Identify Grab Color Block

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. Function Overview

The color recognition and grabbing function mainly detects the HSV values of colors in the image frame, distinguishes the current building block color based on HSV values, then the robotic arm rotates to the corresponding color position on the map to grab the building block and place it at the center cross position.

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

Code path:

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

2. Code Block Design

- Import header files

```
import cv2 as cv
import threading
import ipywidgets as widgets
from IPython.display import display
from color_grab import Color_Grab
from dofbot_utils.fps import FPS
from dofbot_utils.robot_controller import Robot_Controller
```

- Create instances, initialize parameters

```
grab = Color_Grab()
robot = Robot_Controller()
robot.move_look_front()

fps = FPS()
model = 'Start'
```

- Create widgets

```
# 创建控件布局  Create widget layout
button_layout  = widgets.Layout(width='200px', height='70px',
align_self='center')
# 输出打印  Output printing
output = widgets.Output()
# 退出按钮  exit button
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='center'))
# 垂直放置  Vertical placement
controls_box = widgets.VBox([imgbox, exit_button],
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():
    # 打开摄像头 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)
    while capture.isOpened():
        try:
            if model == 'Exit':
                capture.release()
                break
            _, img = capture.read()
            fps.update_fps()
            # 获得运动信息  Get motion information
            img = grab.start_grab(img)
            fps.show_fps(img)
            imgbox.value = cv.imencode('.jpg', img)[1].tobytes()
        except KeyboardInterrupt:capture.release()
        except Exception as e:
            print(e)
```

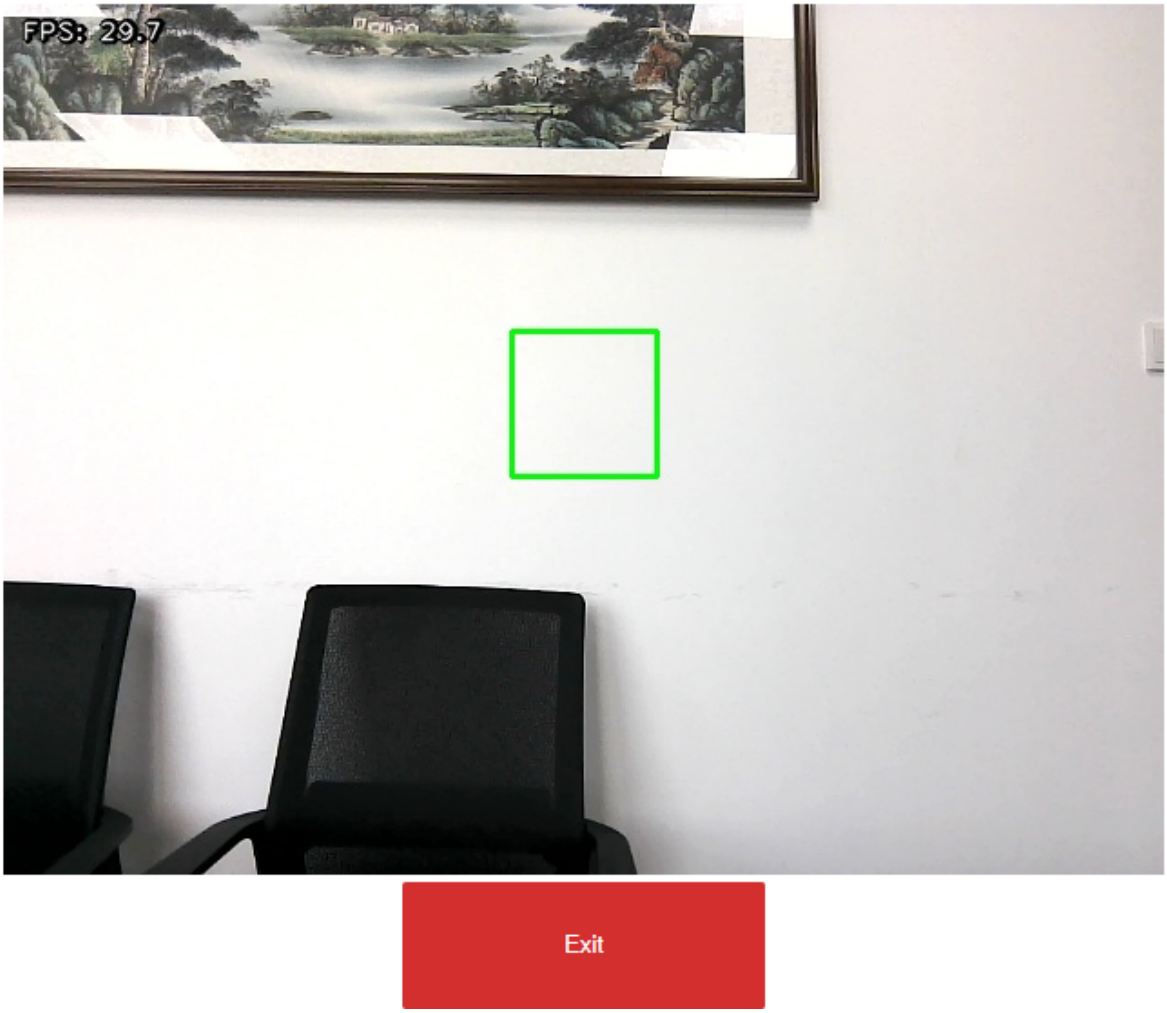
- Startup

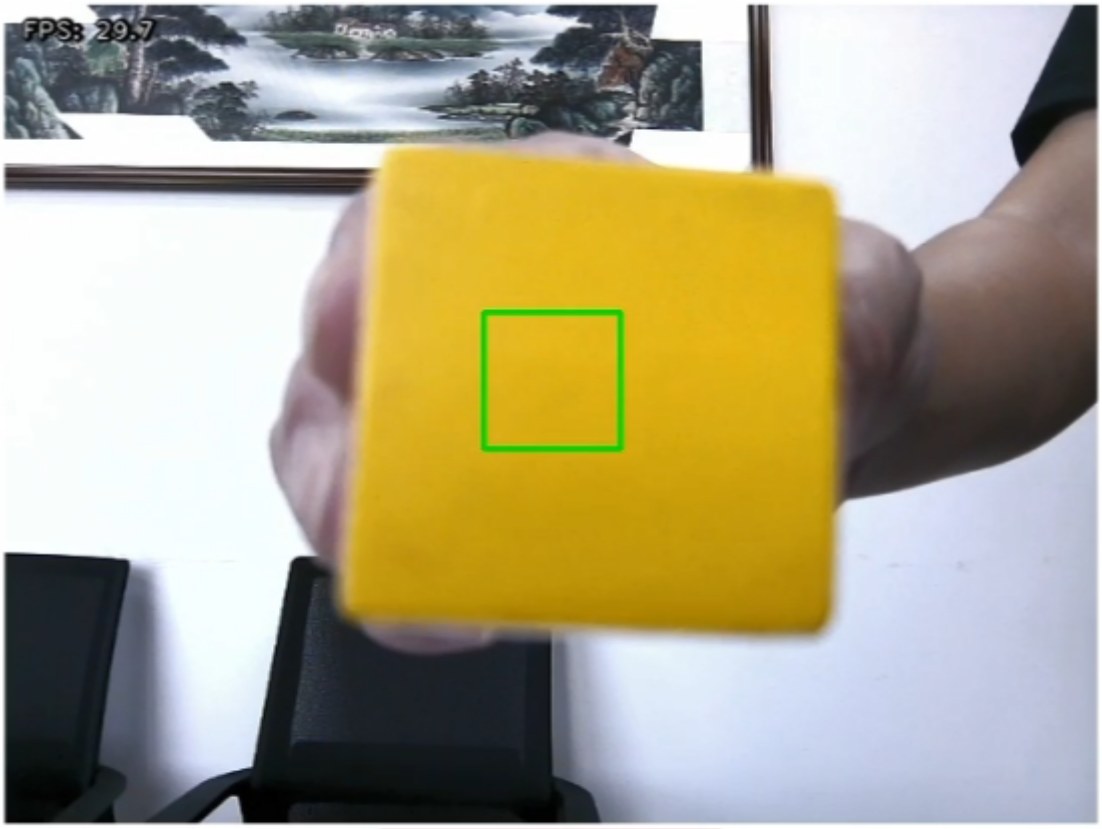
```
display(controls_box,output)
threading.Thread(target=camera, ).start()
```

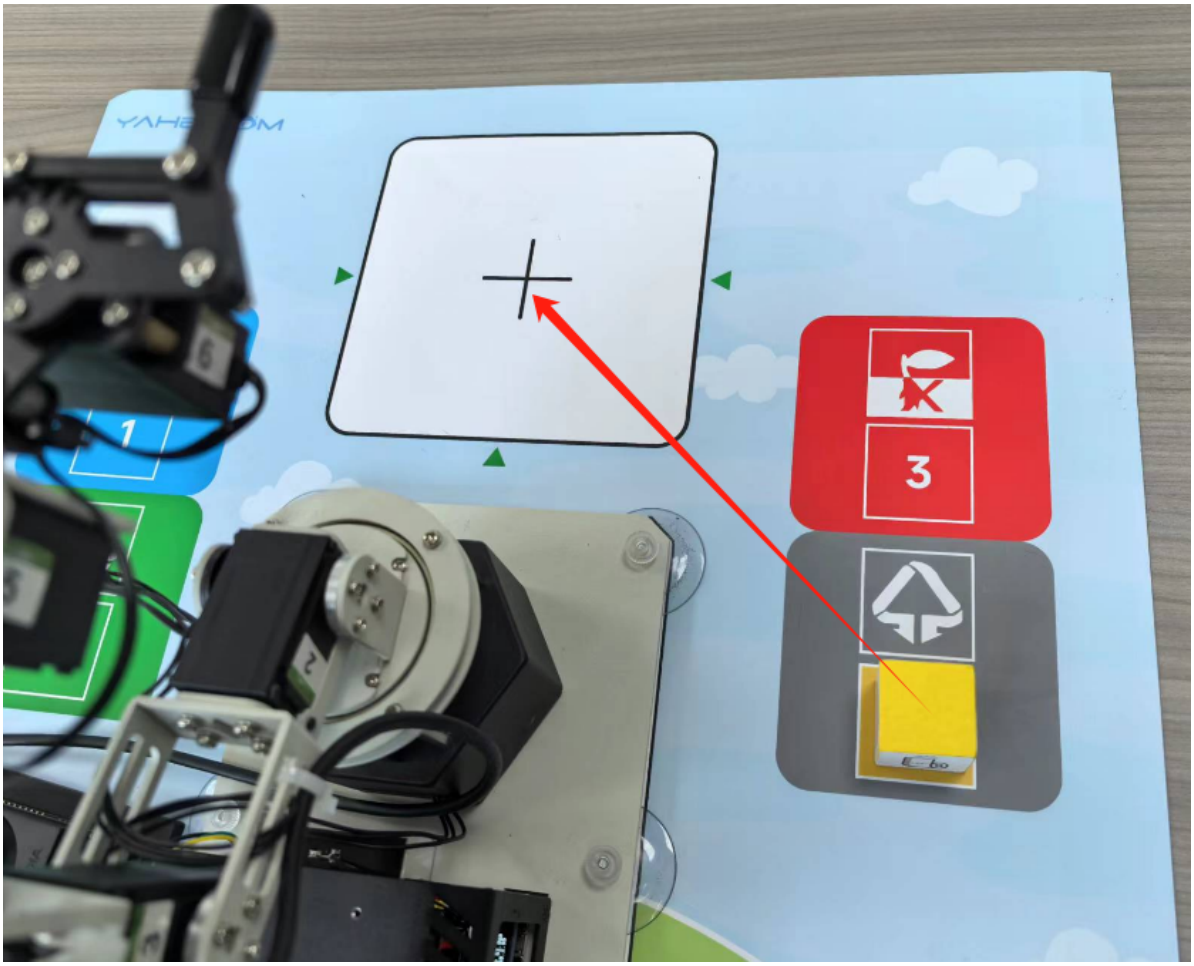
3. Run the Program

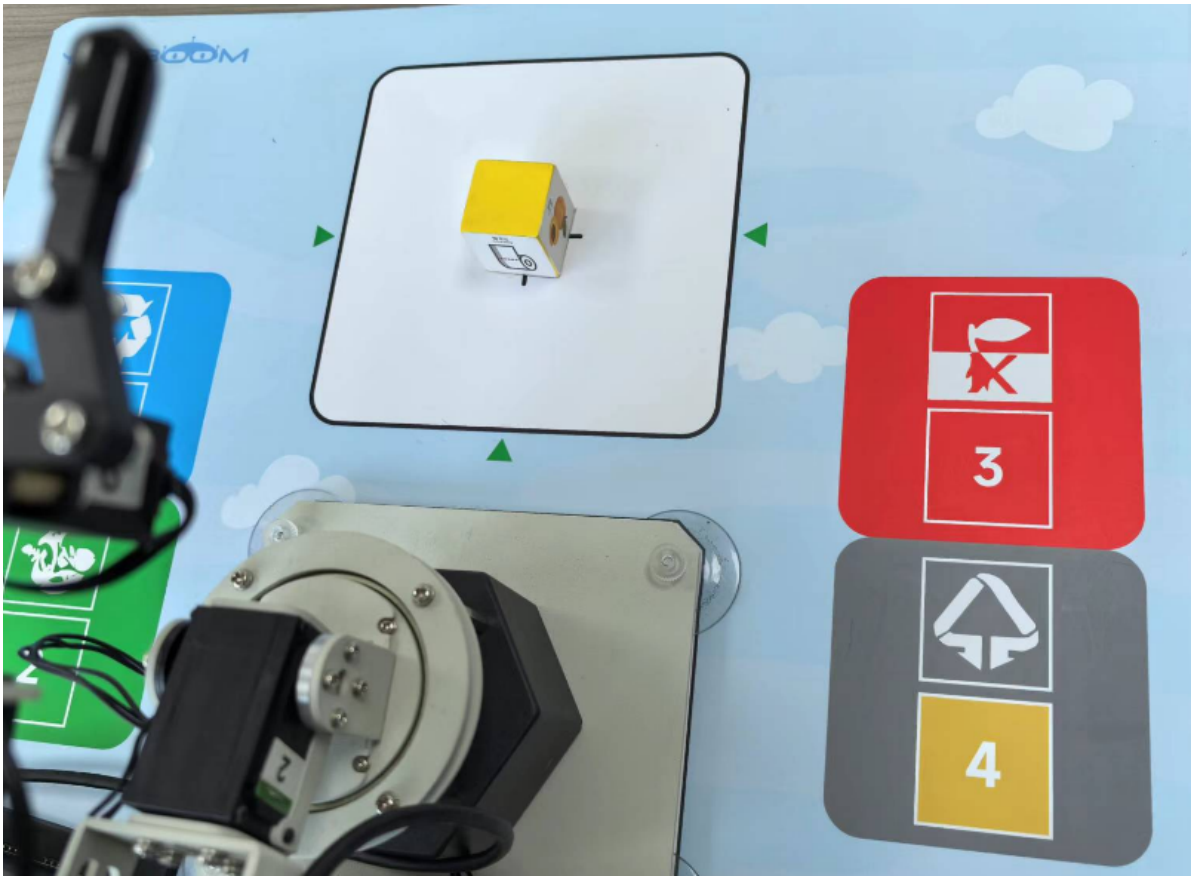
Click the run entire program button on the jupyterlab toolbar, then scroll to the bottom.

You can see a green frame in the center of the camera display. Place the color building block to be grabbed into the frame and wait for recognition to complete. When the buzzer beeps once, place the building block into the corresponding color area. The robotic arm will automatically go to the corresponding color area to grab the building block and place it on the center cross.




FPS: 29.7
Exit


1
3



Before starting the next recognition and grabbing, please remove the building block from the cross.

If you need to end the program, please click the [Exit] button to avoid affecting other programs from calling resources.