北邮第一生禽队报告材料

队员：萧鹏飞、张登博、曹怀允、徐新

指导老师：庄育锋、李海源

1. 作品概述

1.1 作品背景与目的

随着科技的快速发展，四足机器人在物资运送领域的应用逐渐受到重视。本参赛作品旨在通过设计一款具备高效物资识别、精准定位与稳定运输能力的四足机器人，以宇树科技 go1 为基础平台，结合 Arduino Uno 开发板和机械臂装载装置，参与四足机器人物资运送比赛，展示现代机器人技术在物资运输领域的应用与创新。

1.2 整体技术方案

1.2.1 识别与定位能力：

视觉算法：采用 OpenCV 图像处理库，实现道路、物资及路段标识的准确识别。

编程环境:基于 Linux 系统,进行软件开发与算法集成,确保系统的稳定性与可扩展性。

场景识别策略：

道路识别：通过颜色空间转换、颜色过滤、形态学操作及扫线法提取道路中线，实现机器人的实时定位与导航。

物资识别与编号提取：利用颜色差异进行物资分割，提取物资特征并进行编号识别，实现物资的精准识别与定位。

任务分配策略：根据物资编号和倾倒区域位置，优化物资卸载流程，提高运输效率。

1.2.2 算法优化能力：

计算资源优化：通过模型压缩和计算量减少，实现高效的图像处理和目标检测。

运动控制优化：采用 PID 控制算法，实现四足机器人的精准运动控制。

1.3 比赛流程规划

赛前准备：进行机器人调试与测试，确保各项功能正常；检查装载装置，验证其稳定性和承重能力。

比赛过程：机器人启动后，通过视觉识别系统装载指定物资；沿黄色道路指引行走，将物资卸载至对应编号的倾倒区；完成所有任务后，准确返回启停区并示意裁判结束比赛。

附加赛准备：设计并测试机械抓取装置，确保其与机器人协同工作，稳定抓取与卸载物资。

1.4 预期成果与挑战

本作品预期能够成功完成所有比赛任务，在有限时间内高效完成物资运送。然而，我们也面临视觉识别准确性、抗干扰能力以及机械抓取装置稳定性等挑战。

1.5 总结

本参赛作品以四足机器人物资运送挑战赛为契机，展示了现代机器人技术在物资运输领域的创新应用。通过精心设计的识别与定位能力、场景识别策略、任务分配策略以及算法优

化能力，本作品力求在比赛中脱颖而出，为四足机器人在物资运输领域的发展贡献自己的力量。

2. 比赛程序

2.1 运动控制程序 run_x.py

```
#!/usr/bin/python3
import sys
import time
import math
import socket
import serial
sys.path.append('../lib/python/arm64')
import robot_interface as sdk

d_aerror = 0.0
d_derror = 0.0

# 字符串类型转浮点类型
def s_t_f(s):
    return float(s)

# 得到目标偏转弧度，指导偏航，采用 PID 算法控制
def tg_angle(a_error, last_aerror):
    global d_aerror
    d_aerror = a_error - last_aerror
    last_aerror = a_error
    if abs(a_error) > 0.5:
        target = 2.2 * a_error + 0.1 * d_aerror
    else:
        target = 1.1 * a_error + 0.5 * d_aerror
    return last_aerror, target

# 得到目标偏移距离，指导侧向速度
def tg_distance(d_error, last_derror):
    global d_derror
    d_derror = d_error - last_derror
    last_derror = d_error
    if abs(d_error) < 0.08:
        target = 1.1 * d_error + 2.1 * d_derror
    else:
        target = 1.5 * d_error + 0.4 * d_derror
    return last_derror, target
def tg_action(target_angle, target_distance, velocity):
```

```
    action_angle = target_angle
    action_angle = round(action_angle, 2)
    if abs(action_angle) > 2:
        if action_angle > 0:
            action_angle = 2
        else:
            action_angle = -2
    action_distance = target_distance
    action_distance = round(action_distance, 2)
    if abs(action_distance) > 0.4:
        if action_distance > 0:
            action_distance = 0.4
        else:
            action_distance = -0.4

# velocity 前方视觉空间指导前进速度
    if velocity < 10:
        action_velocity = 0
    elif velocity < 30:
        action_velocity = 0.013 * velocity - 0.15
    elif velocity < 40:
        action_velocity = 0.013 * velocity - 0.15
    elif velocity < 70:
        action_velocity = 0.011 * velocity - 0.07
    elif velocity < 80:
        action_velocity = 0.01 * velocity
    elif velocity < 100:
        action_velocity = 0.01 * velocity
    else:
        action_velocity = 0.01
    tmpk = (abs(d_aerror) + 1) * (abs(d_derror) + 1)
    action_velocity = round((action_velocity / tmpk), 2)
    return action_angle, action_distance, action_velocity

#  分割数据函数
def get_data(data):
    try:
        li = data.split("/")
        a = li[0]
        d = li[1]
        v = li[2]
        s = li[3]
        z = li[4]
        r = li[5]
```

```
        put1 = li[6]
        put2 = li[7]
    except:
        a = "1.57"
        d = "-5"
        v = "0"
        s = "0"
        z = "0"
        r = "0"
        put1 = "0"
        put2 = "0"
    try:
        angle = s_t_f(a)
    except:
        angle = 1.57
    try:
        distance = s_t_f(d)
    except:
        distance = -5
    try:
        velocity = s_t_f(v)
    except:
        velocity = 0
    try:
        signal = s_t_f(s[0])
    except:
        signal = 0
    try:
        bz = s_t_f(z[0])
    except:
        bz = 0
    try:
        ruku = s_t_f(r[0])
    except:
        ruku = 0
    try:
        put1 = int(put1)
    except:
        put1 = 0
    try:
        put2 = int(put2)
    except:
        put2 = 0
    return angle, distance, velocity, signal, bz, ruku, put1, put2
```

```
# 一步动作，共 0.5 秒
def motion1(cmd, udp, state, forward=0, side=0, yaw=0):
    t = 0
    while t < 251:
        t = t + 1
        time.sleep(0.002)
        udp.Recv()
        udp.GetRecv(state)
        cmd.mode = 0
        cmd.gaitType = 0
        cmd.speedLevel = 0
        cmd.footRaiseHeight = 0
        cmd.bodyHeight = 0
        cmd.euler = [0, 0, 0]
        cmd.velocity = [0, 0]
        cmd.yawSpeed = 0.0
        cmd.reserve = 0
        if t < 250:
            cmd.mode = 2
            cmd.gaitType = 1
            cmd.yawSpeed=yaw
            #time.sleep(0.002)
            cmd.velocity = [forward, side]
            #cmd.yawSpeed = yaw
        udp.SetSend(cmd)
        udp.Send()

# 站立函数
def stand(cmd, udp, state, forward=0, side=0, yaw=0):
    t = 0
    while t < 226:
        t = t + 1
        time.sleep(0.002)
        udp.Recv()
        udp.GetRecv(state)
        cmd.mode = 0
        cmd.gaitType = 0
        cmd.speedLevel = 0
        cmd.footRaiseHeight = 0
        cmd.bodyHeight = 0
        cmd.euler = [0, 0, 0]
        cmd.velocity = [0, 0]
        cmd.yawSpeed = 0.0
```

```
        cmd.reserve = 0
        if t < 75:
            cmd.mode = 5
            cmd.yawSpeed = yaw
            cmd.velocity = [forward, side]
        elif t < 150:
            cmd.mode = 6
            cmd.yawSpeed = yaw
            cmd.velocity = [forward, side]
        elif t < 225:
            cmd.mode = 1
            cmd.yawSpeed = yaw
            cmd.velocity = [forward, side]
        udp.SetSend(cmd)
        udp.Send()

# 下蹲函数
def sit(cmd, udp, state, forward=0, side=0, yaw=0):
    t = 0
    while t < 1001:
        t = t + 1
        time.sleep(0.002)
        udp.Recv()
        udp.GetRecv(state)
        cmd.mode = 0
        cmd.gaitType = 0
        cmd.speedLevel = 0
        cmd.footRaiseHeight = 0
        cmd.bodyHeight = 0
        cmd.euler = [0, 0, 0]
        cmd.velocity = [0, 0]
        cmd.yawSpeed = 0.0
        cmd.reserve = 0
        if t < 250:
            cmd.mode = 2
            cmd.yawSpeed = yaw
            cmd.velocity = [forward, side]
        elif t < 500:
            cmd.mode = 1
            cmd.yawSpeed = yaw
            cmd.velocity = [forward, side]
        elif t < 750:
            cmd.mode = 6
            cmd.yawSpeed = yaw
```

```
            cmd.velocity = [forward, side]
        elif t < 1000:
            cmd.mode = 5
            cmd.yawSpeed = yaw
            cmd.velocity = [forward, side]
        udp.SetSend(cmd)
        udp.Send()

# 两步动作，共 1 秒
def motion2(cmd, udp, state, forward1=0, side1=0, yaw1=0, forward2=0, side2=0, yaw2=0):
    motion1(cmd, udp, state, forward1, side1, yaw1)
    motion1(cmd, udp, state, forward2, side2, yaw2)

# 主函数
if __name__ == '__main__':
    HIGHLEVEL = 0xee
    LOWLEVEL = 0xff

    # 建立与运动控制模块 udp 通信
    udp = sdk.UDP(HIGHLEVEL, 8080, "192.168.123.161", 8082)
    cmd = sdk.HighCmd()
    state = sdk.HighState()
    udp.InitCmdData(cmd)

    # 建立与机械臂控制串口通信，建立后开始机械臂抓取
    port = "/dev/ttyUSB0"
    rates = 9600
    ser = serial.Serial(port, rates)
    ser.flushInput()
    print("start")

    # 键盘按任意按键，站立
    user_input1 = input()
    print("stand")
    stand(cmd, udp, state, 0, 0, 0)

    # 键盘按任意按键，控制狗横向移动离开启停区
    user_input2 = input()
    motion1(cmd, udp, state, 0, 0.7, 0)

    # 建立与视觉处理部分 socket 通信
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_ip = "192.168.123.13"
    server_port = 6011
```

```
client_socket.connect((server_ip, server_port))
client_socket.settimeout(1)

# 变量定义
action_angle = 0
action_distance = 0
action_velocity = 0
last_aerror = 0
last_derror = 0
a_error = 0
d_error = 0
velocity = 0
global a
global b
global attack
time1 = 0
time3 = 0
time2 = 0
time4 = 0
time5 = 0
tine6 = 0
count = 0
a = 0
b = 0
flag = 1
attack = 0
time.now = 0
avi_count = 0
time_start = time.time()

# 主程序循环
while True:

    # 接收数据初始化
    data="0/0/0/0/0/0/0/0/"
    data1 = "1/1/"
    time2 = time.time()
    time4 = time2
    if a > 1 or b > 1:
        if (time2 - time1) >= 3:
            a = 0
        if (time4-time3) >= 3:
            b = 0
```

```
# 接受数据，做分割
try:
    data = client_socket.recv(1024).decode()
except socket.timeout:
    data = "1.57/-5/0/0/0/0/" + data1
    print('chaoshiiiiiiii')
angle, distance, velocity, signal, bz, ruku, put1, put2 = get_data(data)

# 启动时对投放环岛方向做判断，传递到机械臂控制部分
if flag == 1:
    flag = 0
    if (put1 == 1 and put2 == 2):
        data1 = "4"
        #ser.write(data1.encode('utf-8'))
    if (put1 == 3 and put2 == 4):
        data1 = "5"
        #ser.write(data1.encode('utf-8'))
    else:
        data1 = "6"
        #ser.write(data1.encode('utf-8'))
    print(data1)

# 返回启停区
if ruku == 1 and (time.time() - time_start) > 46:
    print("finish : " + str(time2 - time_start))
    motion1(cmd, udp, state, 1, 0, 0)
    motion2(cmd, udp, state, 0, -0.7, 0, 0, -0.7, 0)
    break

# 特殊直行条件判断，用于出二号倾倒区矫正姿态
if count == 1 and (time.time() - time5) > 4:
    motion1(cmd, udp, state, 0.8, 0.1, 0)
    count = 0
    datax = client_socket.recv(32768).decode()

# 各倾倒区的运动与物资投放部分
if signal == 1:

    # 一号倾倒区
    if attack < 1:
        print("Corner 1 : " + str(time2 - time_start))
        if put1 == 1:
            motion2(cmd, udp, state, 0.9, 0, 0.8, 0.6, 0, 0.5)
            motion1(cmd, udp, state, 0.5, 0, 1.6)
```

```
            sit(cmd, udp, state, 0, 0, 0)
            data1 = "2"
            ser.write(data1.encode('utf-8'))
            print("put 1")
            datax = client_socket.recv(32768).decode()
            time.sleep(3)
            stand(cmd, udp, state, 0, 0, 0)
        else:
            motion2(cmd, udp, state, 0.8, 0, 0, 0.8,0,0)
            print("pass 1")
            datax = client_socket.recv(32768).decode()
        time1 = time.time()
        tim2 = time1
        attack = attack + 1
        continue

# 二号倾倒区
if attack == 1:
        time1 = time.time()
        time2 = time1
        attack += 1
        print("Corner 2 : " + str(time2 - time_start))
        if put1 == 2:
            motion2(cmd, udp, state, 0.8, 0, 0.3, 0.8, 0, 0)
            motion2(cmd, udp, state, 0.2, 0, 1.5,0.5,0,1.1)
            motion1(cmd, udp, state, 0.3, 0 , 0.9)
            sit(cmd, udp, state, 0, 0, 0)
            data1 = "2"
            ser.write(data1.encode('utf-8'))
            print("put 2")
            time.sleep(3)
            stand(cmd, udp ,state, 0, 0, 0)
            count = 1
            time5 =time.time()
            datax = client_socket.recv(32768).decode()
        elif put2 == 2:
            motion2(cmd, udp, state, 0.8, 0, 0.3, 0.8, 0, 0)
            motion2(cmd, udp, state, 0.2, 0, 1.5,0.5,0,1.1) # 1.65
            motion1(cmd, udp, state, 0.3,0,0.9)
            sit(cmd, udp ,state, 0, 0, 0)
            data1 = "3"
            ser.write(data1.encode('utf-8'))
            print("put 2")
            time.sleep(3)
```

```
            stand(cmd, udp ,state, 0, 0, 0)
            count = 1
            time5 = time.time()
            datax = client_socket.recv(32768).decode()
        else:
            motion2(cmd, udp, state, 0.6,0,0,1,0,0)
            print("pass 2")
            datax = client_socket.recv(32768).decode()
        continue

# 三号倾倒区
if attack == 2:
        attack += 1
        print("Corner 3 : " + str(time.time() - time_start))
        if put1 == 3:
            motion1(cmd, udp, state, 1, 0, 0)
            motion1(cmd, udp, state, 0.2, 0, 0)
            sit(cmd, udp ,state, 0, 0, 0)
            data1 = "2"
            ser.write(data1.encode('utf-8'))
            print("put 3")
            datax = client_socket.recv(32768).decode()
            time.sleep(3)
            stand(cmd, udp, state, 0, 0, 0)
        elif put2 == 3:
            motion1(cmd, udp, state, 1, 0, 0)
            motion1(cmd, udp, state, 0.2, 0, 0)
            sit(cmd, udp ,state, 0, 0, 0)
            data1 = "3"
            ser.write(data1.encode('utf-8'))
            print("put 3")
            datax = client_socket.recv(32768).decode()
            time.sleep(3)
            stand(cmd, udp ,state, 0, 0, 0)
        else:
            motion1(cmd, udp, state, 0.5, 0, 0)
            motion2(cmd, udp, state, 0.1, 0, 1.5,0.1,0,1.5)
            print("pass 3")
            datax = client_socket.recv(32768).decode()
        continue

# 四号倾倒区
if attack == 3:
        attack += 1
```

```
            print("Corner 4 : " + str(time.time() - time_start))
            if put2 == 4:
                motion2(cmd, udp, state, 0.8, -0.1, -0.5, 0.9, -0.2, 0)
                sit(cmd, udp ,state, 0, 0, 0)
                data1 = "3"
                ser.write(data1.encode('utf-8'))
                print("put 4")
                datax = client_socket.recv(32768).decode()
                time.sleep(3)
                stand(cmd, udp ,state, 0, 0, 0)
            else:
                motion1(cmd, udp, state, 0.7, 0, 0)
                motion2(cmd, udp, state, 0.1, 0, 1.3,0.1,0,1.3)
                print("pass 4")
                datax = client_socket.recv(32768).decode()
            continue
        else:
            continue

# 避障处理，并恢复避障标志位，避免重复避障
if bz == 1 and avi_count==0:
    print("Avoid : " + str(time2 - time_start))
    motion1(cmd, udp, state, 0, 0.6,0)
    motion2(cmd, udp, state, 1,0,0,0, 0,-1.5)
    motion1(cmd, udp, state, 0.7,0,0)
    datax = client_socket.recv(32768).decode()
    avi_count = 1

# 正常运动部分
else:

    # 数据初步处理
    if angle > 0:
        a_error = angle - 1.57
    else:
        a_error = 1.57 + angle
    a_error += 0.08
    if abs(a_error) > 2:
        continue
    d_error = (distance + 5) * 0.0022
    last_aerror, target_angle = tg_angle(a_error, last_aerror)
    last_derror, target_distance = tg_distance(d_error, last_derror)

    # 得到动作值
```

```
            action_angle, action_distance, action_velocity = tg_action(target_angle,
target_distance, velocity)
            udp.Recv()
            udp.GetRecv(state)

            # 初始化
            cmd.mode = 0
            cmd.gaitType = 0
            cmd.speedLevel = 0
            cmd.footRaiseHeight = 0
            cmd.bodyHeight = 0
            cmd.euler = [0, 0, 0]
            cmd.velocity = [0, 0]
            cmd.yawSpeed = 0.0
            cmd.reserve = 0

            cmd.mode = 2
            cmd.gaitType = 1
            time2 = time.time()
            if (time1 is not 0) and ((time2 - time1) <= 4.4):
                action_velocity = 0.55
                # print("set velocity")
            cmd.yawSpeed = action_angle
            time.sleep(0.002)
            cmd.velocity = [action_velocity, action_distance]
            udp.SetSend(cmd)
            udp.Send()
            time.sleep(0.002)

# 执行完成，关闭 socket 与串口通信
    client_socket.terminate()
    client_socket.close()
    ser.close()
```

2.2 视觉处理程序 view_x.py

```
import cv2
import numpy as np
import time
import socket

# 读取摄像头
cap = cv2.VideoCapture(1)
cap.set(cv2.CAP_PROP_FPS, 100)
```

```
# hsv 阈值设置
lower_red1 = np.array([0, 70, 100])
upper_red1 = np.array([14, 255, 255])
lower_yellow = np.array([15, 26, 46])
upper_yellow = np.array([55, 255, 255])
lower_green = np.array([56, 91, 60])
upper_green = np.array([80, 255, 255])
lower_blue = np.array([90, 60, 70])
upper_blue = np.array([110, 255, 255])
lower_purple = np.array([111, 40, 70])
upper_purple = np.array([155, 255, 255])
lower_red2 = np.array([156, 70, 100])
upper_red2 = np.array([180, 255, 255])

# 颜色识别函数，找原图指定区域数量最多颜色块并返回值
def color_find(img):
    kernel = np.ones((35, 35), np.uint8)
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    Open = cv2.morphologyEx(hsv, cv2.MORPH_OPEN, kernel)
    hist = cv2.calcHist([Open], [0], None, [180], [0, 180])
    hist_max = np.where(hist == np.max(hist))
    if lower_yellow[0] < hist_max[0] < upper_yellow[0]:
        print('yellow')
        return 1
    elif lower_purple[0] < hist_max[0] < upper_purple[0]:
        print('purple')
        return 2
    elif lower_green[0] < hist_max[0] < upper_green[0]:
        print('green')
        return 3
    elif lower_red1[0] < hist_max[0] < upper_red1[0]:
        print('red')
        return 4
    elif lower_red2[0] < hist_max[0] < upper_red2[0]:
        print('red')
        return 4
    else:
        return 0

# 返回启停区识别函数，通过对特定区域二值化图像做像素点统计方式实现
def ruku_find(img):
    frame = cv2.GaussianBlur(img, (13, 13), 10, 20)
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    bin1 = cv2.inRange(hsv, lower_blue, upper_blue)
```

```
    bin2 = bin1[300:420, 200:350]
    white_pixels = np.sum(bin2 == 255)
    print(white_pixels)
    return white_pixels

# 两个物资识别，编号
print("start")
while True:
    flag, frame = cap.read()
    frame = frame[100:260, 160:320]
    put1 = color_find(frame)
    if put1 == 0:
        continue
    else:
        break
while True:
    flag, frame = cap.read()
    frame = frame[100:260, 160:320]
    put2 = color_find(frame)
    if put2 == put1 or put2 == 0:
        continue
    else:
        break
data2 = str(put1) + "/" + str(put2) + "/"
print(data2)

# 建立与运动部分 socket 通信
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_ip = '192.168.123.13'
server_port = 6011
server_socket.bind((server_ip, server_port))
server_socket.listen()
print('waiting for client')
client_socket, client_address = server_socket.accept()
print('connecting!!')
cv2.setNumThreads(1)

# 变量定义
point_centre_up = 220   # 所取直线的的上中点
point_centre_down = 220   # 所取直线的下中点
angle = 1.57   # 弧度
distance = 0   # 横向偏差
extent = 0   # 上中点到界外的垂直距离
scan1 = -4   # 扫线起点偏移量
```

```
point_left_down_state = 0   # 左下拐点状态
sum_left = 0   # 上（下）最终左边线横坐标和
sum_right = 0   # 上（下）最终右边线横坐标和
area = 0   # 特殊颜色的面积
count_point = 0   # 拐点识别次数
avoid_count = 0   # 避障次数统计
corner = 0   # 倾倒区计数
counter = 0   # 计数器计数
count = 0   # 避障标志位
# 进各倾倒区前计数器阈值
c0 = 30
c1 = 350
c2 = 120
c3 = 120
c4 = 70
rightConer = False     # 拐角标志位
rightroundabout = 0   # 拐角标志位
start_time = time.time()

# 根据是否进入倾倒区，对各计数器阈值做修正，优化判断
if put1 == 1:
    c1 == 450
if put1 == 2 or put2 == 2:
    c2 = 200
if put1 == 3 or put2 == 3:
    c3 = 220
if put2 == 4:
    c4 = 180

# 主函数循环
while True:
    ret, frame = cap.read()
    imgCopy = frame
    if not ret:
        break
    counter += 1
    frame = cv2.GaussianBlur(frame, (13, 13), 10, 20)
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    now_time = time.time()

    # 终点启停区识别
    if corner == 4 and ruku_find(frame) > 1500 and counter > c4:
        sentmes = "1.57/0/0/0/0/1/" + data2
        print("Final : " + sentmes)
```

```
            while True:
                sentmes = "1.57/0/0/0/0/1/" + data2
                client_socket.send(sentmes.encode())
        else:

        # 图像处理，做过滤、腐蚀、平滑等操作得到二值化图像
            cv2.putText(imgCopy, "avo= 0", (5, 150), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255, 255, 255), 2)
            bin = cv2.inRange(hsv, lower_yellow, upper_yellow)
            kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
            close = cv2.morphologyEx(bin, cv2.MORPH_CLOSE, kernel, iterations=1)
            num_labels, labels, stats, centroids = cv2.connectedComponentsWithStats(close, connectivity=8)
            image_filtered = np.zeros_like(close)
            for (i, label) in enumerate(np.unique(labels)):
                if label == 0:
                    continue
                if (stats[i][1] + stats[i][3]) > 350:
                    image_filtered[labels == i] = 255
            _,frame1 = cv2.threshold(image_filtered, 0, 255 ,cv2.THRESH_BINARY + cv2.THRESH_OTSU)

        # 道路过滤，避免图像中多个道路产生混叠和误判
            num_labels, labels, stats, centroids = cv2.connectedComponentsWithStats(~frame1, connectivity=8)
            image_filtered = np.zeros_like(close)
            for (i, label) in enumerate(np.unique(labels)):
                if label == 0:
                    continue
                if stats[i][-1] > 1500:   # 10000
                    image_filtered[labels == i] = 255
            img = ~image_filtered
            if (img[0][0] == 255) and (img[399][479] == 255):
                print("Out of road: 1")
                point_centre_down = 220
                point_centre_up = 220
                continue
            point_left_down_state = 0

            # 环岛倾倒区识别
            if (corner == 0 and counter > c0) or (corner == 1 and counter > c1):
                point_left_down_state = 0
                s = 330
                lc = 0
```

```
                for c in range(point_centre_down + scan1, 40, -1):
                    if c == 41:
                        lc = c
                        break
                    if img[s][c] == 0:
                        lc = c
                        break
                dif = 0
                for r in range(s - 1, 280, -1):
                    if img[r][lc] == 255:
                        for c in range(lc, lc - 25, -1):
                            if (img[r][c] == 0) or (c < (lc - 23)):
                                dif = lc - c
                                lc = c
                                break
                        if dif > 15:
                            point_left_down_state = 1
                            cv2.circle(imgCopy, (lc, r), 10, (100, 255, 255), -1)
                            cv2.putText(imgCopy, "Find A Corner", (120, 200),
cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 5)
                            break
                        else:
                            continue
                    if img[r][lc] == 0:
                        for c in range(lc, lc + 150, 2):
                            if (img[r][c] == 255) or (c > (lc + 70)):
                                dif = c - lc
                                lc = c
                                break
                        if dif > 20:
                            break
                        else:
                            continue

            # 十字倾倒区识别
            rightConer = 0
            white_point = 0
            if (counter > c2 and corner == 2) or (counter > c3 and corner == 3) or (counter >
c4 and corner == 4):
                for i in range(248, 252):
                    for j in range(0, 400, 1):
                        if img[i][j] == 255:
                            white_point += 1
                if white_point > 800:
```

```
                rightConer = 1

            # 环岛倾倒区通信发送，根据倾倒区识别所得标志位发送信息
            if point_left_down_state == 1:
                corner += 1
                half_time = time.time()
                counter = 0
                sentmes = "1.57/0/0/1/0/0/" + data2
                print("corner corner" + str(corner) + "     " + sentmes)
                client_socket.send(sentmes.encode())

            # 十字倾倒区通信发送，根据倾倒区识别所得标志位发送信息
            if rightConer and corner < 4:
                corner += 1
                half_time = time.time()
                counter = 0
                '''**********通讯发送'''
                sentmes = "1.57/0/0/1/0/0/" + data2
                print("corner corner" + str(corner) + "     " + sentmes)
                client_socket.send(sentmes.encode())

            # 避障识别，通过统计二值化图像特定区域像素点数目实现
            if avoid_count == 0:
                bin = cv2.inRange(hsv, lower_green, upper_green)
                cnts = cv2.findContours(bin, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_NONE)[0]
                area = 0
                for c in cnts:
                    tmp = cv2.contourArea(c)
                    if tmp > area:
                        area = tmp
            if (avoid_count == 0) and (area > 6000):
                avoid_count = 1
                counter -= 60
                cv2.putText(imgCopy, "avo= 1", (5, 150), cv2.FONT_HERSHEY_SIMPLEX, 0.75,
(255, 255, 255), 2)
                sentmes = "1.57/0/0/0/1/0/"+data2
                print("avoid : " + sentmes)
                client_socket.send(sentmes.encode())
            else:

                # 正常道路处理
                sum_left = 0
                sum_right = 0
```

```
        # 上方中点扫线
        for r in range(299, 302):
            cr = False
            cl = False

            # 特殊扫线，三叉路口
            if img[r][point_centre_up] == 0:
                for c in range(point_centre_up + scan1, 50, -2):
                    if (not cr) and (not cl) and (img[r][c] == 255):
                        sum_right += c
                        cr = True
                        continue
                    if (not cl) and (not cr) and (img[r][440 - c] == 255):
                        sum_left += 440 - c
                        cl = True
                        continue
                    if cr and (img[r][c] == 0):
                        sum_left += c
                        break
                    if cl and (img[r][440 - c] == 0):
                        sum_right += 440 - c
                        break
                    if (c < 60) and (not cl) and (not cr):
                        print("No line_ left/right")
                        cv2.putText(imgCopy, "No line_ left/right", (120, 300),
cv2.FONT_HERSHEY_SIMPLEX, 1.5,
                                    (200, 200, 200), 5)
                        break
                continue

            # 正常道路，常规扫线
            for c in range(point_centre_up + scan1, 80, -3):
                if c <= 85:
                    sum_left += c
                    break
                if img[r][c] == 0:
                    sum_left += c
                    break
            for c in range(point_centre_up + scan1, 360, 3):
                if c >= 355:
                    sum_right += c
                    break
                if img[r][c] == 0:
```

```
                    sum_right += c
                    break

        # 根据上点周围五行中点位置，计算平均值
        point_centre_up = int((sum_right + sum_left) / 6)

        # 下方中点扫线
        sum_left = 0
        sum_right = 0
        for r in range(349, 352):
            cr = False
            cl = False

            # 特殊扫线，三岔路口
            if img[r][point_centre_down] == 0:
                for c in range(point_centre_down, 50, -3):
                    if (not cr) and (not cl) and (img[r][c] == 255):
                        sum_right += c
                        cr = True
                        continue
                    if (not cl) and (not cr) and (img[r][440 - c] == 255):
                        sum_left += 440 - c
                        cl = True
                        continue
                    if cr and (img[r][c] == 0):
                        sum_left += c
                        break
                    if cl and (img[r][440 - c] == 0):
                        sum_right += 440 - c
                        break
                    if (c < 60) and (not cl) and (not cr):
                        print("No line_ left/right")
                        cv2.putText(imgCopy, "No line_ left/right", (120, 300),
cv2.FONT_HERSHEY_SIMPLEX, 1.5,
                                    (200, 200, 200), 5)
                        break
                continue

            # 正常道路，常规扫线
            for c in range(point_centre_down, 0, -3):
                if c <= 10:
                    sum_left += c
                    break
                if img[r][c] == 0:
```

```
                    sum_left += c
                    break
            for c in range(point_centre_down, 480, 3):
                if c >= 470:
                    sum_right += c
                    break
                if img[r][c] == 0:
                    sum_right += c
                    break

        # 根据下点周围五行中点位置，计算平均值
        point_centre_down = int((sum_right + sum_left) / 6)

        # 计算偏转角 angle 和偏转距离 distance
        if abs(point_centre_down - point_centre_up) < 2:
            angle = 1.57
        else:
            k = - 50 / (point_centre_down - point_centre_up)
            angle = np.arctan(k)
        distance = int(220 - point_centre_down)

        # 计算上中点到界外的垂直距离 extent
        extent = 0
        for r in range(300, 0, -2):
            if img[r][point_centre_up] == 0:
                break
            extent += 2
        for r in range(300, 0, -2):
            if img[r][point_centre_up + 3] == 0:
                break
            extent += 2
        for r in range(300, 0, -2):
            if img[r][point_centre_up - 3] == 0:
                break
            extent += 2
        extent /= 3
        if (extent <= 15) and ((1.39 < angle) or (angle < -1.39)):
            cv2.putText(imgCopy, "Find A Fork", (120, 100),
cv2.FONT_HERSHEY_SIMPLEX, 1.5, (0, 0, 255), 5)
            angle = -0.57
            print("fork**********")
            point_centre_up = 120

        # 通讯发送
```

```
                sentmes = str(angle) + "/" + str(distance) + "/" + str(extent) + "/0/0/0/" + data2
                client_socket.send(sentmes.encode())

# 执行完成，释放摄像头，关闭 socket 通信，关闭 cv2 窗口
cap.release()
client_socket.shutdown(2)
client_socket.close()
cv2.destroyAllWindows()
```

2.3 机械臂控制程序

```
#include <Wire.h>
#include <Adafruit_PWMServoDriver.h>

Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver();

//180 度舵机
#define SERVO_0 102
#define SERVO_140 384
#define SERVO_60 218
#define SERVO_20 121
#define SERVO_48 200

// 定义旋转时间（毫秒）
unsigned long startTime;
unsigned long duration = 600;
int ll = 0 ;
int lr = 0;
int rr = 0;
int count = 0;
int pos = 0;

// 初始化串口通信，初始化舵机
void setup() {
  Serial.begin(9600);
  pwm.begin();
  pwm.setPWMFreq(50);
  delay(10);
  pwm.setPWM(1, 0, 200);
  pwm.setPWM(2, 0, 200);
  pwm.setPWM(3, 0, 102);
  delay(1000);
}

// 机械臂控制函数
```

```
void loop() {
  while(count == 0){
    for(pos = 140; pos<=285 ; pos +=2){
      pwm.setPWM(0, 0, pos);
      delay(75);
    }
    delay(1000);
    pwm.setPWM(1, 0, 380);
    //delay(1000);
    pwm.setPWM(2, 0, 380);
    delay(1000); //  等待 1 秒
    for(pos = 285; pos>=140     ; pos -=2){
      pwm.setPWM(0, 0, pos);
      delay(75);
    }

    count = 1;
  }
  pwm.setPWM(3, 0, 280);
  delay(1000);

// 串口读取数据，12 左左  13 左右  14 左右  23 左右  24 左右  34 右右
  if (Serial.available() > 0) {
    char incomingByte = Serial.read();
    if (incomingByte == '4') {
      ll = 1;
    } else if (incomingByte == '5') {
      rr = 1;
    } else if(incomingByte == '6'){
        lr = 1;
    }

// 根据接收到的字符控制舵机
    if (incomingByte == '2') {
        if(ll == 1 || lr == 1){
          pwm.setPWM(3, 0, 500);
          delay(1000);
          for(pos = 140; pos<=240 ; pos +=4){
            pwm.setPWM(0, 0, pos);
            delay(75);
          }
          pwm.setPWM(1, 0, 200);
          delay(1000);
          for(pos = 240; pos>=140     ; pos -=4){
```

```
            pwm.setPWM(0, 0, pos);
            delay(75);
          }
          pwm.setPWM(3, 0, 280);
          delay(1000);
        }else if(rr == 1){
          pwm.setPWM(3, 0, 102);
          delay(1000);
          for(pos = 140; pos<=240 ; pos +=4){
            pwm.setPWM(0, 0, pos);
            delay(75);
          }
          pwm.setPWM(1, 0, 200);
          delay(1000);
          for(pos = 240; pos>=140     ; pos -=4){
            pwm.setPWM(0, 0, pos);
            delay(75);
          }
          pwm.setPWM(3, 0, 280);
          delay(1000);
        }
} else if (incomingByte == '3') {
        if(ll == 1){
          pwm.setPWM(3, 0, 500);
          delay(1000);
          for(pos = 140; pos<=240 ; pos +=4){
            pwm.setPWM(0, 0, pos);
            delay(75);
          }
          pwm.setPWM(2, 0, 200);
          delay(1000);
          for(pos = 240; pos>=140     ; pos -=4){
            pwm.setPWM(0, 0, pos);
            delay(75);
          }
          pwm.setPWM(3, 0, 280);
          delay(1000);
          }else if(rr == 1 || lr == 1){
            pwm.setPWM(3, 0, 102 );
            delay(1000);
            for(pos = 140; pos<=240 ; pos +=4){
            pwm.setPWM(0, 0, pos);
            delay(75);
            }
```

```
                pwm.setPWM(2, 0, 200);
                delay(1000);
                for(pos = 240; pos>=140     ; pos -=4){
                pwm.setPWM(0, 0, pos);
                delay(75);
                }
                pwm.setPWM(3, 0, 280);
                delay(1000);
            }
        }
    }
}
```

3. 问题分析与技术方案

3.1 通信方式

3.1.1 串口通信

由于机械臂采用 arduino 控制板与电机驱动板控制，而 arduino 直接通过串口数据线与机器狗相连，故运动部分与机械臂控制部分直接使用串口通信更加简洁。

为各种倾倒情况与物资识别约定了对应的标识字符，由于无需机械臂回传数据，实际上为运动主板到 arduino 控制板的单工通信，通过运动主板传递对应字符，arduino 在串口读取数据并解析，完成对应操作。

约定字符：字符 2、3 分别对应左、右侧机械爪的开合，字符 4、5、6 分别对应倾倒区方向为左左、右右、左右，以实现对机械臂舵机处理的通信，提高投放效率。

3.1.2 socket 通信

由于视觉处理部分与运动控制部分在同一局域网下，故直接使用 socket 通信更为简洁，主要问题是约定一个简洁高效的信息编码与发送方式。

在 socket 通信协议下，由于所传数据复杂度不高，无需设计专门的编码方式，我们选择直接采用字符串进行传递，并通过特殊字符做分割，目前采用/字符做数据分割。在一次数据传输中，采用 angle/distance/velocity/signal/bz/ruku/put1/put2，分别传输偏转角、直线距离、垂直距离、倾倒区标志、避障标志、返回启停区标志、目标倾倒区 1 号码、目标倾倒区 2 号码，完成一次数据传递。

使用了阻塞式 socket，设置超时时间 1s，在超时后执行转向寻线操作，以有效防止偏离赛道造成的无法正常行进情况。但对应某些特殊情况，例如在倾倒物资时，由于耗时较长，在阻塞情况下缓冲区产生大量数据，可能造成数据分割错误，因此在倾倒时对视觉处理部分进行了 time.sleep()操作，通过秒级延时，避免缓存区数据堆积。

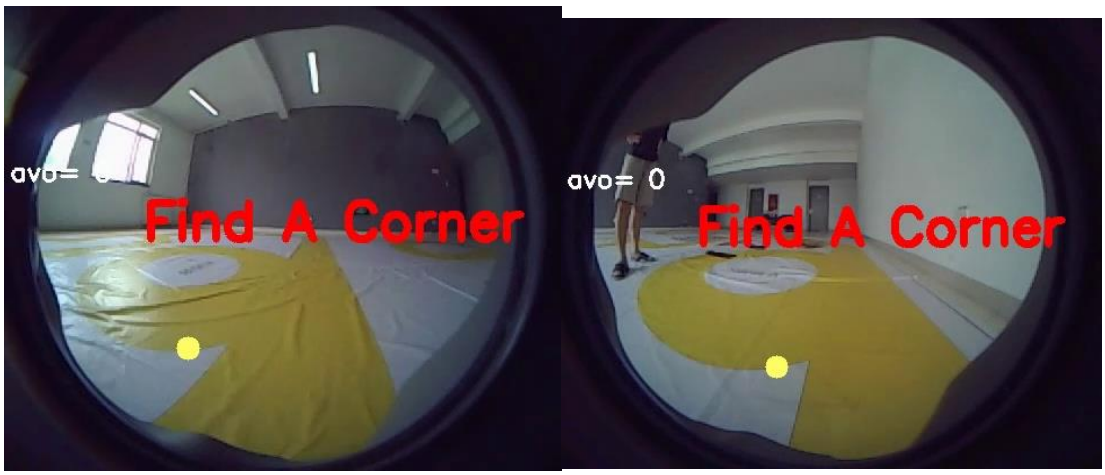
3.2 倾倒区识别

为了提高识别效率与准确度，对与环岛倾倒区域十字倾倒区设计了不同的识别方式做识别。

3.2.1 环岛倾倒区识别

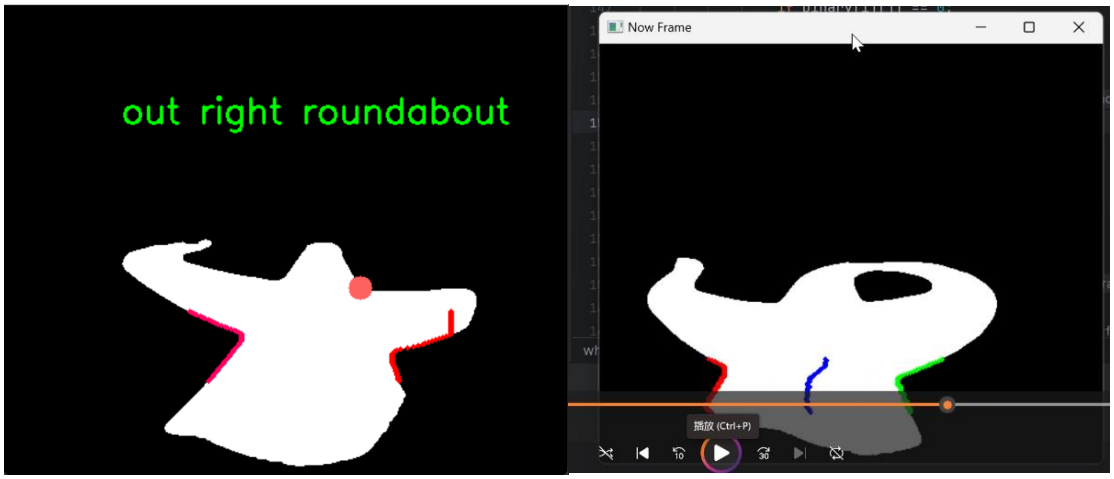
对于两个环岛倾倒区，在进入时特征表现相同，采用基于边线落差来寻找左下拐点的方式实现。即在二值化图像中，根据设定的行号，由边缘点向左搜索寻找黑色像素点，如找到则更新边缘点列号，同时计算落差，如果落差大于设定阈值，就认为找到了左下拐点，即识

别到环岛倾倒区。



3.2.2 十字倾倒区识别

对于两个十字倾倒区，在进入时特征表现相同，十字路口在二值化图像中表现为某些行白色像素点的急速增多，但对于拐点需同时识别左右两个，故我们选择对指定行进行白色像素点统计的方式进行识别。设置了白色像素点阈值，在对制定行进行统计后，判断白色像素点数目是否大于阈值，超出阈值即认为识别到十字倾倒区。



3.3 终点识别

由于机器人头部摄像头及运动姿态影响，对终点识别选取的方案是截取摄像头右下方的特定区域，之后在二值化图像中对白色像素点轮廓进行处理，计算轮廓面积，如果轮廓面积大于特定值，则认为识别到启停区，发送通信数据，执行返回部分。

3.4 扫线实现

3.4.1 常规扫线

常规扫线主要思路为提取中线，找左右边线，然后计算中线，首先寻找当前行中点位置，若为白色像素，则寻找左右边线并记录入数组，根据五行中点位置，取平均值作为最终值。每帧对于图像上部五行像素点与下部五行像素点做上述操作，最终得到两个中心点坐标。

之后对这两个中心点坐标做运算，借助两中心点间斜率取反正切来计算偏向角 angle，使用中心点与坐标中心的横向差值来获得横向偏移 distance，在上方中心点根据选取的左

右像素阈值向下逐行搜索，之后根据设定好的计算方式计算大致活动前方垂直距离 extent。

在运动控制中，根据这三个值指导通过运算，进行运动。

3.4.2 三叉扫线

在上方中点为黑色像素点的情形下，首先以设定步长向左寻找白色像素点，如在规定阈值内找到，则以其为最终值，与下方中点一起参与计算，如未找到，则向右寻找，重复上述过程。

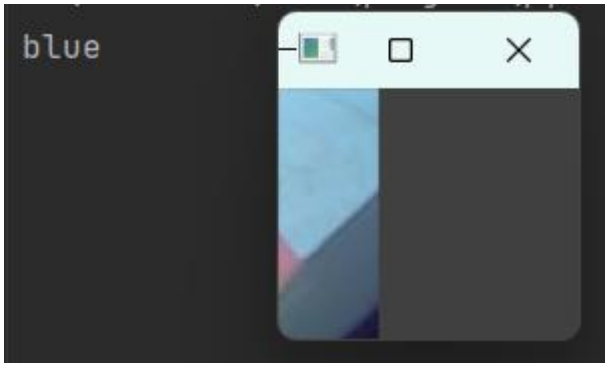
我们做如此扫线决策，主要有以下考量：首先，左侧为直道，便于做处理，转弯情况少，时间更快；其次，在直道上可以修改速度加速，进一步提升时间；最后，直道不易偏离赛道，有效避免罚时。

3.5 物资识别

由于我们选取的物资有明显的颜色差异，且与赛道颜色不同，故在识别部分直接基于 hsv 空间使用统计特定区域像素值数量的方法实现。

首先，设定各颜色 hsv 值，并于物资建立一一对应的映射关系，之后对图像做裁剪处理，截取中心部分画面，避免周围黑色像素影响。对截取到的画面做二值化处理，之后使用 opencv 库对应函数获取区域内像素点最多的颜色，打印输出，并对 put1、put2 标志位赋对应值，指导后续倾倒区选择与机械臂控制。

图



3.6 计数器控制

为了节省计算资源，提高运算时间，保证程序鲁棒性，我们对各非持续部分进行封装，由计数器控制达阈值后再运行判断，如各环岛识别与终点识别部分，都是计数器道对应值之后，才会启动判断，之后再次停止判断，提高运行效率。

同时对于某些特殊位置，如终点等，还加入运行时间判断，由计数器与计时器做双保障，避免某些错误。

3.7 速度控制

3.7.1 前进速度

首先，前进速度使用获取到的垂直距离参数，设定不同档次，由垂直距离的大小，控制前进速度，即前方垂直距离越长，所设定的速度越快，以减少耗时。具体计算在函数中有体现，通过多次测试，设定了一系列参数运算获得阈值。

3.7.2 PID 控制

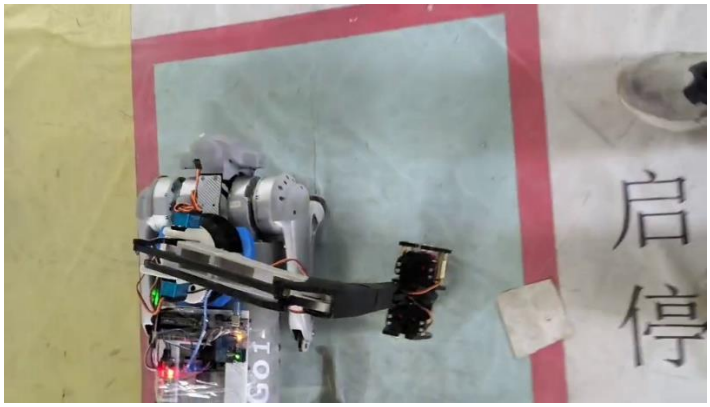
其次为了提高系统的稳定性与运行效率，对偏转角和横向偏移两个参数，我们还引入了 PID 控制算法，主要使用了 P、D 两个参数，提高在快速运行的响应速度与系统稳定性，避免了误差堆积、超调、震荡等问题。

通过多次调试，获取了相对稳定有效的 PID 参数。

3.8 物资倾倒

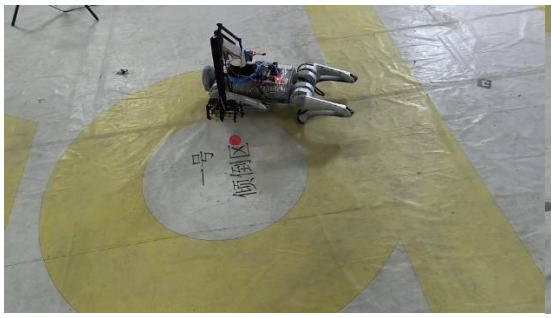
3.8.1 机械臂控制

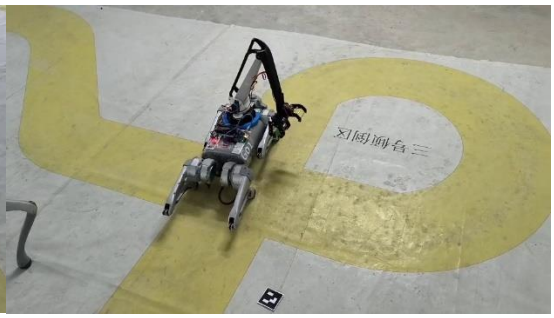
机械臂使用 arduino 板外接舵机控制板控制机械臂舵机运动，机械臂设计了两个机械爪，以满足物资抓取设定，同时提高投放效率。经过实测，为了不影响机器狗在快速移动时的重心，选择将机械臂朝向设置为后向放置，每次投递物资时转动大臂完成倾倒，之后恢复初始状态。



3.8.2 蹲下站立

经过实测，如在机器狗站立情况下使用机械臂倾倒物资，稳定性不强，且可能造成物资的滚动与偏移。因此我们选用蹲下-倾倒-站立的模式完成物资倾倒，在识别到对应环岛，且经过视觉处理判断后，如需倾倒，则首先执行下蹲，之后通过串口发送倾倒指令，在倾倒完成后执行站立，继续运动。如不需倾倒，则不执行下蹲站立执行，继续由视觉处理获取参数指定运动。





3.9 避障处理

3.9.1 避障识别

避障识别采用统计二值化图像特定区域像素点的方式实现，首先根据绿色阈值与划定的图像区域，对图像进行二值化处理，之后在处理完的二值化图像中，对白色像素点（即前景）进行计数，根据计数结果与阈值大小关系判断是否遇到障碍物。

3.9.2 避障执行

整体避障运动思路较为简单，设置为接收到避障信息后，机器狗依次执行中断道路识别程序，执行预设左移、直行、右转步骤，之后恢复道路识别程序，继续由视觉控制运动。经过多次测试，对左移、直行、右转步骤的参数做了调整，效果达到预期。

3.9.3 其他处理

为了避免在运动中视觉部分多次识别障碍物造成多次避障以及误识别产生的影响，采用了双重保险措施。在视觉部分与运动部分均设置了标志位，在视觉部分，识别到障碍物后即将避障识别标志位置零，此后运动中再不会执行避障识别代码，既避免了后续误识别，也节约了算力。在运动部分，同样接收到视觉传来的避障信息后即将避障运动标志位置零，此后不再执行避障运动代码，既避免了视觉程序暂停带来的 socket 阻塞的影响，也节约了算力。

同时，考虑到避障运动与正常运动的差异，经过多次调试，在避障程序中对倾倒区计数器阈值进行了细微修正，有效节约了算力并提高倾倒区识别准确性。

4. 结果感想

参与本次四足机器人物资运送比赛，我们收获颇丰，也深感其中的挑战与乐趣并存。整个参赛过程不仅是对我们技术能力的考验，更是对我们团队协作和问题解决能力的锻炼。

首先，我们深感在比赛中遇到的各种技术问题与挑战的重要性。在设计和实现四足机器人的物资运送系统时，我们遇到了视觉识别准确性、抗干扰能力以及机械臂稳定性等多方面的挑战。这些问题让我们更加深入地理解了机器人技术的复杂性和精细性，也让我们更加明白了不断学习和创新的重要性。

其次，我们深刻体会到了团队协作的力量。在比赛过程中，我们团队成员之间紧密合作，共同攻克技术难关。我们通过定期的交流和讨论，不断优化和完善系统设计，最终成功完成了比赛任务。这次经历让我们更加珍惜团队中的每一个成员，也让我们更加明白了团队协作对于项目成功的重要性。

此外，我们还从比赛中汲取了宝贵的经验。我们学会了如何在有限的时间内高效地解决问题，如何调整策略以适应不同的比赛场景，以及如何在压力下保持冷静和专注。这些经验对于我们未来的学习和工作都将具有积极的指导意义。

最后，我们要感谢组织者和评委们的辛勤付出。他们为我们提供了一个展示自己才华和能力的平台，也为我们提供了宝贵的反馈和建议。我们将认真听取评委们的建议，继续改进和完善我们的系统，争取在未来的比赛中取得更好的成绩。

总的来说，这次参赛经历对我们来说是一次宝贵的学习和锻炼机会。我们将以更加饱满的热情和更加扎实的技术基础，继续探索机器人技术的奥秘，为推动该领域的发展贡献自己的力量。