颜色追踪

注：虚拟机、ROS-wifi图传模块需要与microROS控制板ROSDOMAINID需要一致，都要设置成20，可以查看【MicroROS控制板参数配置】来设置microROS控制板ROSDOMAINID，查看教程【连接MicroROS代理】判断ID是否一致。

运行实验前请确保microros小车和ROS-wifi图传模块在虚拟机(linux带有humbleROS2系统)上正确开启了代理

1、程序功能说明

MicroROS机器人颜色追踪，具备可以随时识别多种颜色，并自主储存当前识别的颜色，控制小车云台追随检测识别的颜色。

MicroROS机器人的颜色追踪还可以实现HSV实时调控的功能，通过调节HSV的高低阈值，过滤掉干扰的颜色，使得方块在复杂的环境中能够非常理想的被识别出来，如果在取色中效果不理想的话，这个时候需要将小车移动到不同环境下校准一下，以达到可以在复杂环境中，识别我们所需的颜色。

2、程序代码参考路径

~/yahboomcar\_ws/src/yahboom\_esp32ai\_car/yahboom\_esp32ai\_car/colorHSV.py  
~/yahboomcar\_ws/src/yahboom\_esp32ai\_car/yahboom\_esp32ai\_car/colorTracker.py

colorHSV.py

主要是完成图像处理，计算出被追踪物体的中心坐标。

colorTracker.py

主要是根据被追踪物体的中心坐标和深度信息，计算出舵机运动角度数据给底盘。

3、程序启动

3.1、启动命令

终端输入，

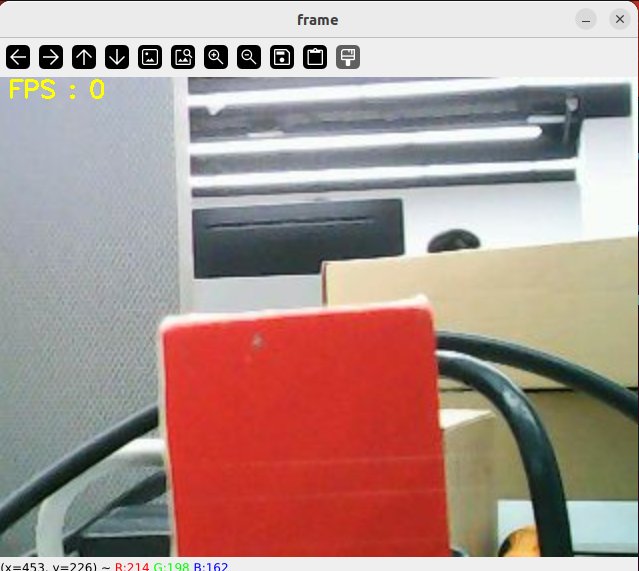
#启动舵机校准程序  
#当舵机的角度不处于中间的时候，需要执行以下命令  
ros2 run yahboom\_esp32\_mediapipe control\_servo   
  
# 舵机运动后，舵机还是不处于中间，需要手动重新安装云台，可以不断电操作，让舵机处于一个锁住状态，方便调整

舵机处于中间后，要按"Ctrl+C"终止程序，避免影响接下来的实验操作

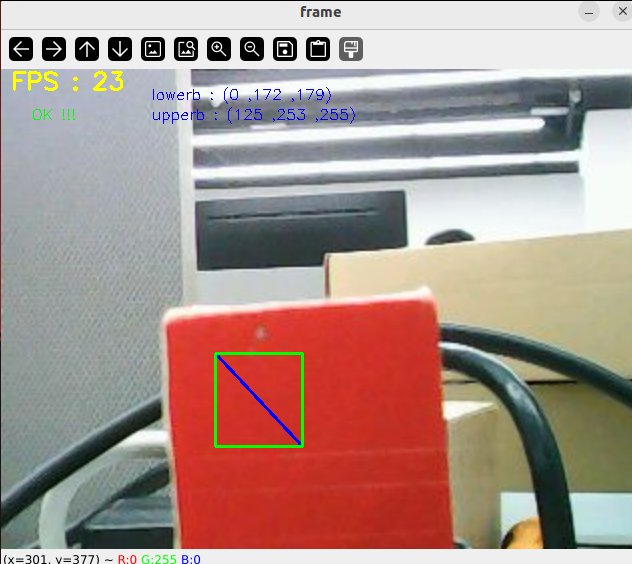
#启动颜色追踪程序  
ros2 run yahboom\_esp32ai\_car colorHSV  
ros2 run yahboom\_esp32ai\_car colorTracker

如果摄像头的画面图像出现倒置，需要看到3.摄像头画面纠正(必看)文档自己纠正，该实验不再阐述。

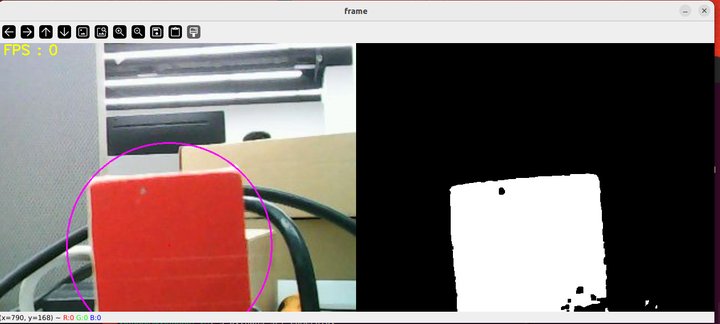
以追踪红色为例，程序启动后，会出现以下画面，



然后按下键盘上的r/R键进入选色模式，用鼠标框出一片区域（该区域只能有一中颜色），



选完后，效果如下图，

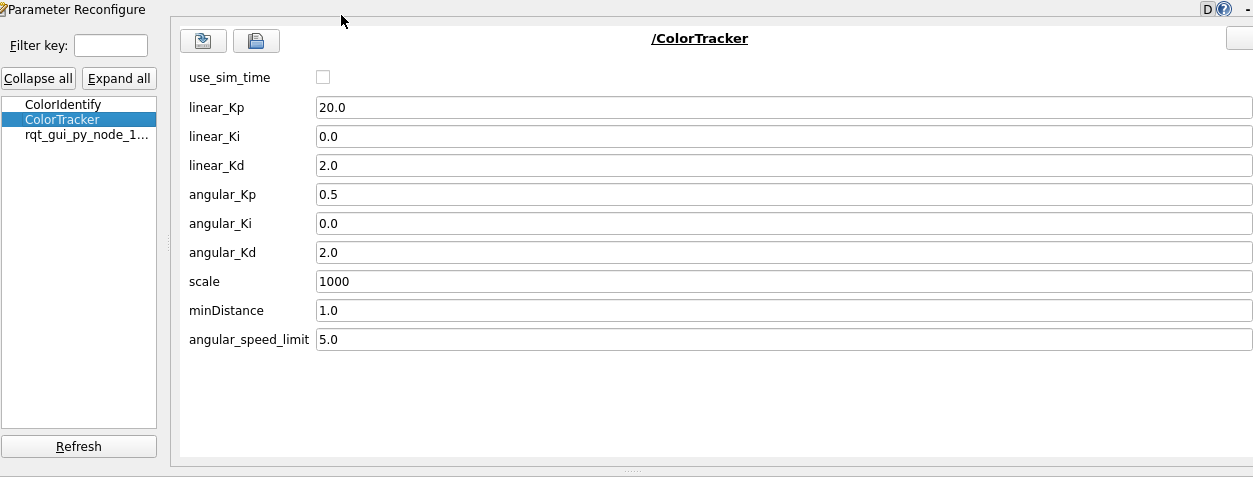


然后，按下空格键进入追踪模式，缓慢移动物体，能看到机器人云台会追踪色块运动。

3.2、动态参数调节

Docker终端输入，

ros2 run rqt\_reconfigure rqt\_reconfigure



修改完参数，点击GUI空白处写入参数值。由上图可知，

colorTracker主要调节PID三个参数来让云台更灵敏。

4、核心代码

4.1、colorHSV.py

这个程序主要有以下几个功能：

打开摄像头，获取图像；

获取键盘鼠标事件，用于切换模式和取色；

处理图像以及发布追踪物体的中心坐标并且发布出去。

部分核心代码如下，

#创建发布者，发布追踪物体的中心坐标  
self.pub\_position = self.create\_publisher(Position,"/Current\_point", 10)  
#获取键盘鼠标事件，得到hsv的值；  
 if action == 32: self.Track\_state = 'tracking'  
 elif action == ord('i') or action == ord('I'): self.Track\_state = "identify"  
 elif action == ord('r') or action == ord('R'): self.Reset()  
 elif action == ord('q') or action == ord('Q'): self.cancel()  
 if self.Track\_state == 'init':  
 cv.namedWindow(self.windows\_name, cv.WINDOW\_AUTOSIZE)  
 cv.setMouseCallback(self.windows\_name, self.onMouse, 0)  
 if self.select\_flags == True:  
 cv.line(rgb\_img, self.cols, self.rows, (255, 0, 0), 2)  
 cv.rectangle(rgb\_img, self.cols, self.rows, (0, 255, 0), 2)  
 if self.Roi\_init[0] != self.Roi\_init[2] and self.Roi\_init[1] != self.Roi\_init[3]:  
 rgb\_img, self.hsv\_range = self.color.Roi\_hsv(rgb\_img, self.Roi\_init)  
 self.gTracker\_state = True  
 self.dyn\_update = True  
 else: self.Track\_state = 'init'  
#计算中心坐标的值，self.circle存放xy值  
rgb\_img, binary, self.circle = self.color.object\_follow(rgb\_img, self.hsv\_range)  
#发布中心坐标的消息  
threading.Thread(target=self.execute, args=(self.circle[0], self.circle[1], self.circle[2])).start()  
def execute(self, x, y, z):  
 position = Position()  
 position.anglex = x \* 1.0  
 position.angley = y \* 1.0  
 position.distance = z \* 1.0  
 self.pub\_position.publish(position)

4.2、colorTracker.py

这个程序主要有以下几个功能：接收/Current\_point和深度图像话题数据，计算速度大小，然后发布速度数据。

部分代码如下，

#定义订阅者接收需要的话题数据  
self.sub\_depth = self.create\_subscription(Image,"/image\_raw", self.depth\_img\_Callback, 1)  
self.sub\_JoyState =self.create\_subscription(Bool,'/JoyState',self.JoyStateCallback,1)  
self.sub\_position =self.create\_subscription(Position,"/Current\_point",self.positionCallback,1)  
#定义速度发布者  
self.pub\_cmdVel = self.create\_publisher(Twist,'/cmd\_vel',10)  
self.pub\_Servo1 = self.create\_publisher(Int32,"servo\_s1" , 10)  
self.pub\_Servo2 = self.create\_publisher(Int32,"servo\_s2" , 10)  
  
#两个重要的回调函数，获取到self.Center\_x值和distance\_值  
def positionCallback(self, msg):  
def depth\_img\_Callback(self, msg):  
#self.Center\_x值和distance\_值根据计算线速度，角速度  
self.execute(self.Center\_x, distance\_)  
 def execute(self, point\_x, point\_y):  
 [x\_Pid, y\_Pid] = self.linear\_pid .update([point\_x - 320, point\_y - 240])  
 if self.img\_flip == True:  
 self.PWMServo\_X += x\_Pid  
 self.PWMServo\_Y += y\_Pid  
 else:  
 self.PWMServo\_X -= x\_Pid  
 self.PWMServo\_Y += y\_Pid  
  
 if self.PWMServo\_X >= 45:  
 self.PWMServo\_X = 45  
 elif self.PWMServo\_X <= -45:  
 self.PWMServo\_X = -45  
 if self.PWMServo\_Y >= 40:  
 self.PWMServo\_Y = 40  
 elif self.PWMServo\_Y <= -90:  
 self.PWMServo\_Y = -90  
  
 # rospy.loginfo("target\_servox: {}, target\_servoy: {}".format(self.target\_servox, self.target\_servoy))  
 print("servo1",self.PWMServo\_X)  
 servo1\_angle = Int32()  
 servo1\_angle.data = int(self.PWMServo\_X)  
 servo2\_angle = Int32()  
 servo2\_angle.data = int(self.PWMServo\_Y)  
 self.pub\_Servo1.publish(servo1\_angle)  
 self.pub\_Servo2.publish(servo2\_angle)