Color Block Positioning

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.

The function of color block positioning is to detect the position of color blocks by judging the HSV values of the camera image, frame the color blocks, and print the position coordinates of the color blocks.

1. Main Code

Code path:

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

- Import header files

```
import cv2 as cv
import threading
import random
import ipywidgets as widgets
from IPython.display import display
from color_position import Color_Position
from dofbot_utils.robot_controller import Robot_Controller
from dofbot_utils.fps import FPS
from dofbot_utils.dofbot_config import *
```

- Main recognition function, simultaneously obtains the target center point of the color block (color_x, color_y)

```
def process(self, img, HSV_config):
        (color_lower, color_upper) = HSV_config
        # self.img = cv.resize(img, (640, 480), )
        self.img = img.copy()
        img1 = cv.GaussianBlur(self.img, (5, 5), 0)
        hsv = cv.cvtColor(img1, cv.COLOR_BGR2HSV)
        mask = cv.inRange(hsv, color_lower, color_upper)
        mask = cv.erode(mask, None, iterations=2)
        mask = cv.dilate(mask, None, iterations=2)
        mask = cv.GaussianBlur(mask, (5, 5), 0)
        cnts = cv.findContours(mask.copy(), cv.RETR_EXTERNAL,
cv.CHAIN_APPROX_SIMPLE)[-2]
```

```
        pos = None
        if len(cnts) > 0:
            cnt = max(cnts, key=cv.contourArea)
            (color_x, color_y), color_radius = cv.minEnclosingCircle(cnt)
            if color_radius > 10:
                # Mark the detected color with the prototype coil
                cv.circle(self.img, (int(color_x), int(color_y)),
int(color_radius), (255, 0, 255), 3)
                pos = (int(color_x), int(color_y))
        return self.img, pos
```

- Create widgets

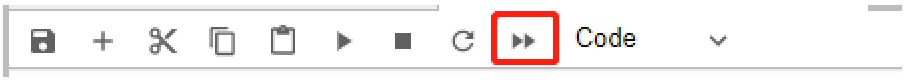
```
button_layout = widgets.Layout(width='200px', height='100px',
align_self='center')
# 输出控件 Output widget
output = widgets.Output()
# 颜色定位 Color position
color_position = widgets.Button(description='color_position',
button_style='success', layout=button_layout)
# 选择颜色 Select color
choose_color = widgets.ToggleButtons(options=['red', 'green', 'blue', 'yellow'],
button_style='success',
             tooltips=['Description of slow', 'Description of regular',
'Description of fast'])
# 取消追踪 Cancel tracking
position_cancel = widgets.Button(description='position_cancel',
button_style='danger', layout=button_layout)

# 退出 exit
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='auto'))
# 垂直布局 Vertical layout
img_box = widgets.VBox([imgbox, choose_color],
layout=widgets.Layout(align_self='auto'))
# 垂直布局 Vertical layout
Slider_box = widgets.VBox([color_position,position_cancel,exit_button],
                          layout=widgets.Layout(align_self='auto'))
# 水平布局 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']
```

Main process:

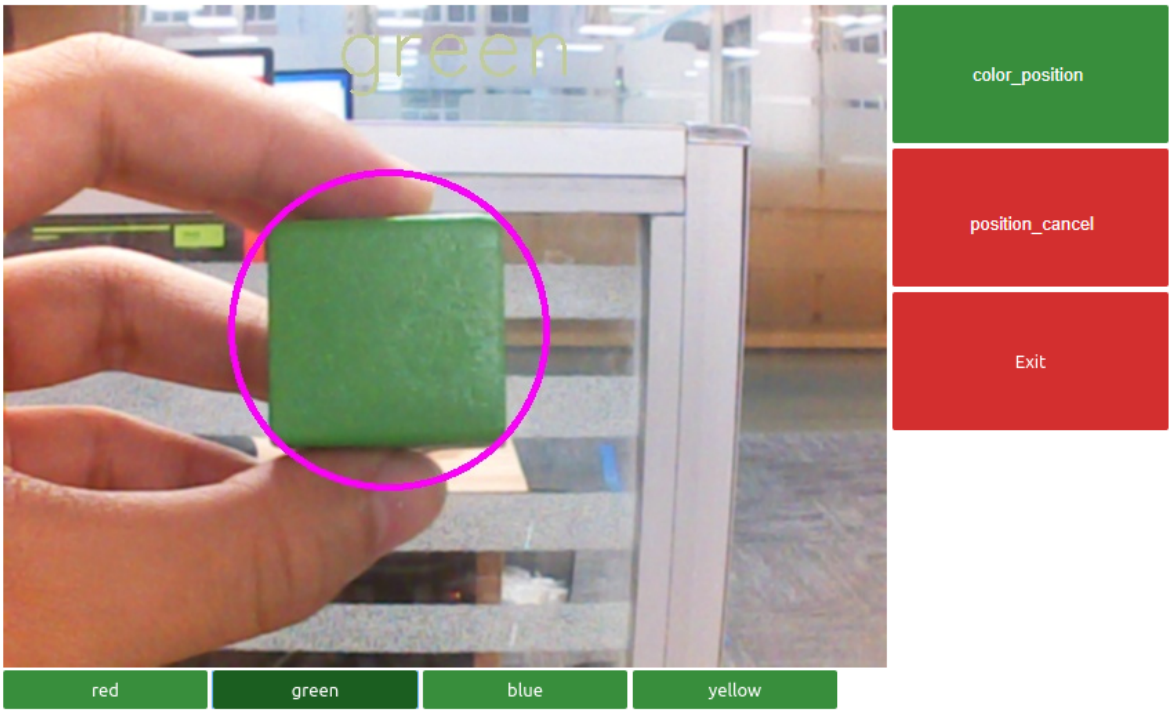
```
def camera():
    global model
    # 打开摄像头 Open camera
    capture = cv.VideoCapture(0, cv.CAP_V4L2)
    capture.set(3, 640)
    capture.set(4, 480)
    capture.set(5, 30)
    # Be executed in loop when the camera is opened normally
    while capture.isOpened():
        try:
            _, img = capture.read()
            fps.update_fps()
            if model == 'color_position':
                img, pos = position.process(img, color_hsv[choose_color.value])
                cv.putText(img, choose_color.value, (int(img.shape[0] / 2), 50),
cv.FONT_HERSHEY_SIMPLEX, 2, color[random.randint(0, 254)], 2)
                if pos is not None:
                    print("x={}, y={}".format(pos[0], pos[1]))
            if model == 'Exit':
                cv.destroyAllWindows()
                capture.release()
                break
            fps.show_fps(img)
            imgbox.value = cv.imencode('.jpg', img)[1].tobytes()
        except:capture.release()
```

2. Run the Program

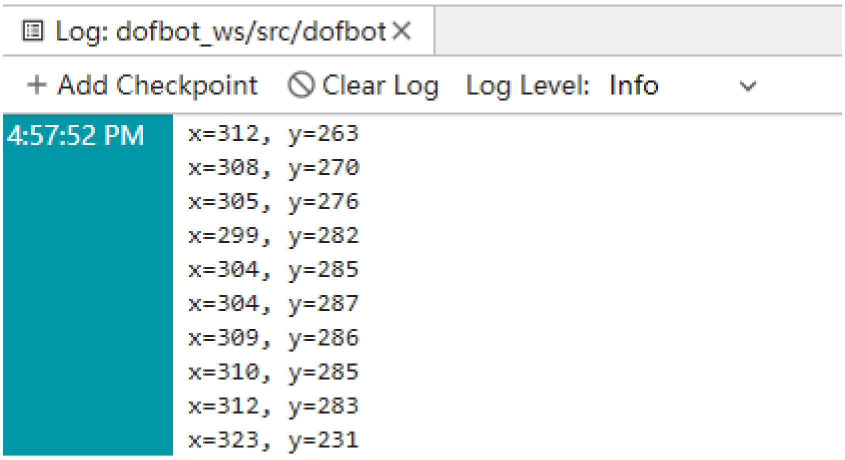
Click the run entire program button on the jupyterlab toolbar, then scroll to the bottom.



You can see the camera display. Click [color_position], select the color to position, for example, select [green] here. You can see that the green block has been framed.



Open Log information, select Log Level as Info, you can see the printed position coordinates of the color block.



```
4:57:52 PM  x=312, y=263
            x=308, y=270
            x=305, y=276
            x=299, y=282
            x=304, y=285
            x=304, y=287
            x=309, y=286
            x=310, y=285
            x=312, y=283
            x=323, y=231
```

If you need to close the function, please click the [position_cancel] button.

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

Note: If color recognition is not accurate, you can use the ~/dofbot_pro/dofbot_color_follow/scripts/HSV_calibration.ipynb file to calibrate colors.