Target Tracking

This tutorial is an advanced way of playing the color block positioning experiment, adding a mechanical arm following function.

Overview: Target tracking includes three gameplay methods: color tracking, color tracking (learning tracking), and face tracking. The principle is to perform certain processing on 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, and drive the robotic arm through PID algorithm debugging. move. Make the target center point coincide with the image center point.

```
How to play:
1) First close the big program
2) Color tracking gameplay and color tracking (learning tracking) gameplay paths:
/root/dofbot_ws/src/dofbot_color_follow/Color tracking and learning
tracking.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
```

- Initialize the robot arm position

```
import Arm_Lib
Arm = Arm_Lib.Arm_Device()
joints_0 = [90, 135, 20, 25, 90, 30]
Arm.Arm_serial_servo_write6_array(joints_0, 1000)
```

- Create an instance and initialize parameters

```
follow = color_follow()
#Initialization mode
model = 'General'
#Initialize HSV_learning value
HSV_learning = ()
#Initialize HSV value
color_hsv = {"red" : ((0, 43, 46), (10, 255, 255)),
         "green" : ((35, 43, 46), (77, 255, 255)),
         "blue" : ((100, 43, 46), (124, 255, 255)),
         "yellow": ((26, 43, 46), (34, 255, 255))}
# Set random color
color = [[random.randint(0, 255) for _ in range(3)] for _ in range(255)]
```

- Create controls

```
button_layout = widgets.Layout(width='200px', height='100px',align_self='center')
# Output control
output = widgets.Output()
# Color Tracking
color_follow = widgets.Button(description='color_follow', button_style='success',
layout=button_layout)
# choose the color
choose_color = widgets.ToggleButtons(options=['red', 'green', 'blue', 'yellow'],
button_style='success',tooltips=['Description of slow', 'Description of regular',
'Description of fast'])
# Cancel tracking
follow_cancel = widgets.Button(description='follow_cancel',
button_style='danger', layout=button_layout)
# learn colors
learning_color = widgets.Button(description='learning_color',
button_style='primary', layout=button_layout)
#Learn color tracking
learning_follow = widgets.Button(description='learning_follow',
button_style='success', layout=button_layout)
# quit
exit_button = widgets.Button(description='Exit', button_style='danger',
layout=button_layout)
#Image control
imgbox = widgets.Image(format='jpg', height=480, width=640,
layout=widgets.Layout(align_self='auto'))
# Vertical layout
img_box = widgets.VBox([imgbox, choose_color],
layout=widgets.Layout(align_self='auto'))
# Vertical layout
Slider_box = widgets.VBox([color_follow, learning_color,
learning_follow,follow_cancel,exit_button],layout=widgets.Layout(align_self='auto
'))
#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'

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

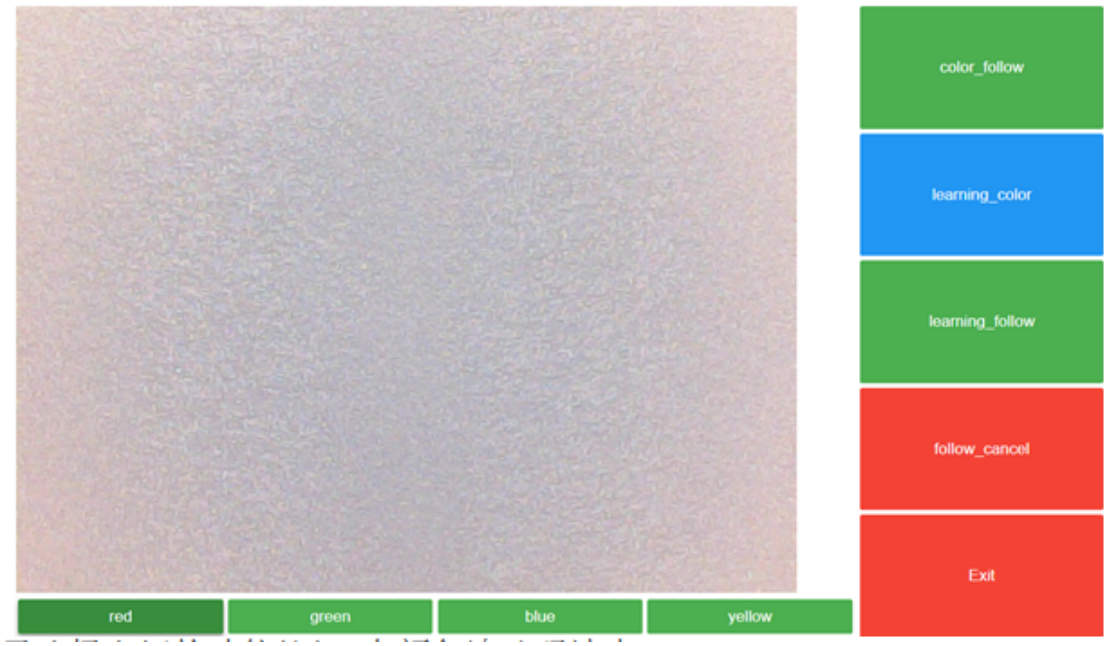
```
def camera():
     global HSV_learning,model
     #Open camera
     capture = cv.VideoCapture(0)
     capture.set(3, 640)
     capture.set(4, 480)
     capture.set(5, 30) #Set frame rate
     # Loop execution when the camera is turned on normally
     while capture.isOpened():
         try:
             # Read each frame of the camera
             _, img = capture.read()
             # Unify image size
             img = cv.resize(img, (640, 480))
             if model == 'color_follow':
                 img = follow.follow_function(img,color_hsv[choose_color.value])
                 # Add text

                 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':
                 img = follow.learning_follow(img, HSV_learning)
                 # Add text
                 cv.putText(img,'LeColor', (200, 50), cv.FONT_HERSHEY_SIMPLEX, 1,
color[random.randint(0, 254)], 1)
             if model == 'Exit':
                 cv.destroyAllWindows()
                 capture.release()
                break

             imgbox.value = cv.imencode('.jpg', img)[1].tobytes()
         except KeyboardInterrupt:capture.release()
```

- start up

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

The adjustment of the control algorithm PID is discussed in the PID algorithm section. If you need to adjust, please check the PID algorithm to learn. For the library code of color tracking and color tracking, please see /root/dofbot_ws/src/dofbot_color_follow/color_follow.py

2. Color tracking

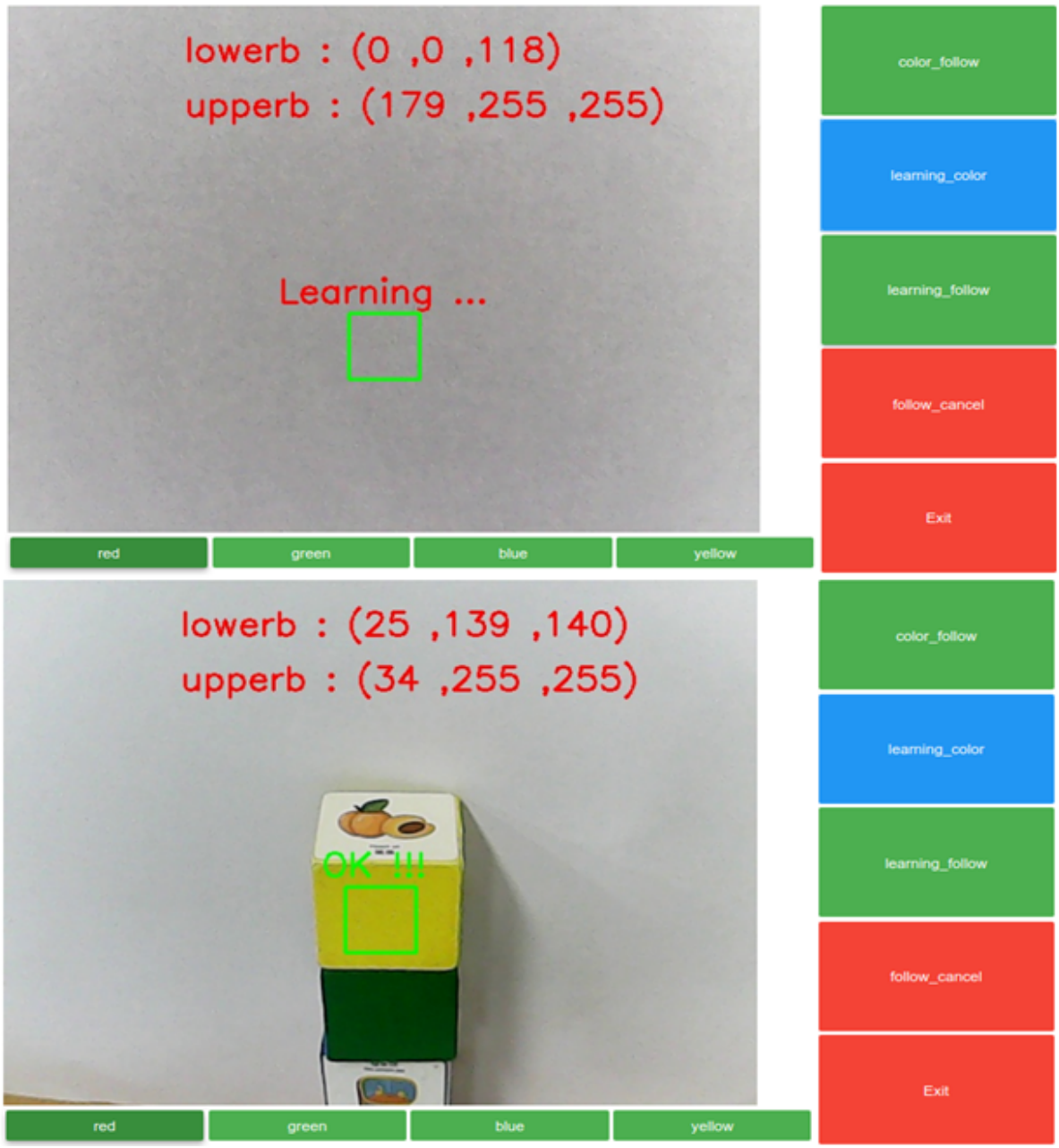


The gameplay does not track by default. You need to select the function button on the right. In the color tracking gameplay,

Start the code block, click the [color_follow] button to turn on color tracking, and the color can be switched below the image. Place the block under the field of view, wait for the camera to recognize it, and the robot arm will track the movement of the block and face the center of the block.

Click the [follow_cancel] button to cancel tracking without exiting the program. Click the Exit button to exit the program.

3. Color tracking



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

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

(3) Click the [leraning_follow] button, and as long as the outline of the learning color can be detected, it can be tracked in real time.

(4) Click the [follow_cancel] button to cancel tracking without exiting the program. Click the Exit button to exit the program.

4. Code analysis

- Get the range of HSV in a certain area

```
# Convert color image to HSV
HSV = cv.cvtColor(img, cv.COLOR_BGR2HSV)
# Draw a rectangular frame
cv.rectangle(img, (290, 190), (350, 250), (0, 255, 0), 2)
# Take out the H, S, and V values of each row and column in turn and put them
into the container.
for i in range(290, 350):
       for j in range(190, 250):
           H.append(HSV[j, i][0])
           S.append(HSV[j, i][1])
           V.append(HSV[j, i][2])
```

- Calculate the maximum and minimum values of H, S and V respectively

```
H_min = min(H);H_max = max(H)
S_min = min(S);S_max = max(S)
V_min = min(V);V_max = max(V)
```

- HSV range adjustment. There are certain differences in the HSV values of blocks at different positions in the image, so appropriate adjustments are required.

```
if H_max + 2 > 255:H_max = 255
else:H_max += 2
if H_min - 2 < 0:H_min = 0
else:H_min -= 2
if S_min-10<0:S_min=0
else:S_min -= 15
if V_min-10<0:V_min=0
else:V_min -= 15
S_max = 255; V_max = 255
```

- Format conversion, convert the calculated data into int format

```
hsv_range = ((int(H_min), int(S_min), int(V_min)),(int(H_max), int(S_max),
int(V_max)))
```

The following contour detection, PID control and tracking and color tracking are the same.