Color Block Tracking Experiment

This tutorial is an advanced gameplay of the color block positioning experiment, adding a robotic arm tracking function.

Overview: Target tracking includes two gameplays: color tracking and color picking tracking (learning tracking). The principle is to process the image through the camera, identify the target in a specific way, obtain the coordinate position of the target under the camera, calculate the deviation value of the target center point from the image center point, debug through the PID algorithm, and drive the robotic arm to move. Make the target center point coincide with the image center point.

Code path:

```
~/dofbot_pro/dofbot_color_follow/scripts/Color_follow.ipynb
```

1. Control design

- Import header files

```
import cv2 as cv
import threading
import random
from time import sleep
import ipywidgets as widgets
from IPython.display import display
from color_follow import color_follow
from dofbot_utils.fps import FPS
from dofbot_utils.robot_controller import Robot_Controller
from dofbot_utils.dofbot_config import *
```

- Create an instance and initialize parameters

```
robot = Robot_Controller()
robot.move_init_pose()
fps = FPS()
follow = color_follow()
model = 'General'
HSV_learning = ((0, 240, 54), (8, 255, 255))
# HSV_learning = ()
color_hsv = {"red": ((0, 25, 90), (10, 255, 255)),
             "green": ((53, 36, 40), (80, 255, 255)),
             "blue": ((110, 80, 90), (120, 255, 255)),
             "yellow": ((25, 20, 55), (50, 255, 255))}
color = [[random.randint(0, 255) for _ in range(3)] for _ in range(255)]
HSV_path="/home/jetson/dofbot_pro/dofbot_color_follow/scripts/HSV_config.txt"
try: read_HSV(HSV_path,color_hsv)
except Exception: print("Read HSV_config Error !!!")
```

- Create controls

```

```

```
button_layout = widgets.Layout(width='200px', height='100px',
align_self='center')
# 输出控件 Output widget
output = widgets.Output()
# 颜色追踪 Color tracking
color_follow = widgets.Button(description='color_follow', button_style='success',
layout=button_layout)
# 选择颜色 Select color
choose_color = widgets.ToggleButtons(options=['red', 'green', 'blue', 'yellow'],
button_style='success',
             tooltips=['Description of slow', 'Description of regular',
'Description of fast'])
# 取消追踪 Cancel tracking
follow_cancel = widgets.Button(description='follow_cancel',
button_style='danger', layout=button_layout)
# 学习颜色 Learn color
learning_color = widgets.Button(description='learning_color',
button_style='primary', layout=button_layout)
# 学习颜色追踪 Learn color tracking
learning_follow = widgets.Button(description='learning_follow',
button_style='success', layout=button_layout)
# 退出 exit
exit_button = widgets.Button(description='Exit', button_style='danger',
layout=button_layout)
# 图像控件 Image widget
imgbox = widgets.Image(format='jpg', height=480, width=640,
layout=widgets.Layout(align_self='auto'))
# 垂直布局 Vertical layout
img_box = widgets.VBox([imgbox, choose_color],
layout=widgets.Layout(align_self='auto'))
# 垂直布局 Vertical layout
Slider_box = widgets.VBox([color_follow, learning_color,
learning_follow,follow_cancel,exit_button],
                          layout=widgets.Layout(align_self='auto'))
# 水平布局 Horizontal layout
controls_box = widgets.HBox([img_box, Slider_box],
layout=widgets.Layout(align_self='auto'))
# ['auto', 'flex-start', 'flex-end', 'center', 'baseline', 'stretch', 'inherit',
'initial', 'unset']
```

- Mode switching

```
def color_follow_Callback(value):
    global model
    model = 'color_follow'
def learning_color_Callback(value):
    global model
    model = 'learning_color'
def learning_follow_Callback(value):
    global model
    model = 'learning_follow'
def follow_cancel_Callback(value):
    global model
    model = 'General'
    robot.move_init_pose()
```

```
def exit_button_Callback(value):
    global model
    model = 'Exit'
color_follow.on_click(color_follow_Callback)
learning_color.on_click(learning_color_Callback)
learning_follow.on_click(learning_follow_Callback)
follow_cancel.on_click(follow_cancel_Callback)
exit_button.on_click(exit_button_Callback)
```

- Main program

```
def camera():
    global HSV_learning,model
    # 打开摄像头 Open camera
    capture = cv.VideoCapture(0, cv.CAP_V4L2)
    capture.set(3, 640)
    capture.set(4, 480)
    capture.set(5, 30)
    # Be executed in loop when the camera is opened normally
    # 当摄像头正常打开的情况下循环执行
    while capture.isOpened():
        try:
            _, img = capture.read()
            fps.update_fps()
            if model == 'color_follow':
                img = follow.follow_function(img, color_hsv[choose_color.value])
                cv.putText(img, choose_color.value, (int(img.shape[0] / 2), 50),
cv.FONT_HERSHEY_SIMPLEX, 2, color[random.randint(0, 254)], 2)
            if model == 'learning_color':
                img,HSV_learning = follow.get_hsv(img)
            if model == 'learning_follow' :
                if len(HSV_learning)!=0:
                    print("HSV_learning", HSV_learning)
                    img = follow.learning_follow(img, HSV_learning)
                    cv.putText(img,'LeColor', (240, 50), cv.FONT_HERSHEY_SIMPLEX,
1, color[random.randint(0, 254)], 1)
            if model == 'Exit':
                cv.destroyAllWindows()
                capture.release()
                break
            fps.show_fps(img)
            imgbox.value = cv.imencode('.jpg', img)[1].tobytes()
        except KeyboardInterrupt:capture.release()
```

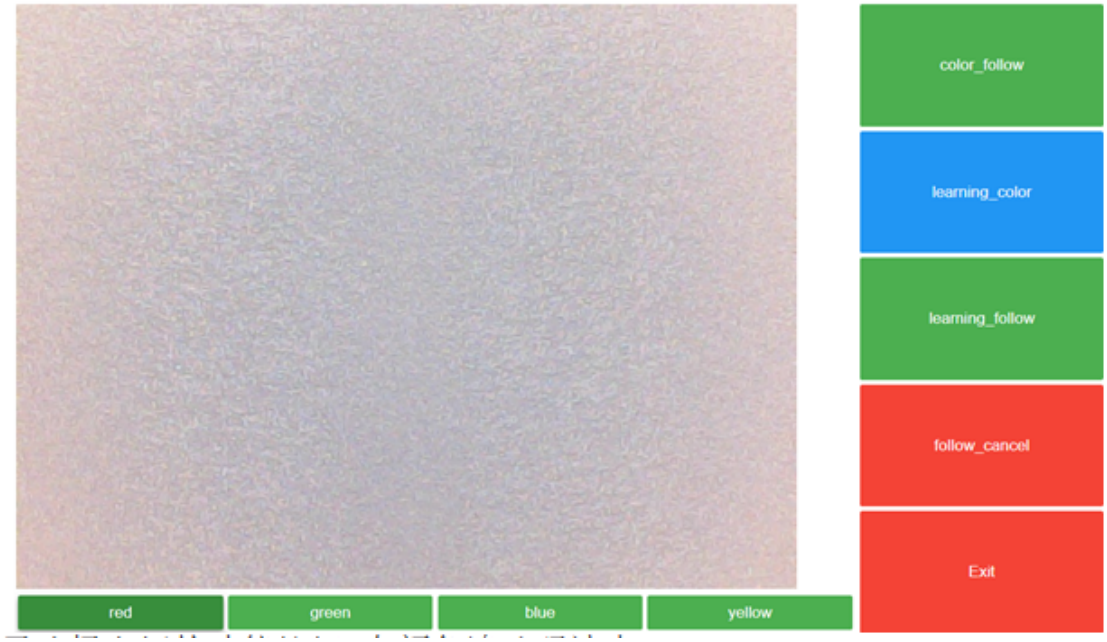
- start up

```
display(controls_box,output)
threading.Thread(target=camera, ).start()
```

2. Running programSequence

Click the Run Entire Program button on the Jupyterlab toolbar, and then pull it to the bottom.

- **Color Tracking**



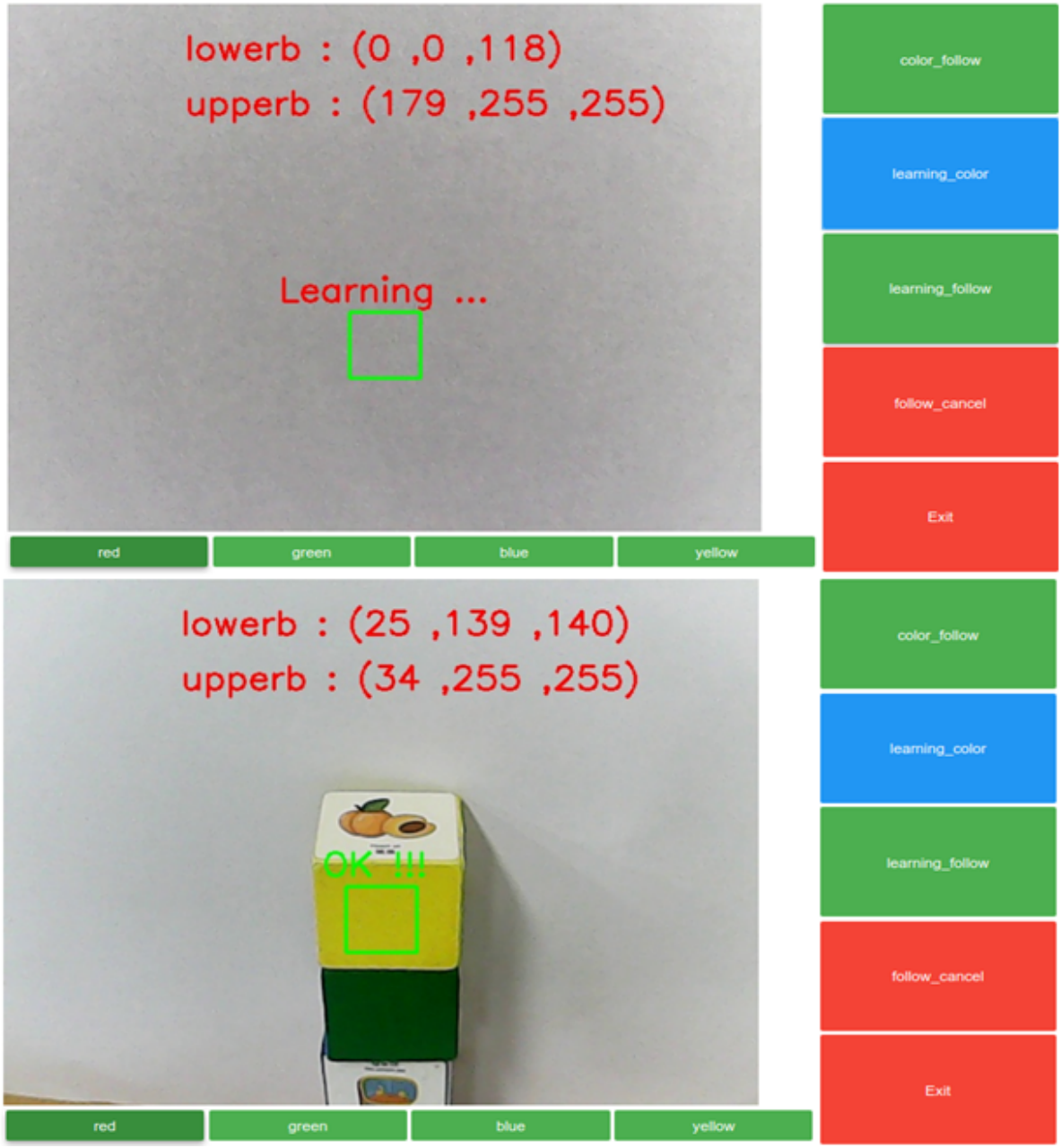
The gameplay is not tracked by default, and the corresponding function button needs to be selected.

Start the code block, click the [color_follow] button to turn on color tracking, and the color can be switched below the image. Put the block in the field of view, wait for the camera to recognize it, and the robot arm tracks the movement of the block, facing the center of the block.

Click the [follow_cancel] button to cancel tracking.

Click the [Exit] button to exit the program.

- **Color tracking**



(1) Start the code block and click the [leraning_color] button. A box will appear in the center of the screen, as shown in the left picture.

(2) Place an object in the box and print the high and low thresholds of HSV in real time. When the words [OK!!!] appear below the box, it means that the recognition is successful.

(3) Click the [leraning_follow] button. As long as the outline of the learning color can be detected, real-time tracking can be performed.

(4) Click the [follow_cancel] button to cancel tracking.

Click the [Exit] button to exit the program.