期末考核 【8 字绕桩】

指导教师： 王海鹏

学生姓名： 蔡亦扬 学生姓名： 周训哲 学生姓名： 孔恩燊

学号：23307130258 学号：20307110315 学号：23307130021

专业：技术科学试验班 专业：计算机 专业：技术科学试验班

日 期： 2024.6.7

一、 考核内容：

在赛道中间随机设置障碍物，毯子上会提供地面引导线，可以通过超声、摄像头等传感器完成8字绕桩前行（不一定需要严格循线走，但绕行方向必须一致）。

考核的一些注意事项：

1. 正式竞速前有且仅有一次试跑机会，试跑不计入成绩。
2. 1分钟内计完成多少圈，过半圈点不满一圈按半圈记。
3. 每组3次机会，取最好成绩。
4. 中途如线路错误，即时结束本次测试。
5. 按照成绩排序（允许并列），按照第一名，按每降一名降0.5分计分。
6. 本次考核顺序由第1组开始，依次到第20组。
7. 如果当前考核小组试跑后觉得结果不理想，需要调整小车，可以申请往后顺延2组，去别处场地迅速调试好小车。当正式考核开始时，就不允许去别处场地调试了，需在考核场地完成整个考核。
8. 最后一点，大家考核当天调试时，关注一下充电宝电量，不要出现考核时充电宝没电的尴尬情况。

二、 实验原理：

1. 视觉模块：

（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。

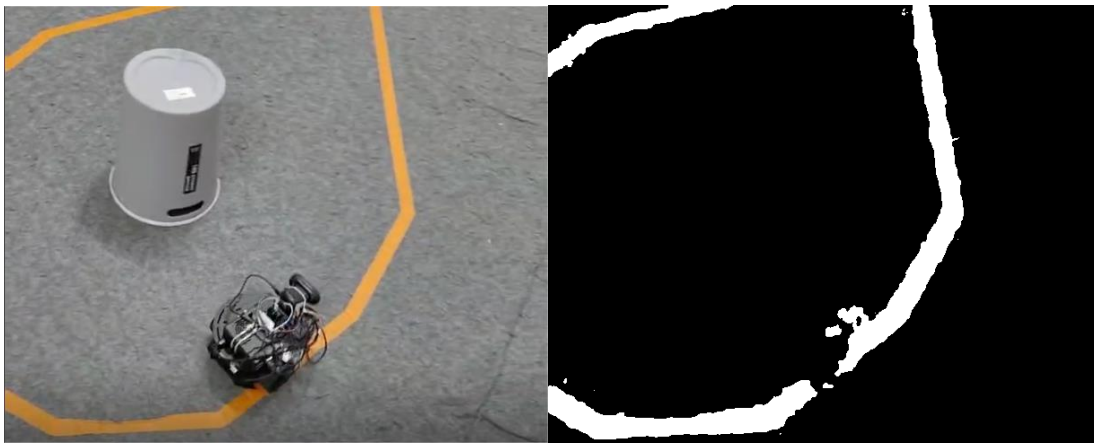
实验中对图像进行了多次处理：

a）YCrCb_orange = cv2.cvtColor(frame, cv2.COLOR_BGR2YCrCb) 将捕捉到的彩色图像从 BGR 色彩空间转换为 YCrCb 色彩空间。Y 表示亮度分量，Cr 和 Cb 表示颜色分量。

b） Cr_orange = cv2.GaussianBlur(YCrCb_orange[:,:,1],(3,3),0) 从转换后的 YCrCb 图像中提取 Cr 通道，并对其应用高斯模糊，以减少噪声。(3,3)表示高斯核的大小。

c）ret,dst = cv2.threshold(Cr_orange,130,255,cv2.THRESH_BINARY) 对模糊处理后的 Cr 通道图像进行二值化处理。阈值设为 130，超过阈值的像素值设为 255，低于阈值的像素值设为 0。这一步将图像转换为二值图像，其中白色区域对应于 Cr 通道值较高的区域。

处理前后的图像对比：



2. PID控制循迹模块：

在本次考核中，提供了地面引导线，我们便想到使用 PID 控制摄像头中心位置和前方引导线轨迹中心位置偏差来实现小车循迹的过程。

（一）自动控制

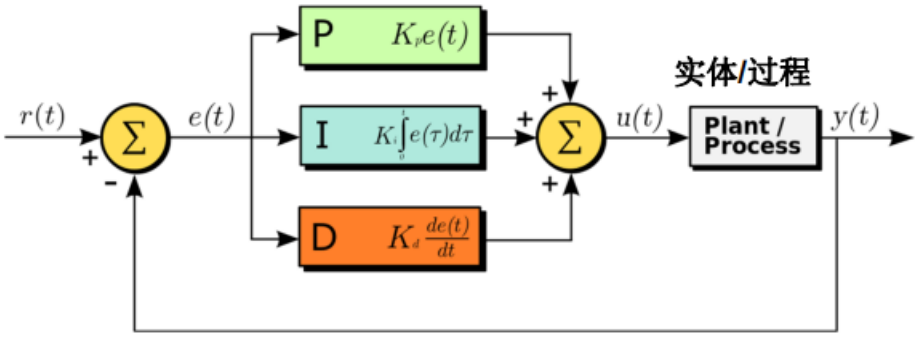
在没有人直接参与的情况下，利用外加的设备或装置，使机器、设备或生产过程的某个工作状态或参数自动地按照预定的规律运行。

（二）PID 控制

（1）简介：

当今的闭环自动控制技术都是基于反馈的概念以减少不确定性。反馈理论的要素包括三个部分：测量、比较和执行。测量关键的是被控变量的实际值，与期望值相比较，用这个偏差来纠正系统的响应，执行调节控制。在工程实际中，应用最为广泛的调节器控制规律为比例(proportion)、积分(integral)、微分(derivative)控制，简称 PID 控制，又称 PID 调节。

（2）控制原理：



如上图所示

r(t)是所需的设定值；

y(t)是测量的过程输出值；

e(t) 即偏差是设定值与过程值的差（r(t)-y(t)）；

u(t) 是经过计算得到的控制变量，输入到控制器中。

P ： proportion （非负比例系数），就是偏差乘以一个常数。

I ： integral （非负积分系数），就是对偏差进行积分运算。

D ： derivative （非负微分系数），对偏差进行微分运算 。

以控制小车为例：将小车速度设定值 r(t)为 3m/s，由光电码盘测得速度为 3.2m/s，即过程输出值 y(t)；偏差 e(t)为-0.2m/s。

P：将-0.2m/s 乘以一个系数（正）输入到控制器中，以减小输出的占空比，则车轮转速将降低，向设定值靠近。KP 越大则调节的灵敏度越大，但过大可能会使实际速度低于 3m/s（超调）。

I：只经过比例调节的小车，可能稳定后的速度为 3.1m/s，存在稳态误差-0.1m/s；虽然误差很小，但是因为积分项也会随着时间的增加而加大，它推动控制器的输出增大，从而使稳态误差进一步减小，直到等于 0。

D：小车中有些组件存在较大惯性或者滞后性，其变化总是落后于误差的变化。假设经比例调节后实际速度为 3.1m/s，则设定速度与实际速度的差值由-0.2m/s 变为-0.1m/s，e(t)的差分为 0.1m/s2，将此差分乘以系数（正）加到控制变量中，相比只有比例环节减缓了速度降低的趋势（减小超调量）。

（3）数学表达：（Kp、Ki、Kd 都是非负的，分别表示比例项，积分项和微分项的系数。）

$$u(t) = K_{\mathrm{p}} e(t) + K_{\mathrm{i}} \int_0^t e(t')\,dt' + K_{\mathrm{d}} \frac{de(t)}{dt}$$

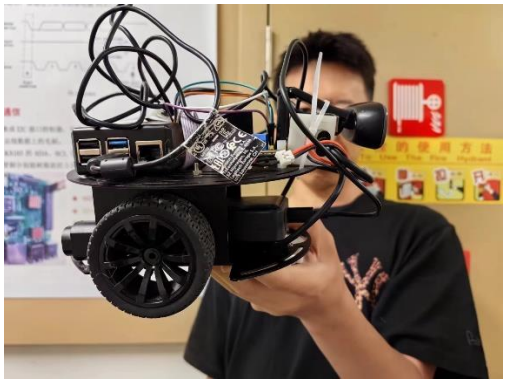
三、 实验内容：

1. 方案设计：

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. 硬件设计：

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，持续执行上一步的操作，可能会导致浪费较多的时间绕弧度更大的圆。所以我们尝试将摄像头架高，以获得更大的视野。同时，我们还尝试使用不同的摄像头角度，发现抬高摄像头会获得更好的视野，小车会绕更小的圆，但是有不稳定的情况出现，容易走出引导线。最终调节摄像头角度和位置如下：



3. 参数调节：

本次实验中主要调节的参数为 pid 和占空比，以及在左转和右转时使用的左右轮差速设置 control。在实验一开始，我们首先尝试了较小的占空比以达到一个合理的绕桩结果，后续我们不断尝试调高占空比，同时增大 p 值使小车更快达到一个巡线的稳态。我们还调节了 i 和 d 值，使得小车在巡线直行过程中减小波动，达到更短的运行时间。我们尝试调节 control：过大会导致小车直行时晃动严重，过小则会导致转弯时无法行进更小的弯实现更短的运行时间。于是我们加入判断条件，使用引导线与摄像头中心的偏差作为判断条件，调节不同情况下的 control。

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. 代码展示：

```
# coding:utf-8
import RPi.GPIO as GPIO
import time
import cv2
import numpy as np
#前期准备部分

#限制较大值
def sign(x):
    if x > 0:
        return 1.0
    else:
        return -1.0

# 定义引脚
EA, I2, I1, EB, I4, I3 = (13, 19, 26, 16, 20, 21)
FREQUENCY = 50

# 设置 GPIO 口
GPIO.setmode(GPIO.BCM)

```

```
# 设置 GPIO 口为输出
GPIO.setup([EA, I2, I1, EB, I4, I3], GPIO.OUT)
GPIO.output([EA, I2, EB, I3], GPIO.LOW)
GPIO.output([I1, I4], GPIO.HIGH)

pwma = GPIO.PWM(EA, FREQUENCY)
pwmb = GPIO.PWM(EB, FREQUENCY)

# pwm 波初始化
pwma.start(0)
pwmb.start(0)

# center 定义
center_now = 320

# 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)

# PID 定义和初始化三个 error 和 adjust 数据
error = [0.0] * 3
adjust = [0.0] * 3

# PID 参数配置、目标值、左右轮基准占空比和占空比偏差范围（根据实际情况调整）
kp = 1.5
ki = 0.4 #yuanlai 0.4
kd = 1.4 #yuanlai 1.4
target = 320
lspeed = 100
rspeed = 100
control = 70
ret, frame = cap.read()

#前期准备完毕，节省时间
print("准备完毕！按下 Enter 启动！")
input()

try:
    while True:
        ret, frame = cap.read()
        # 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)
```

```
65.         # 二值化，130 为阈值
66.         ret,dst = cv2.threshold(Cr_orange,130,255,cv2.THRESH_BINARY)
67.
68.         cv2.imshow("display", dst)
69.
70.         # 单看第 400 行的像素值 s
71.         color = dst[400]
72.         # 找到 black 色的像素点个数
73.         black_count = np.sum(color == 255)
74.
75.         # 防止 black_count=0 的报错
76.         if black_count == 0:
77.             continue
78.         else:
79.             black_index = np.where(color == 255)
80.         # 找到黑色像素的中心点位置
81.         center_now = (black_index[0][black_count - 1] + black_index[0
    ][0]) / 2
82.
83.         # 计算出 center_now 与标准中心点的偏移量
84.         direction = center_now - 320
85.
86.         if direction > 90:
87.             control=85 #yuanlai 85
88.         elif direction < -90:
89.             control=85 #yuanlai 85
90.         else:
91.             control=40 #yuanlai 50
92.
93.         #print("偏差值：", direction)
94.
95.         # 更新 PID 误差
96.         error[0] = error[1]
97.         error[1] = error[2]
98.         error[2] = center_now - target
99.
100.                         # 更新 PID 输出（增量式 PID 表达式）
101.                         adjust[0] = adjust[1]
102.                         adjust[1] = adjust[2]
103.                         adjust[2] = adjust[1] \
104.                                     + kp * (error[2] - error[1]) \
105.                                     + ki * error[2] \
```

```
                                    + kd * (error[2] - 2 * error[1] +
    error[0]);
                        #print(adjust[2])

                        # 饱和输出限制在 control 绝对值之内
                        if abs(adjust[2]) > control:
                            adjust[2] = sign(adjust[2]) * control
                        # print(adjust[2])

                        # 执行 PID

                        # 右转
                        if adjust[2] > 20:
                            #print(f"{lspeed},{rspeed},{adjust}")
                            pwma.ChangeDutyCycle(rspeed - adjust[2])

                            pwmb.ChangeDutyCycle(lspeed)

                        # 左转
                        elif adjust[2] < -20:
                            pwma.ChangeDutyCycle(rspeed)
                            pwmb.ChangeDutyCycle(lspeed + adjust[2] )
    #yuanlaimeiyou

                        else:
                            pwma.ChangeDutyCycle(rspeed)
                            pwmb.ChangeDutyCycle(lspeed)

                        if cv2.waitKey(1) & 0xFF == ord('q'):
                            break
                except KeyboardInterrupt:
                    print("结束！")
                    pass
                # 释放清理

                cap.release()
                cv2.destroyAllWindows()
                pwma.stop()
                pwmb.stop()
                GPIO.cleanup()
```

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。

考试结果如下展示：最终计时一分钟 6 圈