Tracking game

1. Functional description

The function of luring the snake out of the hole is based on the color recognition function, combined with the kinematics forward and inverse solutions to control the movement of the robot arm. After selecting a color, place the building block of the selected color in front of the robot arm camera, and the distance between the current robot arm and the building block will be calculated. According to the distance, the posture of the robot arm is adjusted to control the robot arm. The closer the building block is, the more the robot arm retreats, and the farther the building block is, the more the robot arm moves forward. When the robot arm reaches the farthest distance, the building block is clamped and placed in the corresponding color area.

The color recognition function uses HSV color recognition. The path where the HSV color calibration file is saved is ~/dofbot_ws/src/dofbot_snake_follow/scripts/HSV_config.txt. If the color recognition is not accurate enough, please open the ~/dofbot_ws/src/dofbot_snake_follow/scripts/HSV_calibration.ipynb file and recalibrate the HSV value of the block color. After the calibration operation is completed, it will be automatically saved to the HSV_config file. Rerun the program without additional code modification.

Note: Before starting the program, please follow the [Assembly and Assembly Tutorial] -> [Install Map] tutorial and install the map correctly before operating.

2. Code block design

- Import header file

```
import cv2 as cv
import threading
from time import sleep

import ipywidgets as widgets
from IPython.display import display
from snake_target import snake_target
from snake_ctrl import snake_ctrl
from dofbot_utils.dofbot_config import *
```

- Create an instance and initialize parameters

```
import Arm_Lib
Arm = Arm_Lib.Arm_Device()
joints_0 = [90, 135, 0, 45, 0, 180]
Arm.Arm_serial_servo_write6_array(joints_0, 1000)
```

- Create an instance and initialize parameters

```
snake_target = snake_target()
snake_ctrl = snake_ctrl()
model = 'General'
color = [[random.randint(0, 255) for _ in range(3)] for _ in range(255)]
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))}
HSV_path="/home/jetson/dofbot_ws/src/dofbot_snake_follow/scripts/HSV_config.txt"
try: read_HSV(HSV_path,color_hsv)
except Exception: print("Read HSV_config Error!!!")
```

- Creating Controls

```
button_layout      = widgets.Layout(width='150px', height='27px',
align_self='center')
output = widgets.Output()
choose_color=widgets.ToggleButtons( options=['red', 'green', 'blue','yellow'],
button_style='success',
    tooltips=['Description of slow', 'Description of regular', 'Description of
fast'])
# 退出 exit
exit_button = widgets.Button(description='Exit', button_style='danger',
layout=button_layout)
imgbox = widgets.Image(format='jpg', height=480, width=640,
layout=widgets.Layout(align_self='center'))
down_box = widgets.HBox([choose_color,exit_button],
layout=widgets.Layout(align_self='center'));
controls_box = widgets.VBox([imgbox, down_box],
layout=widgets.Layout(align_self='center'))
# ['auto', 'flex-start', 'flex-end', 'center', 'baseline', 'stretch', 'inherit',
'initial', 'unset']
```

- Switching Mode

```
def exit_button_Callback(value):
    global model
    model = 'Exit'
#     with output: print(model)
exit_button.on_click(exit_button_Callback)
```

- Main Program

```
def camera():
    # 打开摄像头 Open camera
    capture = cv.VideoCapture(0)
    capture.set(cv.CAP_PROP_FRAME_WIDTH, 640)
    capture.set(cv.CAP_PROP_FRAME_HEIGHT, 480)
    # 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()
```

```
            # 获得运动信息 Get motion information
            img, snake_msg = snake_target.target_run(img, color_hsv)
            if len(snake_msg) == 1:
                threading.Thread(target=snake_ctrl.snake_main, args=
(choose_color.value, snake_msg,)).start()
            if model == 'Exit':
                capture.release()
                break
            cv.putText(img, choose_color.value, (int(img.shape[0] / 2), 50),
cv.FONT_HERSHEY_SIMPLEX, 2, color[random.randint(0, 254)], 2)
            imgbox.value = cv.imencode('.jpg', img)[1].tobytes()
        except KeyboardInterrupt:capture.release()
```

- Start Display

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

3. Start the program

Start the ROS node service

Open the system terminal and enter the following command. If it is already started, you don't need to start it again.

```
sudo systemctl start yahboom_arm.service
```

Start the program

Open the jupyterlab webpage and find the corresponding .ipynb program file.

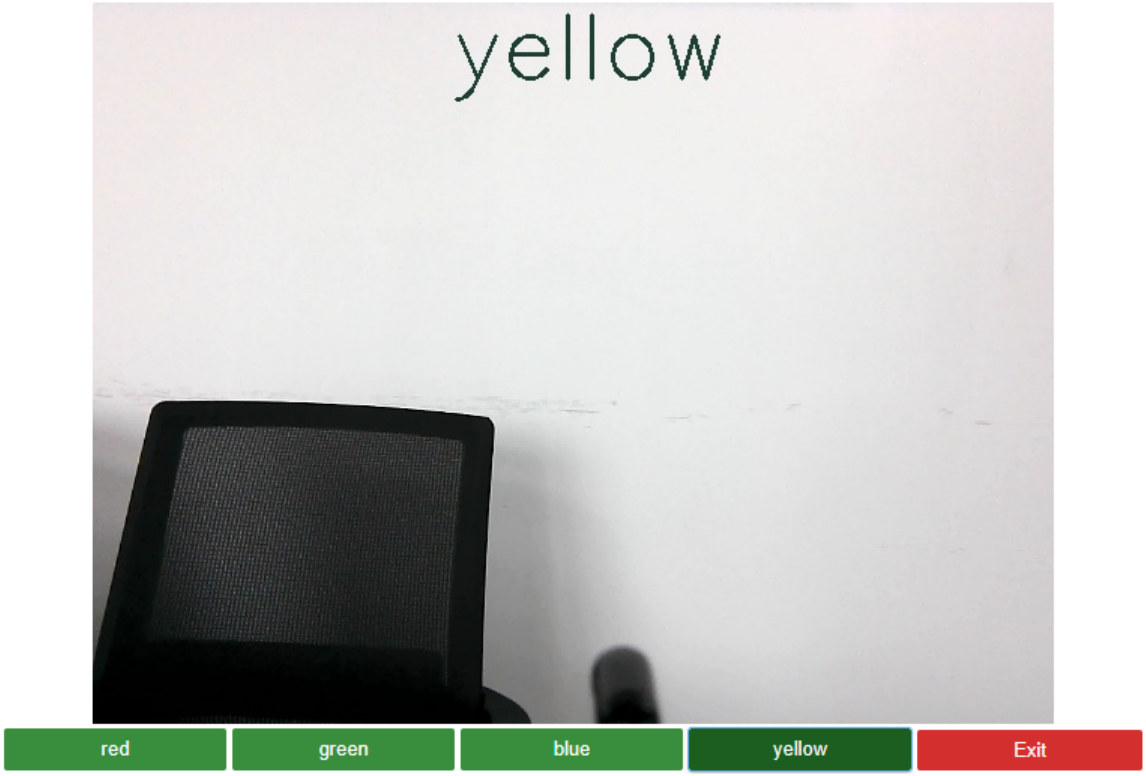
Code path:

```
~/dofbot_ws/src/dofbot_snake_follow/scripts/Snake_Follow.ipynb
```

Click the Run the entire program button on the jupyterlab toolbar and pull it to the bottom.

4. Experimental effect

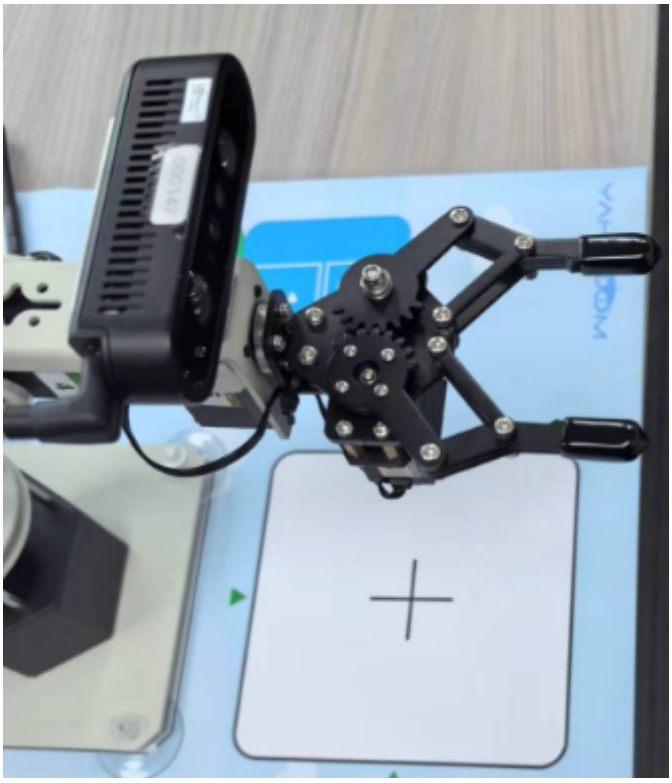
After the program runs, the interface shown in the figure is displayed. Click the color button to select a color at will. The selected color is displayed in the middle above the image. Here, yellow is taken as an example.



When the selected color appears in the field of view, the robot arm estimates the location of the block based on the area of the block in the image.

When the area becomes smaller, the robot arm drives forward, and when the area becomes larger, the robot arm will move backward.

When the robot arm reaches the farthest distance, the gripper opens, and the building block is placed in the gripper, the building block is picked up, and placed in the corresponding color area.



Before the next recognition operation, please remove the building blocks to avoid conflicts when placing them.

If you need to exit the program, please click the [Exit] button.