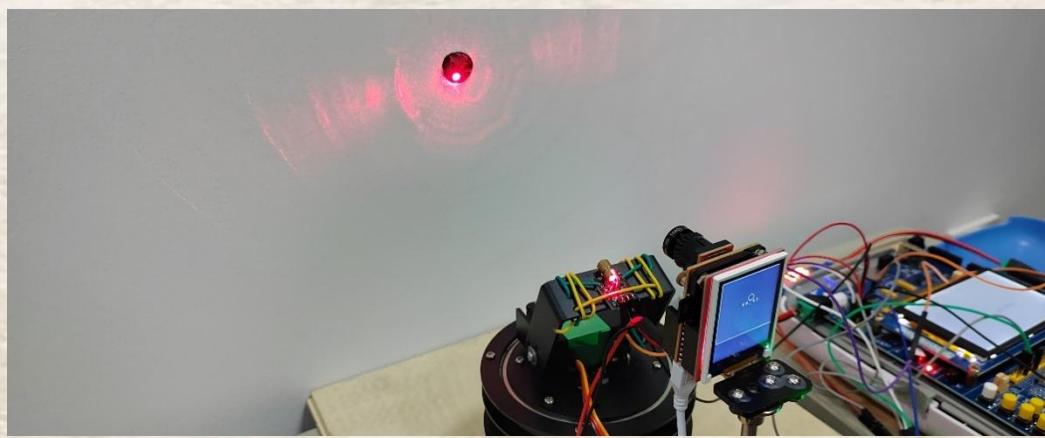
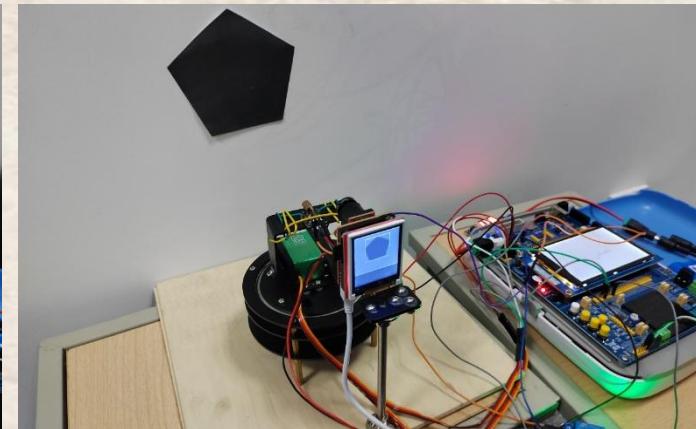
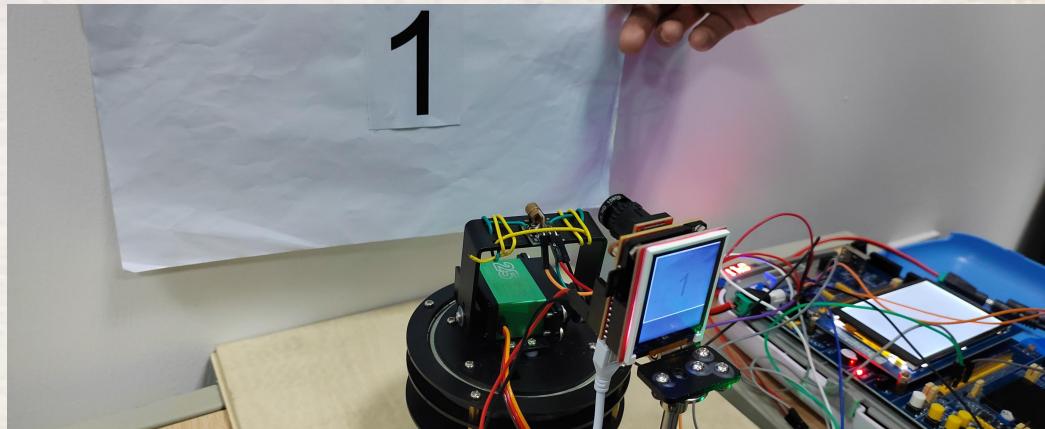
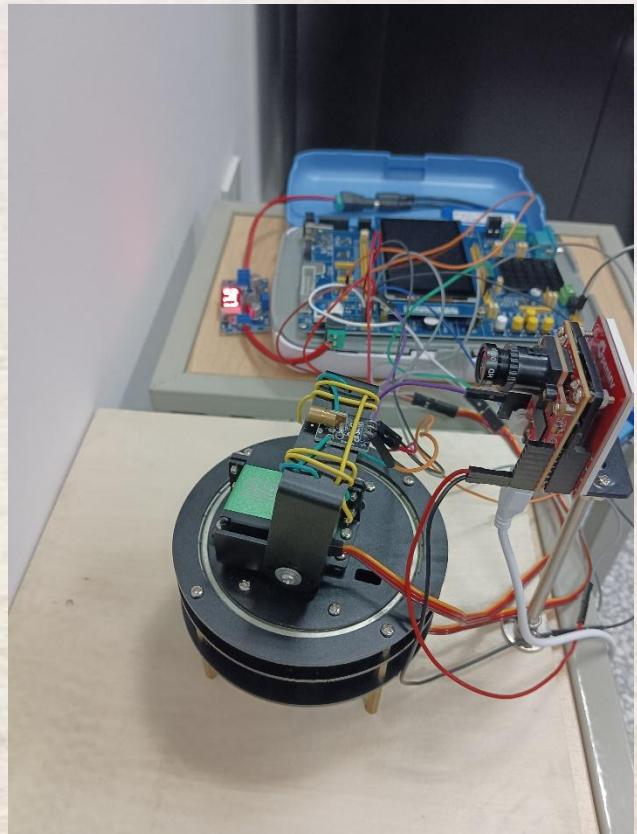
3.0

是正五边形

bytearray(b'\r\xff\xff\xff\xff\x05\xff\xff\n')

6

实验结果展示



7

创新点及总结

一、小组分工

08022311陈鲲龙 串口接收、舵机控制、主函数代码编写，报告PPT流程图制作

08022317支皓为 所有openmv图像识别相关部分，报告PPT流程图制作

61522509钱思畅 报告PPT流程图制作，所有基础I0元件初始化编写

08022112刘翔宇 所有基础实验，LCD屏幕代码编写，报告PPT流程图制作

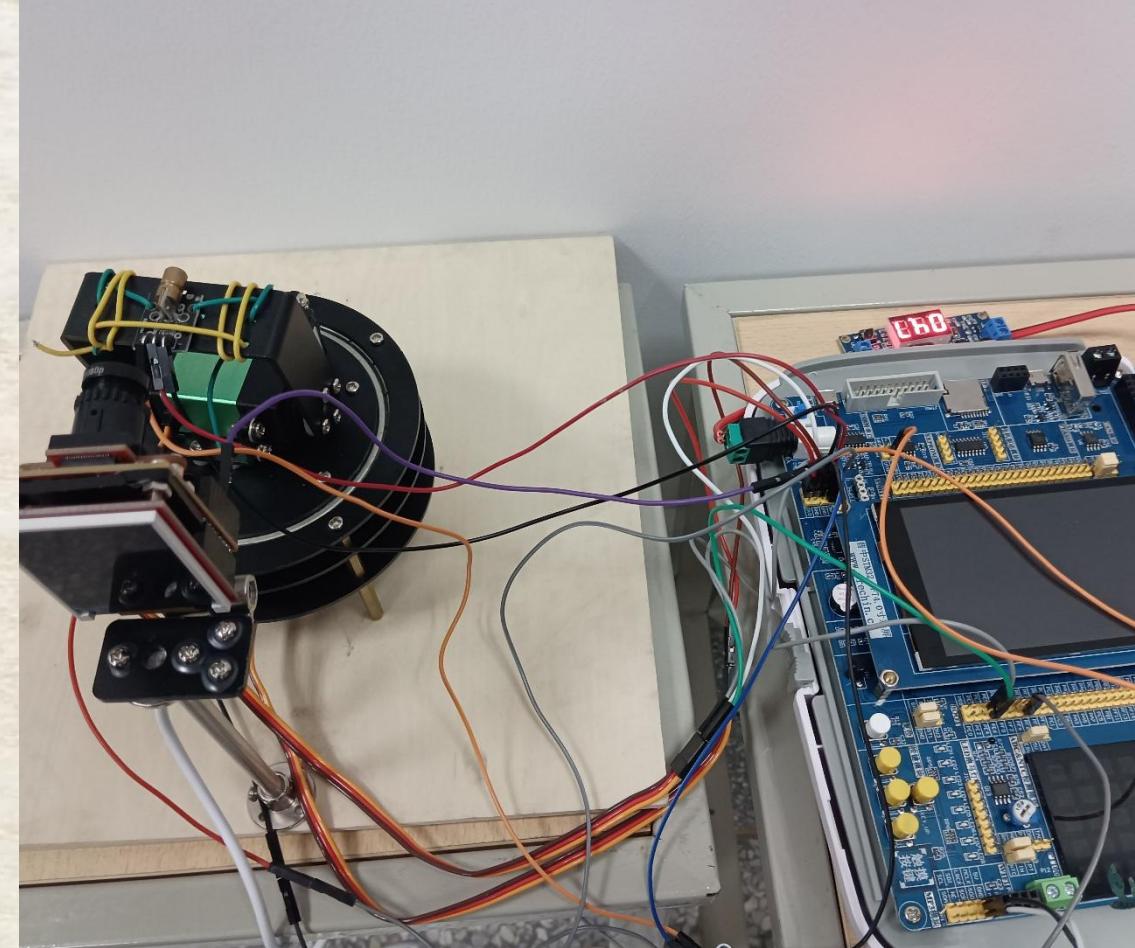
二、创新点

将OpenMV与STM32结合使用的创新点在于构建一个智能视觉控制系统。OpenMV负责实时图像处理，如目标检测和识别，而STM32则接收处理结果并执行相应的控制操作，如驱动电机或调整设备。这种结合能够将高效的图像分析与灵活的控制系统整合，提高自动化设备的智能化水平，实现快速响应和精确控制。

OpenMV识别多边形放弃了官网上传统的利用canny算子和霍夫变换进行直线识别。然后判断直线边数的方法。借鉴CSDN上“万无一失的OpenMV识别矩形、圆形、三角形方法”，选择根据max_blob.solidity()和max_blob.density()来判断形状，使得判断形状稳定性提高。

二、实验小结

本次单片机实验通过搭建基础电路、编写控制程序、调试系统，成功实现了课设的功能。实验过程中，我加深了对单片机工作原理的理解，掌握了硬件连接与编程技巧，同时提升了问题解决能力。遇到的一些电路和编程问题，通过仔细调试和修正得以解决，进一步提高了实践操作的能力。此次实验不仅巩固了理论知识，还增强了动手实践的经验，为未来的学习打下了坚实的基础。



谢谢大家