1. Color tracking

1.1. Experimental Objectives

In this section, we combine hardware to complete the camera pan-tilt color tracking. By identifying the red object and detecting the difference between the x, y coordinates of the circumscribed circle of the identified color and the center of the picture, we use the PID algorithm to control the servo movement so that the identified target is located in the center of the picture.

1.2. Experimental Code

Source code path:

/home/pi/project_demo/08.AI_Visual_Interaction_Course/01.Color_tracking/01_Color_Tracking_with_PTZ.ipynb

```
#导入Raspbot驱动库 Import the Raspbot library
from Raspbot_Lib import Raspbot
# 创建Rosmaster对象 bot Create the Rosmaster object bot
bot = Raspbot()
import sys
sys.path.append('/home/pi/software/oled_yahboom/')
from yahboom_oled import *
# 创建oled对象 Create an oled object
oled = Yahboom_OLED(debug=False)
```

```
import numpy as np
import math
#bgr8转jpeg格式 bgr8 to jpeg format
import enum
import cv2
def bgr8_to_jpeg(value, quality=75):
    return bytes(cv2.imencode('.jpg', value)[1])
```

```
#显示摄像头组件 Display camera components
import traitlets
import ipywidgets.widgets as widgets
from IPython.display import display
import time
# 线程功能操作库 Thread function operation library
import threading
import inspect
import ctypes


image_widget = widgets.Image(format='jpeg', width=640, height=480)
```

```
def _async_raise(tid, exctype):
    """raises the exception, performs cleanup if needed"""
    tid = ctypes.c_long(tid)
    if not inspect.isclass(exctype):
        exctype = type(exctype)
    res = ctypes.pythonapi.PyThreadState_SetAsyncExc(tid,
ctypes.py_object(exctype))
    if res == 0:
        raise ValueError("invalid thread id")
    elif res != 1:
        # """if it returns a number greater than one, you're in trouble,
        # and you should call it again with exc=NULL to revert the effect"""
        ctypes.pythonapi.PyThreadState_SetAsyncExc(tid, None)

def stop_thread(thread):
    _async_raise(thread.ident, SystemExit)
```

```
image = cv2.VideoCapture(0)
image.set(3, 640)
image.set(4, 480)
image.set(5, 30)  #设置帧率 Setting the frame rate
# image.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
# image.set(cv2.CAP_PROP_BRIGHTNESS, 62) #设置亮度 -64 - 64  0.0 Set Brightness
-64 - 64 0.0
# image.set(cv2.CAP_PROP_CONTRAST, 63) #设置对比度 -64 - 64  2.0 Set Contrast -64
- 64 2.0
# image.set(cv2.CAP_PROP_EXPOSURE, 4800) #设置曝光值 1.0 - 5000  156.0 Set the
exposure value 1.0 - 5000 156.0
ret, frame = image.read()


#CSI
# from picamera2 import Picamera2, Preview
# import libcamera
# picam2 = Picamera2()
# camera_config = picam2.create_preview_configuration(main=
{"format":'RGB888',"size":(320,240)})
# camera_config["transform"] = libcamera.Transform(hflip=0, vflip=1)
# picam2.configure(camera_config)
# picam2.start()
# frame = picam2.capture_array()
```

```
image_widget.value = bgr8_to_jpeg(frame)
```

```
global g_mode
g_mode = 0
global color_x, color_y, color_radius
color_x = color_y = color_radius = 0
global target_valuex
target_valuex = 1500
global target_valuey
target_valuey = 1500
```

```
global color_lower
#color_lower = np.array([0, 43, 46])
global color_upper
#color_upper = np.array([10, 255, 255])

color_lower = np.array([0,70,72])
color_upper = np.array([7, 255, 255])
```

```
Redbutton = widgets.Button(
    value=False,
    description='red',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='uncheck' )
Greenbutton = widgets.Button(
    value=False,
    description='green',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='uncheck' )
Bluebutton = widgets.Button(
    value=False,
    description='blue',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='uncheck' )
Yellowbutton = widgets.Button(
    value=False,
    description='yellow',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='uncheck' )
Orangebutton = widgets.Button(
    value=False,
    description='orange',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
```

```python
    tooltip='Description',
    icon='uncheck' )
Closebutton = widgets.Button(
    value=False,
    description='close',
    disabled=False,
    button_style='', # 'success', 'info', 'warning', 'danger' or ''
    tooltip='Description',
    icon='uncheck' )
output = widgets.Output()

def ALL_Uncheck():
    Redbutton.icon = 'uncheck'
    Greenbutton.icon = 'uncheck'
    Bluebutton.icon = 'uncheck'
    Yellowbutton.icon = 'uncheck'
    Orangebutton.icon = 'uncheck'

def on_Redbutton_clicked(b):
    global color_lower, color_upper, g_mode
    global target_valuex, target_valuey
    ALL_Uncheck()
    b.icon = 'check'
    #color_lower = np.array([0, 43, 46])
    #color_upper = np.array([10, 255, 255])
    color_lower = np.array([0,43,89])
    color_upper = np.array([7, 255, 255])
    g_mode = 1
    with output:
        bot.Ctrl_WQ2812_ALL(1,0)#红色 red
        oled.clear()
        oled.add_line("Color_Tracking", 1)
        oled.add_line("color: red", 3)
        oled.refresh()
        print("RedButton clicked.")

def on_Greenbutton_clicked(b):
    global color_lower, color_upper, g_mode
    global target_valuex, target_valuey
    ALL_Uncheck()
    b.icon = 'check'
    #color_lower = np.array([35, 43, 46])
    #color_upper = np.array([77, 255, 255])
    color_lower = np.array([54,104,64])
    color_upper = np.array([78, 255, 255])
    g_mode = 1
    with output:
        bot.Ctrl_WQ2812_ALL(1,1)#绿色 green
        oled.clear()
        oled.add_line("Color_Tracking", 1)
        oled.add_line("color: green", 3)
        oled.refresh()
        print("GreenButton clicked.")

def on_Bluebutton_clicked(b):
    global color_lower, color_upper, g_mode
    global target_valuex, target_valuey
    ALL_Uncheck()
    b.icon = 'check'
    #color_lower=np.array([100, 43, 46])
    #color_upper = np.array([124, 255, 255])
    color_lower = np.array([92,100,62])
    color_upper = np.array([121, 255, 255])
    g_mode = 1
    with output:
        bot.Ctrl_WQ2812_ALL(1,2)#蓝色 blue
        oled.clear()
        oled.add_line("Color_Tracking", 1)
        oled.add_line("color: blue", 3)
        oled.refresh()
        print("Bluebutton clicked.")

def on_Yellowbutton_clicked(b):
    global color_lower, color_upper, g_mode
    global target_valuex, target_valuey
    ALL_Uncheck()
    b.icon = 'check'
    #color_lower = np.array([26, 43, 46])
    #color_upper = np.array([34, 255, 255])
    color_lower = np.array([26,100,91])
    color_upper = np.array([32, 255, 255])
    g_mode = 1
    with output:
        bot.Ctrl_WQ2812_ALL(1,3)#黄色 yellow
        oled.clear()
        oled.add_line("Color_Tracking", 1)
        oled.add_line("color: yellow", 3)
        oled.refresh()
        print("Yellowbutton clicked.")

def on_Orangebutton_clicked(b):
    global color_lower, color_upper, g_mode
    global target_valuex, target_valuey
    ALL_Uncheck()
    b.icon = 'check'
    color_lower = np.array([11, 43, 46])
    color_upper = np.array([25, 255, 255])
    g_mode = 1
    with output:
        bot.Ctrl_WQ2812_brightness_ALL(255, 48, 0)
        oled.clear()
        oled.add_line("Color_Tracking", 1)
        oled.add_line("color: orange", 3)
        oled.refresh()
        print("Orangebutton clicked.")

def on_Closebutton_clicked(b):
    global g_mode
    ALL_Uncheck()
    g_mode = 0
    with output:
        bot.Ctrl_WQ2812_ALL(0,0)
        oled.clear()
        oled.add_line("Color_Tracking", 1)
        oled.add_line("color: none", 3)
        oled.refresh()
        bot.Ctrl_Servo(1,90)
        bot.Ctrl_Servo(2,25)
        print("CloseButton clicked.")
Redbutton.on_click(on_Redbutton_clicked)
Greenbutton.on_click(on_Greenbutton_clicked)
Bluebutton.on_click(on_Bluebutton_clicked)
Yellowbutton.on_click(on_Yellowbutton_clicked)
Orangebutton.on_click(on_Orangebutton_clicked)
Closebutton.on_click(on_Closebutton_clicked)
```

```
import PID

xservo_pid = PID.PositionalPID(0.8, 0.2, 0.01)#1.1 0.2 0.8
yservo_pid = PID.PositionalPID(0.8, 0.6, 0.01)
```

```
# 定义 target_servox 和 target_servoy 在外部 Define target_servox and target_servoy
externally
target_servox = 90
target_servoy = 25
def servo_reset():
    bot.Ctrl_Servo(1,90)
    bot.Ctrl_Servo(2,25)
servo_reset()
```

No dead zone control, high real-time performance, the servo will always work, frequent jitter

```
def Color_Recongnize():
    oled.init_oled_process() #初始化oled进程 Initialize oled process
    global color_lower, color_upper, g_mode, first_read, while_cnt
    global target_valuex, target_valuey, color_x, color_y,target_servox
    t_start = time.time()
    fps = 0
    ret, frame = image.read()#USB摄像头 USB Camera
    # frame = picam2.capture_array() #CSI摄像头 CSI Camera
    #frame = cv2.resize(frame, (300, 300))
    frame = cv2.GaussianBlur(frame,(5,5),0)
    first_read = 1
    while_cnt = 0
    target_servox_x=0
    time.sleep(1)
    while True:
        ret, frame = image.read()
        #frame = picam2.capture_array() #CSI摄像头 CSI Camera
        #frame = cv2.resize(frame, (300, 300))
        #frame = cv2.GaussianBlur(frame,(3,3),0)
        hsv = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(hsv,color_lower,color_upper)
        mask = cv2.erode(mask,None,iterations=2)
        mask = cv2.dilate(mask,None,iterations=2)
        mask = cv2.GaussianBlur(mask,(5,5),0)
        cnts = 
cv2.findContours(mask.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)[-2]
        if g_mode == 1: # 按钮切换开关 Push button switch
            if len(cnts) > 0:
                cnt = max (cnts, key = cv2.contourArea)
                (color_x,color_y),color_radius = cv2.minEnclosingCircle(cnt)

                if color_radius > 10:
                    # 将检测到的颜色用原形线圈标记出来 Mark the detected color with a 
prototype circle
                    cv2.circle(frame,
(int(color_x),int(color_y)),int(color_radius),(255,0,255),2)
                    xservo_pid.SystemOutput = color_x
                    xservo_pid.SetStepSignal(250)
                    xservo_pid.SetInertiaTime(0.01, 0.1)
                    #print(xservo_pid.SystemOutput)
                    target_valuex = int(1600+xservo_pid.SystemOutput)
                    target_servox = int((target_valuex-500)/10)
                    # if(xservo_pid.SystemOutput<800):
                    target_servox_x = target_servox
                    # print("color_x ", color_x)
                    # print("target_servox ", target_servox)
                    # 将云台转动至PID调校位置 Turn the gimbal to the PID adjustment 
position
                    if target_servox_x > 180:
                        target_servox_x = 180
                    if target_servox_x < 0:
                        target_servox_x = 0

                    # 输入Y轴方向参数PID控制输入 Input Y-axis direction parameter 
PID control input
                    yservo_pid.SystemOutput = color_y
                    yservo_pid.SetStepSignal(200)
                    yservo_pid.SetInertiaTime(0.01, 0.1)
                    target_valuey = int(1150+yservo_pid.SystemOutput)
                    target_servoy = int((target_valuey-500)/10)
                    # print("color_y ", color_y)
                    # print("target_servoy ", target_servoy)
                    if target_servoy > 110:
                        target_servoy = 110
                    if target_servoy < 0:
                        target_servoy = 0
                    #print(target_servoy)

                    bot.Ctrl_Servo(1,target_servox_x)
                    bot.Ctrl_Servo(2,target_servoy)


        fps = fps + 1
        mfps = fps / (time.time() - t_start)
        """
        cv2.putText(frame, "FPS " + str(int(mfps)), (40,40), 
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 3)
```

```
        cv2.putText(frame, "x"+str(int(color_x)), (40,80), 
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 3)
        cv2.putText(frame, "servox"+str(target_servox_x), (40,120), 
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 3)
        cv2.putText(frame, "SystemOutput"+str(xservo_pid.SystemOutput), (40,160), 
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 3)
        cv2.putText(frame, "y"+str(int(color_y)), (300,80), 
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 3)
        cv2.putText(frame, "servoy"+str(target_servoy), (300,120), 
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 3)
        cv2.putText(frame, "1" , (160,120), cv2.FONT_HERSHEY_SIMPLEX, 0.8, 
(0,255,255), 3)
        """
        # 实时传回图像数据进行显示 Real-time image data transmission for display
        image_widget.value = bgr8_to_jpeg(frame)
```

With dead zone control, the real-time following performance is poor, the servo does not move within the dead zone, and the jitter is relatively stable

```
def Color_Recongnize2():
    oled.init_oled_process() #初始化oled进程 Initialize oled process
    oled.clear(refresh=True)
    global color_lower, color_upper, g_mode, first_read, while_cnt
    global target_valuex, target_valuey, color_x, target_servox
    t_start = time.time()
    fps = 0
    ret, frame = image.read()
    #frame = picam2.capture_array() #CSI摄像头 CSI Camera
    #frame = cv2.resize(frame, (300, 300))
    frame = cv2.GaussianBlur(frame,(5,5),0)
    first_read = 1
    while_cnt = 0
    time.sleep(1)
    while True:
        ret, frame = image.read()
        #frame = picam2.capture_array() #CSI摄像头  CSI Camera
        #frame = cv2.resize(frame, (300, 300))
        #frame = cv2.GaussianBlur(frame,(3,3),0)
        hsv = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
        mask = cv2.inRange(hsv,color_lower,color_upper)
        mask = cv2.erode(mask,None,iterations=2)
        mask = cv2.dilate(mask,None,iterations=2)
        mask = cv2.GaussianBlur(mask,(5,5),0)
        cnts = 
cv2.findContours(mask.copy(),cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)[-2]
        if g_mode == 1: # 按钮切换开关 Push button switch
            if len(cnts) > 0:
                cnt = max (cnts, key = cv2.contourArea)
                (color_x,color_y),color_radius = cv2.minEnclosingCircle(cnt)

                if color_radius > 10:
                    # 将检测到的颜色用原形线圈标记出来 Mark the detected color with a 
prototype circle
```

```
                    cv2.circle(frame,
(int(color_x),int(color_y)),int(color_radius),(255,0,255),2)
                    if math.fabs(180 - color_x) > 20:#调试方块半径  Debug Block
Radius
                        xservo_pid.SystemOutput = color_x
                        xservo_pid.SetStepSignal(250)
                        xservo_pid.SetInertiaTime(0.01, 0.05)
                        target_valuex = int(1600+xservo_pid.SystemOutput)
                        target_servox = int((target_valuex-500)/10)
                        #print("color_x %f", color_x)
                        #print("target_servox %d", target_servox)
                        # 将云台转动至PID调校位置 Turn the gimbal to the PID
adjustment position
                        if target_servox > 180:
                            target_servox = 180
                        if target_servox < 0:
                            target_servox = 0
                        bot.Ctrl_Servo(1,target_servox)

                    if math.fabs(180 - color_y) > 75:#调试方块半径    Debug Block
Radius
                        # 输入Y轴方向参数PID控制输入 Input Y-axis direction
parameter PID control input
                        yservo_pid.SystemOutput = color_y
                        yservo_pid.SetStepSignal(200)
                        yservo_pid.SetInertiaTime(0.01, 0.1)
                        target_valuey = int(1150+yservo_pid.SystemOutput)
                        target_servoy = int((target_valuey-500)/10)
                        #print("target_servoy %d", target_servoy)
                        if target_servoy > 100:
                            target_servoy = 100
                        if target_servoy < 0:
                            target_servoy = 0
                        #print(target_servoy)
                        bot.Ctrl_Servo(2,target_servoy)



        fps = fps + 1
        mfps = fps / (time.time() - t_start)
        cv2.putText(frame, "FPS " + str(int(mfps)), (40,40),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,255), 3)
        # 实时传回图像数据进行显示 Real-time image data transmission for display
        image_widget.value = bgr8_to_jpeg(frame)
```

```
display(image_widget)
display(Redbutton, Greenbutton, Bluebutton, Yellowbutton, Orangebutton,
Closebutton, output)
#thread1 = threading.Thread(target=Color_Recongnize)
thread1 = threading.Thread(target=Color_Recongnize2)

thread1.daemon=True
thread1.start()
```

```
stop_thread(thread1)
servo_reset()
# 恢复屏幕基础数据显示 Restore basic data display on screen
os.system("python3 /home/pi/software/oled_yahboom/yahboom_oled.py &")
```

```
#Release resources
image.release()
#picam2.stop_preview()#使用完成对象记住释放掉对象，不然下一个程序使用这个对象模块会被占用，
导致无法使用
#picam2.close()
```

There are two recognition functions in the program. Color_Recongnize() is a dead zone free control with high real-time performance, and the servo will always work. Color_Recongnize2 is a dead zone control with poor real-time performance, and the servo will not move within the dead zone. The purpose of using the dead zone is that the control accuracy of the plastic servo itself is not high, and there may be angle fluctuations. For example, it is originally controlled at 90°, but the servo itself will jump around 90°. By setting the dead zone of recognition, the servo stops working within the range where the recognition image deviates slightly from the center of the image to reduce the impact of this situation. However, this method will result in poorer followability. If you need good results such as accuracy and followability, it is recommended to use a servo with better performance and accuracy.

The startup of the two recognition functions can be modified in the code. Just comment out the ones that are not used.

```
display(image_widget)
display(Redbutton, Greenbutton, Bluebutton, Yellowbutton, Orangebutton,
Closebutton, output)
#thread1 = threading.Thread(target=Color_Recongnize)
thread1 = threading.Thread(target=Color_Recongnize2)

thread1.daemon=True
thread1.start()
```

1.3. Experimental Phenomenon

After the code block is run, we can control the car's servo gimbal to track different colors through buttons. After selection, the camera will rotate with the colored object. At the same time, the RGB light bar will light up the recognized color, and the OLED will display the recognized color. Because the camera is greatly affected by light. If the effect is not good, you need to adjust the HSV value of the color recognized in the code.