电子系统导论实验报告

实验 9 【绕桩竞速】

指导教师：邹卓

学生姓名：李晓婉 学生姓名：陈筠 学生姓名：贺祺

学　　号：22307130364 学　　号：22307130508 学　　号：22307130442

专　　业：技术科学试验班 专　　业：技术科学试验班 专　　业：技术科学试验班

日　　期：2023.5.23

一、 实验目的：

在赛道中间随机设置障碍物，利用提供的地面引导线，通过摄像头等传感器完成自主避障和前行，在走廊里进行指定距离的绕桩竞速。

二、 实验原理：

(一) Opencv和图像处理

OpenCV（开源计算机视觉库）是一个流行的开源计算机视觉库，提供了一整套图像和视频处理工具和算法。可以结合 V4L 驱动以支持实验使用的摄像头模块，作为 opencv 的输入源。

(二) 图像阈值处理

图像阈值处理是一种广泛应用的分割技术，利用图像中要提取的目标物与其背景在灰度特性上的差异，把图像视为具有不同灰度级的两类区域（目标和背景）的组合，选取一个合适的阈值，以确定图像中每个像素点应该属于目标还是背景区域，从而产生相应的二值图像。

实验中对图像进行了多次处理：

a) 将图像从 RGB 色彩空间转换到 YCrCb 色彩空间。YCrCb 色彩空间将 RGB 图像表示为亮度（Y）和两个色度信号（Cr 和 Cb）的组合；

b) 提取 Cr 通道。这使得亮度的变化不会影响实验；

c) 二值化。使得跑道和图像的其余部分完全区分开来：跑道是黑色，其余部分是白色。这么做方便了确定方向偏差的逻辑。

三、 实验内容：

1. 代码构架

   避障可用方式，前期想到的主要有三种：超声测距，摄像头识别，计时避障。综合考虑，由于超声测距只有一个传感器，难以准确识别障碍的相对位置，而且几乎无法实现巡线；计时的偶然性较强，时间与地面摩擦，电压大小，占空比大小都有很大关系，难以做到准确避障，而且失误后纠正空间小。故选择摄像头识别引导线，巡线行驶。

   识别时，场地为灰底橙色线，识别方面选择 YCrCb 色彩空间转换后，识别像素色值，判断其黑白。利用图像变化修正方向。

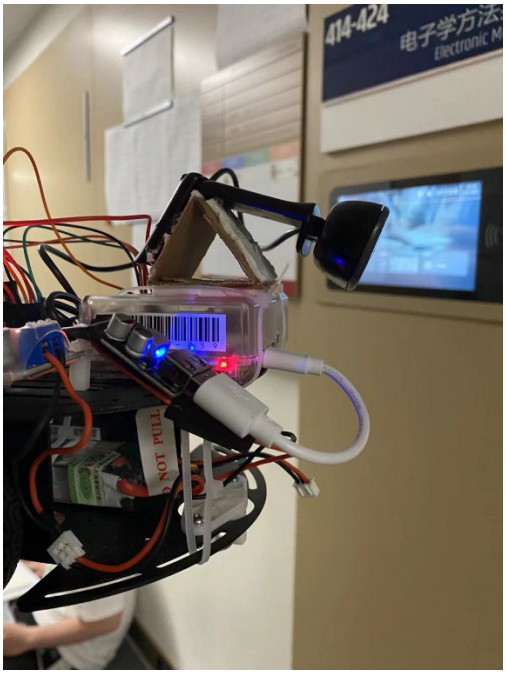
2. 硬件

   使用摄像头时发现，如果摄像头过低，左右偏移时，线的变化很大，一不留神便会丢失视野，程序终止。所以我们尝试将摄像头架高，以获得更大的视野。起初，我们使用较大较重的亚克力块来垫高摄像头，因亚克力块质量过大，小车虽能巡线，却难以提速。

因此，我们改用硬纸板搭建三角形结构将摄像头架高，视野变大，这样容错空间大，便于纠偏。

搭建三角型结构的原因：三角形较四边形结构更为稳定。

摄像头角度及高度大致如图：



除此之外，因原本连接树莓派与直流降压模块的 typec 线过于长且凌乱，可能给竞速带来负面影响，我们将其更换为短且轻便的 typec 线（即上图白色线）。

3. 参数调节

首先使用 pwmkeyboard 文件，找出直行时左右轮占空比。调节左右转纠偏的参数发现，有几种方案，可以只减少一侧占空比，或者只增加一侧占空比，或者改变两侧。实践中发现，减少一侧占空比响应速度更快。所以选择这个。

此外，占空比大小也影响电机启动的速度，若差距过大，会导致启动时，车身由于电机的启动速度差别过大，歪斜严重，线直接消失在视野外，导致巡线失败。

不使用 PID 调节的原因：PID 实现的时间有较大延迟，而规则限制，必须由静止开始，那么 PID 的延迟，让前期小车的直线行驶难度极大，巡线难度高。而且左右修正时，参数差距不能太大，否则，巡线时可能会左右摇摆，速度低，而且转弯时，修正角度过大，可能会导致过弯失败。所以要将所有纠偏的幅度变小，来提升直道时的速度以及过弯成功率。

四、 实验分析：

1. YCrCb 色彩空间相较 RGB 色彩空间在本次实验中的优势：

主要优势是去除亮度对实验的影响。

赛道的颜色为橙色，相较一个黑色的赛道，其 RGB 更容易被亮度所影响。亮度主要受到红、绿、蓝三种原色强度的影响。对于橙色，如果亮度增加，红色和绿色通道的值都会增加；而黑色由于不含有原色，因此亮度增加不会引起 RGB 的变化。

因而橙色的灰度可能受亮度影响，但是其 Cr 通道相对稳定，保证实验可以在不同环境下成功。

2. 摄像头摆放位置的影响：
   摄像头摆放位置直接关乎小车的视野，需要恰当的角度和足够的高度。
3. 调试过程中，我们的电机有时不能正常运转，或是运转速度较正常时慢，经检查发现是电线的接触有问题，更换电线后好转。
4. 代码分析
   代码一：

```
import RPi.GPIO as GPIO
import time
import cv2
import numpy as np
import matplotlib.pyplot as plt

EA, I2, I1, EB, I4, I3 = (13, 19, 26, 16, 20, 21) #  定义引脚
FREQUENCY = 50 # PWM 波的频率为 50Hz
#  初始化 GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup([EA, I2, I1, EB, I4, I3], GPIO.OUT)
GPIO.output([EA, I2, EB, I3], GPIO.LOW)
GPIO.output([I1, I4], GPIO.HIGH)

center = 320 #  定义图像的标准中点 center
#  打开摄像头，图像尺寸 640*480（长*高），opencv 存储值为 480*640（行*列）
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

#  初始化 PWM
pwma = GPIO.PWM(EA, FREQUENCY)
pwmb = GPIO.PWM(EB, FREQUENCY)
pwma.start(0) #  此时 ENA 引脚持续产生占空比为 0 的 PWM 输出
pwmb.start(0) #  此时 ENB 引脚持续产生占空比为 0 的 PWM 输出

count = 0
while 1:
    count += 1
    ret, frame = cap.read() #  摄像头读取图像
```

```
        # RGB 图像转换到 YCrCb 空间
        YCrCb_orange = cv2.cvtColor(frame,cv2.COLOR_BGR2YCrCb)
        # 提取 Cr 通道并高斯去噪，（3,3）表示高斯核大小
        Cr_orange = cv2.GaussianBlur(YCrCb_orange[:,:,1],(3,3),0)
        # 二值化，160 为阈值
        ret,imgbinary = cv2.threshold(Cr_orange,140,255,cv2.THRESH_BINARY)
        cv2.imshow('display',imgbinary)
#       plt.imshow(imgbinary, cmap='gray')
#       plt.axis("off")
#       plt.show()

        color = imgbinary[100] # 单看第 400 行的像素值
        black_count = np.sum(color == 255) # 找到橙色的像素点个数
        orange_index = np.where(color == 255) # 找到橙色的像素点索引

        # 防止 black_count=0 的报错
        if black_count == 0:
            black_count = 1

        # 找到橙色像素的中心点位置
        center = (orange_index[0][black_count - 1] + orange_index[0][0]) / 2

        # 计算出该橙色像素的中心点位置与标准中心点的偏移量
        direction = center - 320

        print(direction)
        if count == 1:
            cmd = input() # 使小车在输入口令后能立即开始行驶

        # 摄像头无法捕捉到橙色线时小车停止
        if abs(direction) > 319:
            pwma.ChangeDutyCycle(0)
            pwmb.ChangeDutyCycle(0)

        elif abs(direction) < 90:
            pwma.ChangeDutyCycle(45)
            pwmb.ChangeDutyCycle(40)

        # 偏移量非负时小车右转
```

```
        elif direction >= 90:
        # 限制偏移量在 70 以内
            # if direction > 70:
                #direction = 70
            pwma.ChangeDutyCycle(34)
            pwmb.ChangeDutyCycle(46)

        # 偏移量为负时小车左转
        elif direction < -90:
            #if direction < -70:
                #direction = -70
            pwma.ChangeDutyCycle(46)
            pwmb.ChangeDutyCycle(34)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
# 释放清理
cap.release()
cv2.destroyAllWindows()
pwma.stop()
pwmb.stop()
GPIO.cleanup()
```

该版本用于代码结构测试。经过多次测试，小车均能够正常通过终点，但是速度较为缓慢，大约在 13 秒左右。

版本二：

```
import RPi.GPIO as GPIO
import time
import cv2
import numpy as np
import matplotlib.pyplot as plt

EA, I2, I1, EB, I4, I3 = (13, 19, 26, 16, 20, 21) # 定义引脚
FREQUENCY = 50 # PWM 波的频率为 50Hz
# 初始化 GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup([EA, I2, I1, EB, I4, I3], GPIO.OUT)
GPIO.output([EA, I2, EB, I3], GPIO.LOW)
GPIO.output([I1, I4], GPIO.HIGH)
```

```
center = 320 #  定义图像的标准中点 center
#  打开摄像头，图像尺寸 640*480（长*高），opencv 存储值为 480*640（行*列）
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

#  初始化 PWM
pwma = GPIO.PWM(EA, FREQUENCY)
pwmb = GPIO.PWM(EB, FREQUENCY)
pwma.start(0) #  此时 ENA 引脚持续产生占空比为 0 的 PWM 输出
pwmb.start(0) #  此时 ENB 引脚持续产生占空比为 0 的 PWM 输出

count = 0
while 1:
    count += 1
    ret, frame = cap.read() #  摄像头读取图像

    # RGB 图像转换到 YCrCb 空间
    YCrCb_orange = cv2.cvtColor(frame,cv2.COLOR_BGR2YCrCb)
    #  提取 Cr 通道并高斯去噪，（3,3）表示高斯核大小
    Cr_orange = cv2.GaussianBlur(YCrCb_orange[:,:,1],(3,3),0)
    #  二值化，160 为阈值
    ret,imgbinary = cv2.threshold(Cr_orange,140,255,cv2.THRESH_BINARY)
    cv2.imshow('display',imgbinary)
#       plt.imshow(imgbinary, cmap='gray')
#       plt.axis("off")
#       plt.show()

    color = imgbinary[70] #  单看第 70 行的像素值
    black_count = np.sum(color == 255) #  找到橙色的像素点个数
    orange_index = np.where(color == 255) #  找到橙色的像素点索引

    #  防止 black_count=0 的报错
    if black_count == 0:
        black_count = 1

    #  找到橙色像素的中心点位置
    center = (orange_index[0][black_count - 1] + orange_index[0][0]) / 2
```

```
        # 计算出该橙色像素的中心点位置与标准中心点的偏移量
        direction = center - 320

        print(direction)
        if count == 1:
            cmd = input() # 使小车在输入口令后能立即开始行驶

        # 摄像头无法捕捉到橙色线时小车停止
        if abs(direction) > 319:
            pwma.ChangeDutyCycle(0)
            pwmb.ChangeDutyCycle(0)

        elif abs(direction) < 90:
            pwma.ChangeDutyCycle(80)
            pwmb.ChangeDutyCycle(75)

        # 偏移量非负时小车右转
        elif direction >= 90:
        # 限制偏移量在 70 以内
            # if direction > 70:
                #direction = 70
            pwma.ChangeDutyCycle(42)
            pwmb.ChangeDutyCycle(56)

        # 偏移量为负时小车左转
        elif direction < -90:
            #if direction < -70:
                #direction = -70
            pwma.ChangeDutyCycle(66)
            pwmb.ChangeDutyCycle(44)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
# 释放清理
cap.release()
cv2.destroyAllWindows()
pwma.stop()
pwmb.stop()
```

GPIO.cleanup()

相较于第一版代码，修改有以下两点

（1）color = imgbinary[70] # 单看第 70 行的像素值 修改了获取的像素值的行数，使得小车可以“看到”更远的巡线从而提早做出判断

（2）分别修改转弯、直行的占空比从而提高小车的运行速度

版本三：

```
import RPi.GPIO as GPIO
import time
import cv2
import numpy as np
import matplotlib.pyplot as plt

EA, I2, I1, EB, I4, I3 = (13, 19, 26, 16, 20, 21) # 定义引脚
FREQUENCY = 50 # PWM 波的频率为 50Hz
# 初始化 GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup([EA, I2, I1, EB, I4, I3], GPIO.OUT)
GPIO.output([EA, I2, EB, I3], GPIO.LOW)
GPIO.output([I1, I4], GPIO.HIGH)

center = 320 # 定义图像的标准中点 center
# 打开摄像头，图像尺寸 640*480（长*高），opencv 存储值为 480*640（行*列）
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

# 初始化 PWM
pwma = GPIO.PWM(EA, FREQUENCY)
pwmb = GPIO.PWM(EB, FREQUENCY)
pwma.start(0) # 此时 ENA 引脚持续产生占空比为 0 的 PWM 输出
pwmb.start(0) # 此时 ENB 引脚持续产生占空比为 0 的 PWM 输出

count = 0
while 1:
    count += 1
    ret, frame = cap.read() # 摄像头读取图像

    # RGB 图像转换到 YCrCb 空间
```

```
        YCrCb_orange = cv2.cvtColor(frame,cv2.COLOR_BGR2YCrCb)
        # 提取 Cr 通道并高斯去噪，（3,3）表示高斯核大小
        Cr_orange = cv2.GaussianBlur(YCrCb_orange[:,:,1],(3,3),0)
        # 二值化，160 为阈值
        ret,imgbinary = cv2.threshold(Cr_orange,140,255,cv2.THRESH_BINARY)
        cv2.imshow('display',imgbinary)
#         plt.imshow(imgbinary, cmap='gray')
#         plt.axis("off")
#         plt.show()

        color = imgbinary[70] # 单看第 400 行的像素值
        black_count = np.sum(color == 255) # 找到橙色的像素点个数
        orange_index = np.where(color == 255) # 找到橙色的像素点索引

        # 防止 black_count=0 的报错
        if black_count == 0:
            black_count = 1

        # 找到橙色像素的中心点位置
        center = (orange_index[0][black_count - 1] + orange_index[0][0]) / 2

        # 计算出该橙色像素的中心点位置与标准中心点的偏移量
        direction = center - 320

        print(direction)
        if count == 1:
            cmd = input() # 使小车在输入口令后能立即开始行驶

        # 摄像头无法捕捉到橙色线时小车停止
        if abs(direction) > 319:
            pwma.ChangeDutyCycle(0)
            pwmb.ChangeDutyCycle(0)

        elif abs(direction) < 70:
            pwma.ChangeDutyCycle(96)
            pwmb.ChangeDutyCycle(90)

        # 偏移量非负时小车右转
        elif direction >= 70:
```

```
    # 限制偏移量在 70 以内
        # if direction > 70:
            #direction = 70
        pwma.ChangeDutyCycle(55)
        pwmb.ChangeDutyCycle(70)

    # 偏移量为负时小车左转
    elif direction < -70:
        #if direction < -70:
            #direction = -70
        pwma.ChangeDutyCycle(76)
        pwmb.ChangeDutyCycle(51)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    # 释放清理
cap.release()
cv2.destroyAllWindows()
pwma.stop()
pwmb.stop()
GPIO.cleanup()
```

改动以下两点：
（1）速度提高到小车的极限速度
（2）direction 判定调小，使得小车对于转弯更加灵敏（但同时更容易在直行时震荡）

代码有明显的局限性：当小车摄像头扫描不到直线时程序报错，小车将会按照最后的状态运行至结束，因此容错率很低，在版本三的调试中也经常难以抵达终点。
值得注意的是，小车的运行温度，电池电量和轨道的平坦程度都会影响最终的结果，因此每一次实验都需要控制变量。

五、 总结与思考：

本实验要求我们结合软硬件控制小车，需要通过对代码和硬件的不断调整来优化小车巡线行驶的效率，这加深了我们对图像处理、占空比调节等原理的理解，并且考验了我们在理论的基础上解决实际问题的能力以及团队协作的能力。