Gesture Stacking Color Blocks

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

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

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

1. Gesture Recognition Description

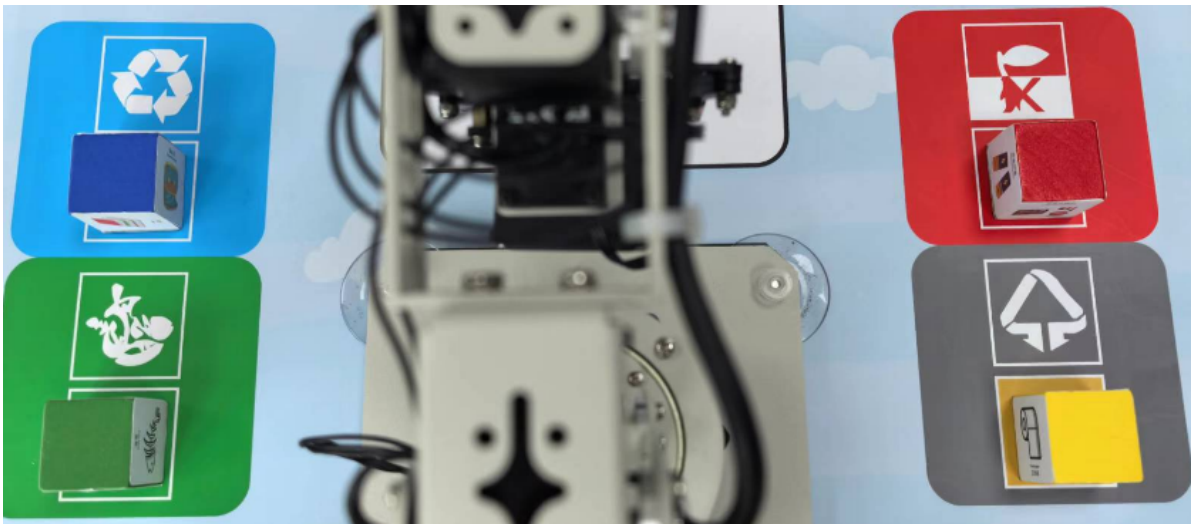
The gesture recognition stacking function is based on the gesture recognition grasping function, modifying the placement method to stacking functionality, and changing gesture 5 to knock down building blocks.

The gesture recognition grasping function mainly combines mediapipe gesture recognition functionality, robotic arm and map gameplay. It recognizes a total of five gestures, gestures 1-5, where gestures 1-4 represent the grabbing positions 1-4 on the map and stack the building blocks in sequence, while gesture 5 performs the function of knocking down building blocks and clears the recognition records.

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. Experimental Setup

Place the building blocks at positions numbered 1-4.



3. Code Block Design

- Import header files

```
import cv2 as cv
import threading
import time
import ipywidgets as widgets
from IPython.display import display
from gesture_stacking import Gesture_Stacking
from dofbot_utils.fps import FPS
from dofbot_utils.robot_controller import Robot_Controller
```

- Create instances, initialize parameters

```
robot = Robot_Controller()
robot.move_init_pose()
fps = FPS()

gesture = Gesture_Stacking()
model = 'General'
```

- Create widgets

```
def exit_button_Callback(value):
    global model
    model = 'Exit'
    with output:
        print(model)

exit_button.on_click(exit_button_Callback)
```

- Mode switching

```
def target_detection_Callback(value):
    global model,debug_pos
    model = 'Detection'
    with output: print(model)
    debug_pos = True
def grap_Callback(value):
    global model
    model = 'Grap'
    with output: print(model)
def exit_button_Callback(value):
    global model
    model = 'Exit'
    with output: print(model)
target_detection.on_click(target_detection_Callback)
grap.on_click(grap_Callback)
exit_button.on_click(exit_button_Callback)
```

- Main program

```
def camera():
    global model, gesture
    # 打开摄像头 Open camera
    capture = cv.VideoCapture(0, cv.CAP_V4L2)
    capture.set(3, 640)
```

```
    capture.set(4, 480)
    # Be executed in loop when the camera is opened normally
    while capture.isOpened():
        try:
            _, img = capture.read()
            fps.update_fps()
            gesture.process(img)
            if model == 'Exit':
                capture.release()
                del gesture
                break
            fps.show_fps(img)
            imgbox.value = cv.imencode('.jpg', img)[1].tobytes()
            time.sleep(0.002)
        except Exception as e:
            print("program end")
            print(e)
            capture.release()
```

- Startup display

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

4. Start the Program

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

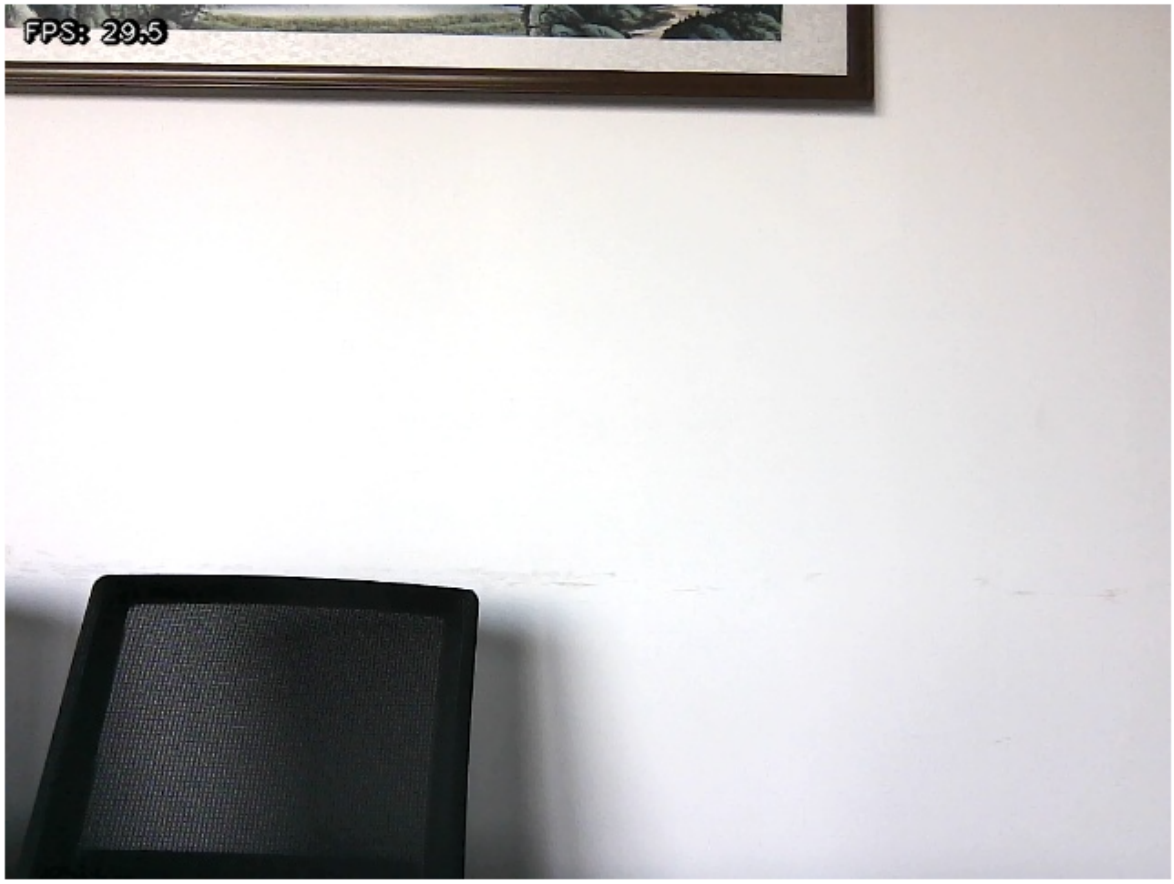
Code path:

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

Then click the run all command.

5. Experimental Effects

After the program runs, scroll to the bottom, and the jupyterlab webpage will display the camera view and related button functions.

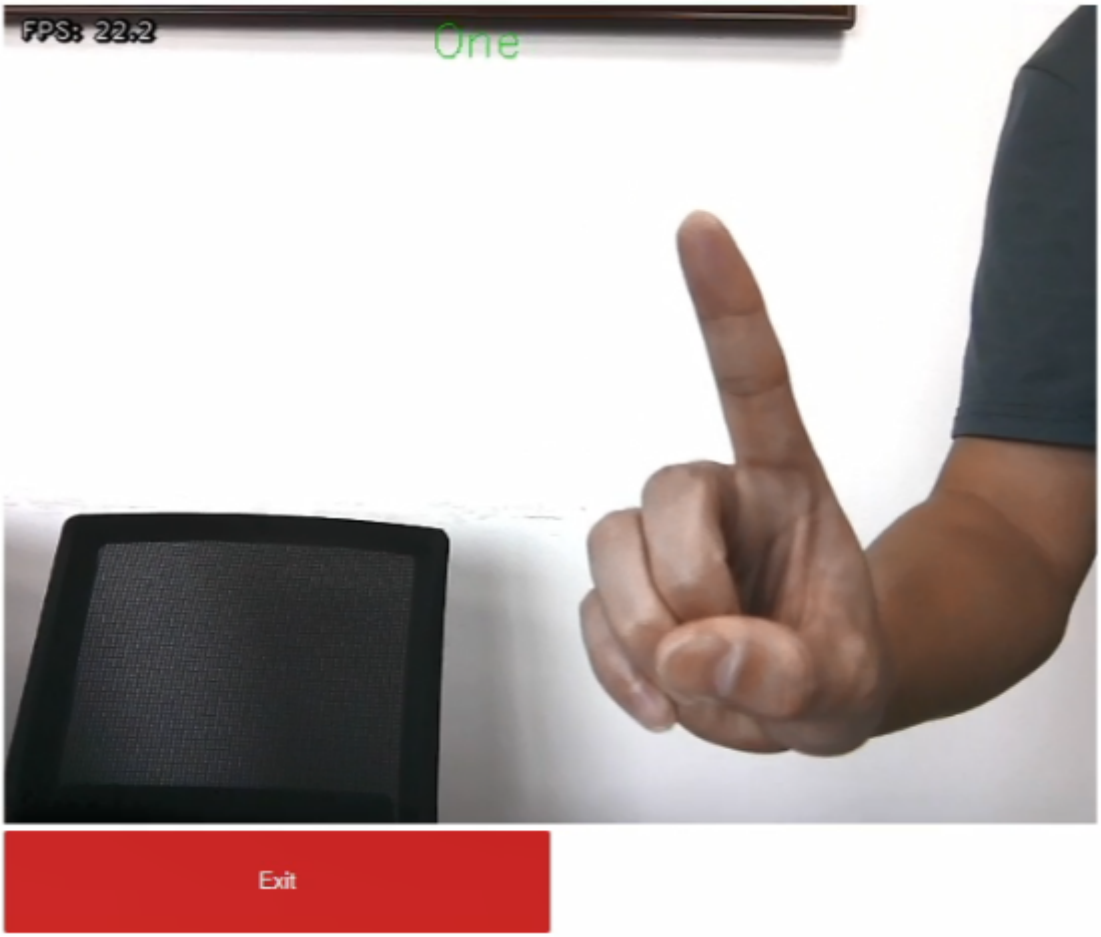


If a set gesture is recognized, the robotic arm will perform the corresponding action. When the robotic arm recognizes a number for the first time, it will go to the corresponding number position to grab the building block and place it on the first layer. When it recognizes a gesture number for the second time, it will go to the corresponding number position to grab the building block and place it on the second layer. When it recognizes a gesture number for the third time, it will go to the corresponding number position to grab the building block and place it on the third layer. When it recognizes a gesture number for the fourth time, it will go to the corresponding number position to grab the building block and place it on the fourth layer; each number can only be recognized once, and multiple recognitions will not execute the action; when gesture five is recognized, the robotic arm will knock down the building blocks, clear the records, and start over.

The gesture and action correspondence in this example is as follows:

Gesture	**Function**
Gesture 1	Grab building block from position 1 and stack it
Gesture 2	Grab building block from position 2 and stack it
Gesture 3	Grab building block from position 3 and stack it
Gesture 4	Grab building block from position 4 and stack it
Gesture 5	Knock down building blocks, clear records

As shown in the figure below:



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