Tag code tracking experiment

1. Functional introduction

According to the tag code positioning function, the tag code tracking function is realized in combination with the robot arm.

Code path:

```
~/dofbot_ws/src/dofbot_apriltag/scripts/Apriltag_Follow.ipynb
```

2. Code block design

- Import header file

```
import cv2 as cv
import threading
import random
from time import sleep
import ipywidgets as widgets
from IPython.display import display
from apriltag_identify import ApriltagIdentify
from apriltag_follow import Apriltag_Follow
from dofbot_utils.fps import FPS
from dofbot_utils.robot_controller import Robot_Controller
```

- Create an instance and initialize parameters

```
apriltag_Identify = ApriltagIdentify()
follow = Apriltag_Follow()
model = 'General'

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

- Creating Controls

```
button_layout      = widgets.Layout(width='320px', height='60px',
align_self='center')
output = widgets.Output()

# 开始追踪 Start tracking
start_button = widgets.Button(description='Start', button_style='success',
layout=button_layout)
# 取消追踪 Cancel tracking
cancel_button = widgets.Button(description='Cancel', button_style='danger',
layout=button_layout)
# 退出 exit
```

```
exit_button = widgets.Button(description='Exit', button_style='danger',
layout=button_layout)

box_button = widgets.VBox([start_button, cancel_button, exit_button],
layout=widgets.Layout(align_self='center'))
# 图像控件 Image widget
imgbox = widgets.Image(format='jpg', height=480, width=640)
# 垂直布局 Vertical layout
display_box = widgets.HBox([imgbox, box_button])
```

- Switching Mode

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

def start_button_Callback(value):
    global model
    model = 'Start'

def cancel_button_Callback(value):
    global model
    model = 'General'

exit_button.on_click(exit_button_Callback)
start_button.on_click(start_button_Callback)
cancel_button.on_click(cancel_button_Callback)
```

- Main Program

```
def camera():
    global model
    # 打开摄像头 Open camera
    capture = cv.VideoCapture(0)
    capture.set(3, 640)
    capture.set(4, 480)
    capture.set(5, 30)
    # 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()
            fps.update_fps()
            img, msg = apriltag_Identify.getApriltagPosition(img)
            # if len(msg):
            #     print(msg)
            if model == 'Start':
                follow.follow_function(msg)
            if model == 'Exit':
                capture.release()
                break
            fps.show_fps(img)
            imgbox.value = cv.imencode('.jpg', img)[1].tobytes()
        except Exception as e:
            print("program end")
            print(e)
```

```
            capture.release()
```

- start up

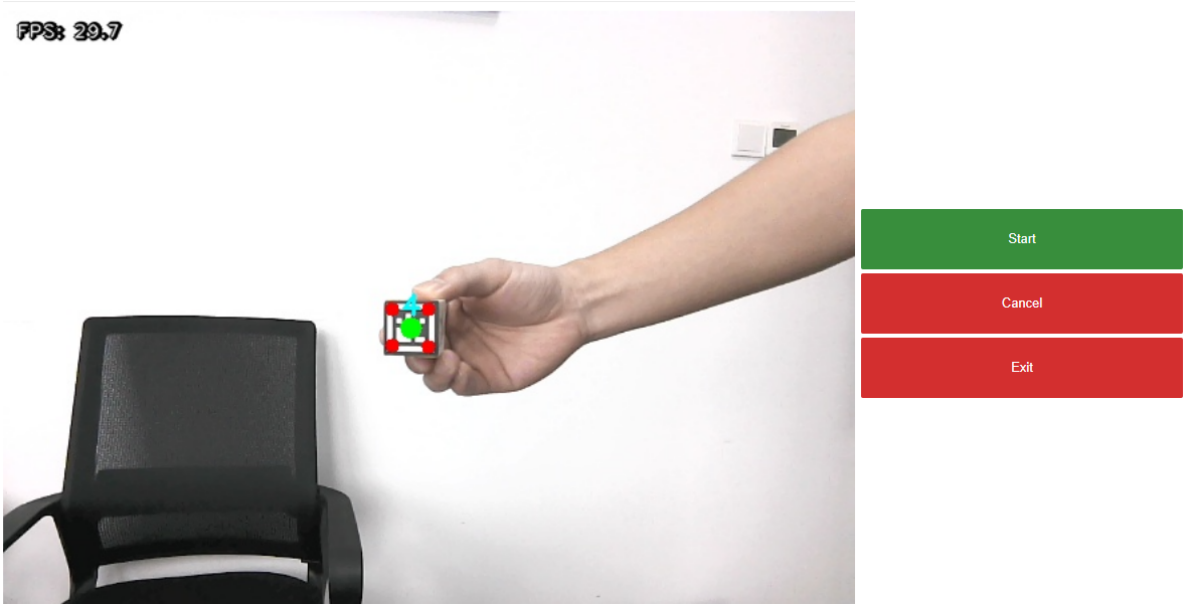
```
display(controls_box,output)
threading.Thread(target=camera, ).start()
```

3. Run the program

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

You can see the camera screen. At this time, click the [Start] button to turn on the tracking function. Put the building block of the label code into the camera screen. You can see the correct positioning of the label code and the ID number written on the label code. Move the label code and the robot will move with the label code.

Note that the speed should not be too fast when moving the label code, otherwise the robot may not be able to keep up because of moving too fast.



If you want to stop tracking, please click the [Cancel] button.

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