Tracking Game

Orin board users can directly open the webpage and enter IP address:8888 to access jupyter-lab and run directly. Jetson-Nano board users need to enter the docker container first, then enter the following command in docker:

```
cd
jupyter-lab --allow-root
```

Then open the webpage and enter IP address:9999 to access jupyter-lab and run the following program.

1. Function Description

The tracking game function is based on color recognition functionality, combined with forward and inverse kinematics to control robotic arm movement. After selecting a color, place the selected color building block in front of the robotic arm camera. It will calculate the current distance between the robotic arm and the building block, and adjust the robotic arm's posture according to the distance. The closer the building block, the more the robotic arm retreats; the farther the building block, the more the robotic arm follows forward. When reaching the maximum distance of the robotic arm posture, it will grab the building block and place it in the corresponding color area.

The color recognition function uses HSV color recognition. The HSV color calibration file is saved at ~/dofbot_pro/dofbot_snake_follow/scripts/HSV_config.txt. If color recognition is not accurate enough, please open the ~/dofbot_pro/dofbot_snake_follow/scripts/HSV_calibration.ipynb file to recalibrate the building block color HSV values. After calibration is completed, it will be automatically saved to the HSV_config file. Restart the program without needing to modify the code.

Note: Before starting the program, please follow the [A2. Assembly Course] -> [Install the Map] tutorial to correctly install the map before proceeding with operations.

2. Code Block Design

- Import header files

```
#!/usr/bin/env python
# coding: utf-8
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 instances, 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 instances, 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_pro/dofbot_snake_follow/scripts/HSV_config.txt"
try: read_HSV(HSV_path,color_hsv)
except Exception: print("Read HSV_config Error!!!")
```

- Create widgets

```
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'])
# 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']
```

- Mode switching

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

- Main program

```
def camera():
    # Open camera
    capture = cv.VideoCapture(0, cv.CAP_V4L2)
    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
    while capture.isOpened():
        try:
            _, img = capture.read()
            # 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()
```

- Startup display

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

3. Start the Program

Start ROS Node Service

Open the system terminal and enter the following command. If it's already running, there's no need to start it again.

```
ros2 run dofbot_pro_info kinemarics_dofbot
```

Start the Program

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

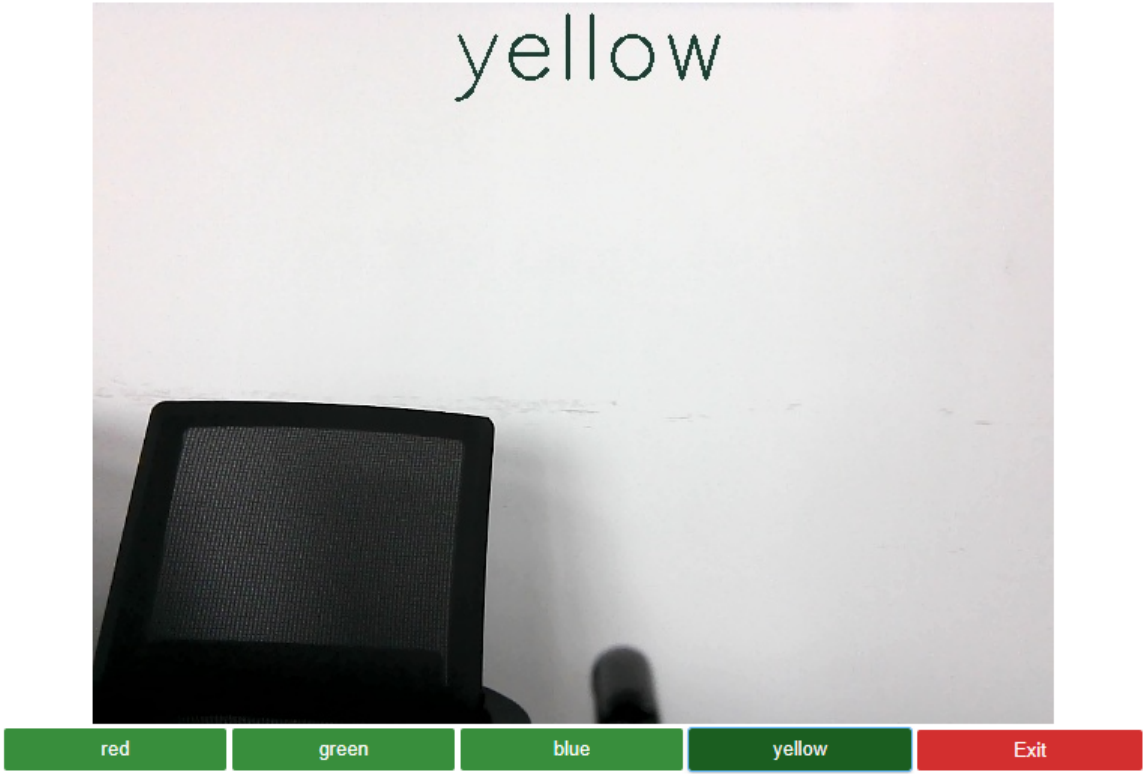
Code path:

```
#Jetson-Nano users need to enter the docker container to view
~/dofbot_pro/dofbot_snake_follow/scripts/Snake_Follow.ipynb
```

Click the "Run entire program" button on the jupyterlab toolbar, then scroll to the bottom.

4. Experimental Effects

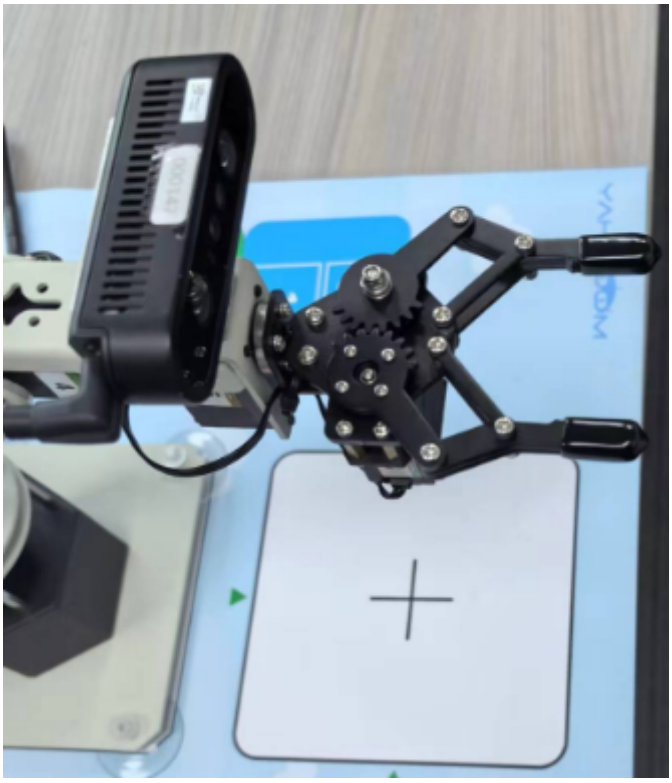
After the program runs, the interface shown in the figure will be displayed. Click the color button to select a color, and the selected color will be displayed in the center above the image. Here we use yellow as an example.



When the selected color appears in the field of view, the robotic arm will estimate the block's position based on its area in the image.

When the area becomes smaller, the robotic arm drives forward; when the area becomes larger, the robotic arm retreats.

When the robotic arm extends to the maximum distance, the gripper opens. At this time, place the building block into the gripper, grab the block, and place it in the corresponding color area.



Before performing the next recognition operation, please remove the building block to avoid conflicts during placement.

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