1.29 工程日志

一、今日工作

1.破冰活动结构设计，负责“电梯”结构的制作与电机的调试。

2.晚上完成香橙派 Dev Container 的配置。

二、遇到问题

1.mixly 中控制电机时，发现使用 Delay 模块的程序电机均无法转动。

2.配置 Dev Container 时，VScode 报错。

三、问题解决

1.利用读取系统运行时间模块控制电机转动时间。

2.阅读文档发现，香橙派没有链接 C 板时需将.json 文件中相应模块的功能注释掉。

工 程 日 志

组别	进阶营 3 组	日期	2024.1.30
姓名	胡文迪	带组助教	孔德浩
工程和任务完成情况	1. 学习 ros 系统相关知识 2. 学习通信与控制相关内容		
困难与问题	1.在阅读官方给出的程序时遇到困难，通过自学 ros 解决了一部分困惑		
想法			

心得体会	自身还有很多不足与知识缺陷，需要加快速度学习。

工 程 日 志

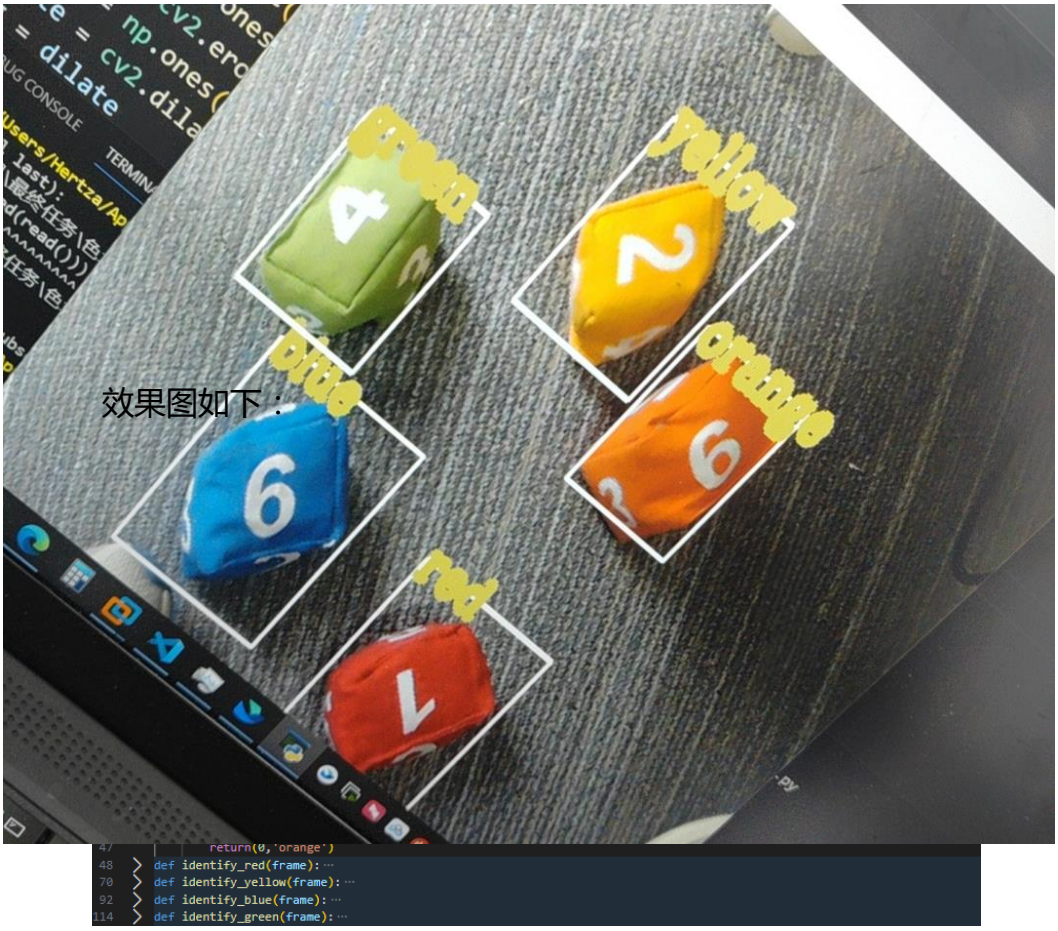
组 别	进阶营 3 组	日 期	2024.1.31
姓 名	胡文迪	带 组 助 教	孔德浩

工程和任务完成情况

1. 运用 Python 与 OpenCv 完成了沙包识别模块的程序，代码如下：

```python
def draw(frame,list_coordinate):
    #return(list_coordinate)
    if list_coordinate[0] != 0:
        x1 = int(list_coordinate[0][0])
        y1 = int(list_coordinate[0][1])
        x2 = int(list_coordinate[0][2])
        y2 = int(list_coordinate[0][3])
        color = str(list_coordinate[1])
    else:
        return(frame)
    if list_coordinate[0] == 0:
        return(frame)
    else:
        img = cv2.rectangle(frame,(x2,y2),(x1,y1),(255,255,255),2)
        cv2.putText(img, color, (x1,y1+10), cv2.FONT_HERSHEY_COMPLEX, 1.0, (100, 200, 200), 5)
        return(img)
while True:
    img = draw(read(),identify_orange(read()))
    img = draw(img,identify_red(read()))
    img = draw(img,identify_yellow(read()))
```

效果图如下：



2. 为该程序编写了一个 Publisher 端口，并在 Docker 中创建了对应的功能包，将程序挂载至功能包中，在功能包中配置程序所需环境。
3. 编写了一个 Describer 程序，用于测试 Publisher 功能是否正常。（由于算法组占用了摄像头资源，2,3 功能尚未测试）
4. 编写了该程序的技术文档

困难与问题	1. 在运用 cv2.boxPoints()函数时，一开始根据其返回的点坐标绘制出的矩形并不准确，经过资料查阅，发现该函数输出的点坐标不一定按顺序对应左上，左下等，需要自己进行排序。 2. 在 Docker 中配置 Numpy 环境时，一开始出现报错，无法找到 Numpy 库。经过反复试验，最后卸载，重启 Docker，再安装 Numpy,成功解决问题。
想法	今天上午编写的 OpenCv 程序，对我来说相对简单，后面编写的 Publisher 与 Subscriber,对我来说属于相对有挑战性的任务。通过在 B 站上学习相应的课程，帮助我完成了对应的任务，是一件很有成就感的事情。
心得体会	自身对于 Docker 的运用还有很多不足与知识缺陷，需要加快速度学习。

工 程 日 志

组 别	进阶营 3 组	日 期	2024.2.1
姓 名	胡文迪	带 组 助 教	孔德浩

工程和任务完成情况

5. 测试了 Identify_Publisher 程序，功能函数截图如下：

```python
def talker():
    rospy.init_node('color', anonymous=True)
    pub = rospy.Publisher('chatter', Int16MultiArray, queue_size=10)

    rate = rospy.Rate(10) # 10hz
    while not rospy.is_shutdown():
        list_color = []
        orange = identify_orange(read())
        list_color.append(orange)
        red = identify_red(read())
        list_color.append(red)
        blue = identify_blue(read())
        list_color.append(blue)
        yellow = identify_yellow(read())
        list_color.append(yellow)
        green = identify_green(read())
        list_color.append(green)

        rospy.loginfo('start sending')
        msg = Int16MultiArray()
        msg.data = list_color
        pub.publish(msg)
        rate.sleep()
```

Publisher 功能正常。

6. 编写了 PID 控制机器人移动到指定位置的程序，代码截图如下：

```python
import rospy
from geometry_msgs.msg import Twist
from geometry_msgs.msg import PoseStamped
import math
class CmdVelPublisher:
    def __init__(self):
        self.cmd_vel_pub = rospy.Publisher('cmd_vel', Twist, queue_size=10)

    def publish_velocity(self, linear_x, linear_y, angular_z):
        twist_msg = Twist()
        twist_msg.linear.x = linear_x
        twist_msg.linear.y = linear_y
        twist_msg.angular.z = angular_z

        # 发布消息
        self.cmd_vel_pub.publish(twist_msg)

class ItemPoseSubscriber:
    def __init__(self):
        # 初始化ROS节点
        rospy.init_node('item_pose_subscriber', anonymous=True)

        # 订阅/item_pose话题
```

```
        # 打印位置和姿态信息
        #print(f"Position: x={position.x}, y={position.y}, z={position.z}")
        #print(f"Orientation (Quaternion): x={orientation.x}, y={orientation.y}, z={orientat
        return position.x, position.y
        # 如果需要，还可以进行进一步的数据处理或控制操作

class PIDController:
    def __init__(self, kP, kI, kD):
        self.kP = kP  # 比例增益
        self.kI = kI  # 积分增益
        self.kD = kD  # 微分增益

        self.last_error = [0.0, 0.0]  # 上一次的位置误差
        self.integral_error = [0.0, 0.0]  # 积分位置误差
        self.set_point = [0.0, 0.0]  # 目标位置

    def set_target(self, target_x, target_y):
        self.set_point = [target_x, target_y]

    def update(self, current_x, current_y):
        error = [self.set_point[0] - current_x, self.set_point[1] - current_y]  # 计算当前位

        # 积分项更新
        self.integral_error[0] += error[0]
        self.integral_error[1] += error[1]

        # 微分项计算（这里采用一阶差分近似微分）
        derivative_error = [error[0] - self.last_error[0], error[1] - self.last_error[1]]

        # 计算PID输出
        output = [
            self.kP * error[0] + self.kI * self.integral_error[0] + self.kD * derivative_error[0],
            self.kP * error[1] + self.kI * self.integral_error[1] + self.kD * derivative_error[1],
        ]

        # 更新上一次错误值
        self.last_error = error

        # 返回速度向量（假设速度限制在(-1, 1)之间）
        return [max(min(output[0], 1), -1), max(min(output[1], 1), -1)]

# 使用示例
pid_controller = PIDController(kP=0.5, kI=0.1, kD=0.05)
pid_controller.set_target(10.0, 20.0)

current_position = [0.0, 0.0]
while True:
    while not rospy.is_shutdown():
        # 获取或模拟当前坐标
        current_position = ItemPoseSubscriber()  # 这里应该替换为获取真实当前坐标的函数

        # 计算并输出速度
        velocity = pid_controller.update(current_position[0], current_position[1])
        print(f"Velocity: {velocity}")

        # 在这里执行将速度发送到驱动器或其他控制设备的操作

        # 延时以模拟周期性控制循环

        rospy.init_node('cmd_vel_publisher', anonymous=True)

        pub = CmdVelPublisher()

        rate = rospy.Rate(10)  # 设置发布频率为每秒10次
        # 假设这是你要发送的线性速度和角速度
        linear_speed = velocity  # 单位: m/s
        angular_speed = 0  # 单位: rad/s

        pub.publish_velocity(linear_speed, 0.0, angular_speed)  # 直线移动时，通常y轴速度为0

        rate.sleep()
```

	由于车被拉去测舵机了，该程序尚未上机实测。
困难与问题	1. 测试 publish 节点时，出现报错，经查询，发现时节点未初始化。 2. 阅读助教写的 PID_Planner 程序时，发现其只有比例系数 Kp，而没有积分与微分系数，所以重新写了一个。
想法	上午测试完 Publisher 节点后，因为看不懂助教写的程序，有一段时间无事可做。后来下午痛定思痛，跟助教沟通了一下程序的相关问题，着手进行了程序的优化。
心得体会	我们应主动踏出自己的舒适圈。

工 程 日 志

组别	进阶营 3 组	日期	2024.2.2
姓名	胡文迪	带 组 助 教	孔德浩

工程和任务完成情况

3. 完成沙包对位程序的调试（未使用 pid）,效果已发至飞书群。
4. 修改 pid_test 程序，添加了接收目标坐标的节点，函数截图如下：

```python
class PositionTargetSubscriber:
    def __init__(self, pid_controller):
        self.pid_controller = pid_controller
        rospy.init_node('position_target_subscriber', anonymous=True)

        # 订阅/position_target话题并传递回调函数
        self.position_target_sub = rospy.Subscriber('/position_target', Float64MultiArray, self.position_callback)

    def position_callback(self, msg):
        """
        当接收到新的/position_target消息时调用此回调函数
        """
        if len(msg.data) >= 2:
            position_x = msg.data[0]
            position_y = msg.data[1]
            self.pid_controller.set_target(position_x, position_y)
```

5. 自己编写了一个 pid 对位沙包程序，并编写了其技术文档，程序截图如下：

```python
import rospy
from std_msgs.msg import Int16MultiArray
import time
from geometry_msgs.msg import Twist
# 创建一个全局变量来保存接收到的消息
received_message = None

def chatter_callback(data):
    global received_message
    received_message = data.data.copy()  # 将接收到的消息保存到全局变量中


class PIDController:
    def __init__(self, kp=0.005, ki=0.0, kd=0.0, output_limits=(-1, 1), integral_limit=None):
        self.kp = kp
        self.ki = ki
        self.kd = kd
        self.output_limits = output_limits
        self.integral_limit = integral_limit or float('inf')
        self.setpoint = 450
        self.previous_error = 0
        self.integral = 0
        self.last_time = time.time()

    def update(self, current_value):
        global received_message
        error = self.setpoint - received_message

        current_time = time.time()
        dt = current_time - self.last_time
        self.last_time = current_time
```

```python
def publish_linear_speed(speed):
    # 创建Twist消息对象
    twist_msg = Twist()

    # 设置x方向的速度值
    twist_msg.linear.x = speed

    # 其他方向的速度和角速度设置为0
    twist_msg.linear.y = 0.0
    twist_msg.linear.z = 0.0
    twist_msg.angular.x = 0.0
    twist_msg.angular.y = 0.0
    twist_msg.angular.z = 0.0

    # 发布cmd_vel话题
    pub = rospy.Publisher('/cmd_vel', Twist, queue_size=10)
    pub.publish(twist_msg)


# 初始化ROS节点
rospy.init_node('chatter_subscriber', anonymous=True)

#创建Subscriber，订阅名为chatter的话题，并指定回调函数为chatter_callback
subscriber = rospy.Subscriber('/chatter', Int16MultiArray, chatter_callback)
# 初始化PID控制器实例
pid = PIDController(kp=0.002, ki=0, kd=0.002)
```

```python
#创建Subscriber，订阅名为chatter的话题，并指定回调函数为chatter_callback
subscriber = rospy.Subscriber('/chatter', Int16MultiArray, chatter_callback)
# 初始化PID控制器实例
pid = PIDController(kp=0.002, ki=0, kd=0.002)
# 运行其他代码或循环以处理received_message变量中的消息
while not rospy.is_shutdown():
    if received_message is not None:
        while True:
            # 更新received_message的值（这里假设从某个传感器读取）
            subscriber = rospy.Subscriber('/chatter', Int16MultiArray, chatter_callback)

            # 调整电机速度
            speed = pid.update(received_message)
            publish_linear_speed(speed)  # 请替换为实际设置速度的函数或操作

            # 等待下一个采样周期
            time.sleep(0.01)  # 根据实际情况调整采样周期

    # 维持节点运行并检查新的消息
    rospy.sleep(0.1)
```

```python
            self.integral = self.integral_limit if self.integral > 0 else -self.integral_limit
        i_term = self.ki * self.integral

        # 微分项
        d_term = 0 if dt == 0 else self.kd * (error - self.previous_error) / dt
        self.previous_error = error

        # 计算PID输出并限制在指定范围内
        output = p_term + i_term + d_term
        output = max(min(output, self.output_limits[1]), self.output_limits[0])

        return output


# 假设你已经初始化了节点
rospy.init_node('I_am_homo')
```

	6. 实机运行，调试 pid_test 程序 7. 与组员协作，调试运动规划程序，通过向 click 节点发布坐标信息，实现移动到指定坐标功能。
困难与问题	1. 由于网络环境问题，当多人同时连接到香橙派的 Docker 时，会出现连接失 败的问题，和助教沟通之后，我们创建了一个 Gitee 仓库，并创建了仓库与 Docker 的通道，使我们在写程序时，直接在本机环境上开发，开发完上传至仓库，再在 Docker 中更新即可。 2. 运行 pid_test 程序时，出现“import”command not found 问题。经过查询，发现需要再程序前加入 Python 解释器的路径，例如“#！use/bin/python3”（尤其要注意 Python3 前的斜杠，非常重要）。 3. 运行 pid_test 程序时，出现节点订阅失败问题。通过与 AI 的沟通，成功解决问题。
想法	组内出现了一些“抢车”现象，即每个人都写了功能包需要上机调试，但是机子只有一台，Docker 也不支持多人同时连接（网络环境问题），所以出现了功能包很多，但是整机功能并不完善的问题。
心得体会	组员之间一定要增进沟通，以推动整组进度。

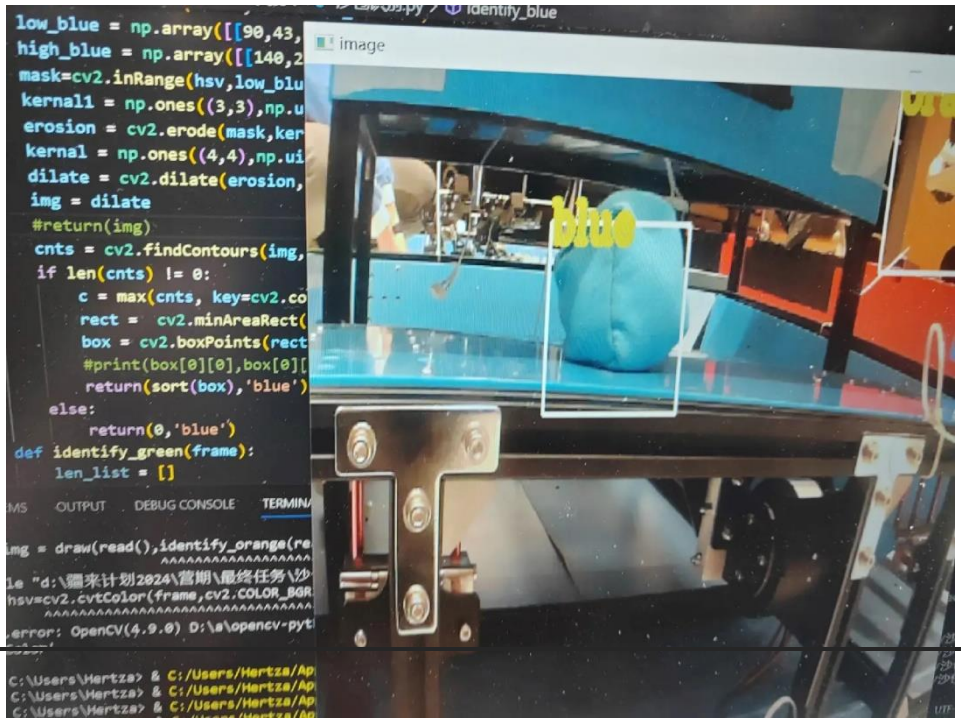
工 程 日 志

组别	进阶营 3 组	日 期	2024.2.3
姓名	胡文迪	带 组 助 教	孔德浩

工程和任务完成情况

1. 试图解决在蓝色板子上识别蓝色沙包问题，方法是在识别蓝色时多腐蚀几次，效果如下：

```
mask=cv2.inRange(hsv,low_blue,high_blue)  #进行掩模运算
kernal1 = np.ones((3,3),np.uint8)
erosion = cv2.erode(mask,kernal1,iterations=15)
kernal = np.ones((4,4),np.uint8)
dilate = cv2.dilate(erosion,kernal,iterations=1)
img = dilate
#return(img)
cnts = cv2.findContours(img, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)[-2]
if len(cnts) != 0:
```



	但是该方案只能在摄像头平视沙包的时候使用，故被淘汰。 2. 重写了 pid 控制运动程序，使其更加简洁精练。 3. 重写了 pid 控制运动程序的技术文档，使其适配新程序。
困难与问题	1.机器时不时会出现识别不到 C 板的现象，调试发现，其在有的电脑上可以识别，有的电脑无法识别。最后更换数据线，解决问题。
想法	
心得体会	