Robotic Arm Tracks the Palm （Orin)

Orin board users can directly open the terminal and input the tutorial commands to run directly. Jetson-Nano board users need to enter the docker container first, then input the tutorial commands in the docker to start the program.

1. Introduction

The robotic arm palm tracking function is based on robotic arm palm target positioning, adding the capability to control robotic arm movement. According to the palm's coordinate position in the camera, combined with PID algorithm to control the robotic arm's joint angles, thereby achieving the function of robotic arm tracking the palm.

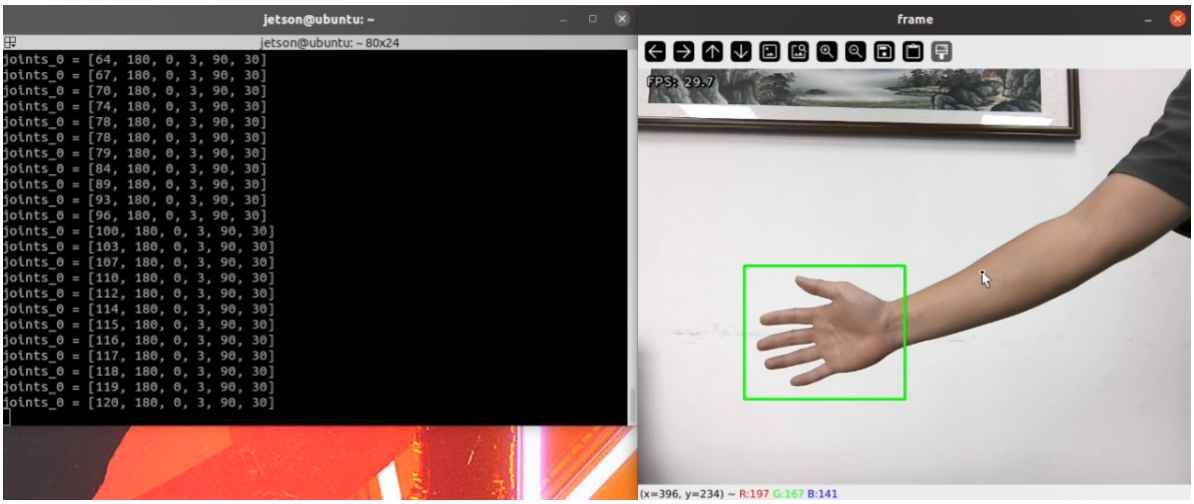
2. Launch

2.1. Preparation Before Program Launch

Note, when this program runs, the robotic arm's movement range is quite large. There should be no other objects around the robotic arm to avoid being hit by the robotic arm.

2.2. Program Description

After the program starts, when the camera captures images, the robotic arm will follow the palm's movement in the frame. Here **the palm's movement speed should not be too fast, otherwise the robotic arm cannot keep up.**



2.3. Program Launch

- Enter the following command to start the program

```
# Jetson-nano board users need to start the camera node in docker, startup
command is ros2 launch orbbec_camera dabai_dcw2.launch.py
ros2 run dofbot_pro_mediapipe 14_HandFollow
```

Press the q key in the image or press Ctrl+c in the terminal to exit the program.

3. Source Code

Code path:

```
# Jetson-Nano users need to enter the docker container to view
~/dofbot_pro_ws/src/dofbot_pro_mediapipe/dofbot_pro_mediapipe/14_HandFollow.py
```

```
#!/usr/bin/env python3
# encoding: utf-8
import threading
import numpy as np
import time
import cv2 as cv
from dofbot_utils.robot_controller import Robot_Controller
from dofbot_utils.fps import FPS
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import Image
from cv_bridge import CvBridge
import sys
import os

sys.path.append('/home/jetson/dofbot_pro_ws/src/dofbot_pro_mediapipe/dofbot_pro_
mediapipe')
from media_library import *
from simple_pid import PID

class HandCtrlArmNode(Node):
    def __init__(self):
        super().__init__('hand_ctrl_arm_node')

        self.target_servox = 0
        self.target_servoy = 0
        self.xservo_pid = PID(Kp=10, Ki=2.5, Kd=5.5, output_limits=(-90, 90))
        self.yservo_pid = PID(Kp=10, Ki=1.5, Kd=5.5, output_limits=(-90, 90))

        self.robot = Robot_Controller()
        self.robot.move_init_pose()

        self.hand_detector = HandDetector()
        self.pTime = 0
        self.event = threading.Event()
        self.event.set()
        sleep(2)

        self.bridge = CvBridge()
        self.publisher_ = self.create_publisher(Image, 'processed_image', 10)

        self.capture = cv.VideoCapture(0, cv.CAP_V4L2)
        self.capture.set(cv.CAP_PROP_FRAME_WIDTH, 640)
        self.capture.set(cv.CAP_PROP_FRAME_HEIGHT, 480)
        self.get_logger().info(f"Camera FPS:
{self.capture.get(cv.CAP_PROP_FPS)}")

        self.timer = self.create_timer(0.1, self.timer_callback)
```

```
    def process(self, frame):
        frame, lmList, bbox = self.hand_detector.findHands(frame)
        if len(lmList) != 0:
            threading.Thread(target=self.find_hand_threading, args=(lmList,
bbox)).start()
        return frame

    def find_hand_threading(self, lmList, bbox):
        hand_x = (bbox[0] + bbox[2]) / 2
        hand_y = (bbox[1] + bbox[3]) / 2
        output_x = 0
        output_y = 0
        hand_x = hand_x / 640
        if abs(hand_x - 0.5) > 0.02:
            pause_x = False
            self.xservo_pid.setpoint = 0.5
            output_x = self.xservo_pid(hand_x)
            self.target_servox = int(min(max(self.target_servox + output_x,
-90), 90))
        else:
            pause_x = True
            self.xservo_pid.reset()

        hand_y = hand_y / 480
        if abs(hand_y - 0.5) > 0.02:
            pause_y = False
            self.yservo_pid.setpoint = 0.5
            output_y = self.yservo_pid(hand_y)
            self.target_servoy = int(min(max(self.target_servoy - output_y, 0),
90))
        else:
            pause_y = True
            self.yservo_pid.reset()
        if not (pause_x and pause_y):
            joints_0 = [self.target_servox + 90, self.target_servoy + 90, 90 -
self.target_servoy, 3, 90, 30]
            self.robot.arm_move_6(joints_0, 1000)

    def timer_callback(self):
        ret, frame = self.capture.read()
        if ret:
            frame = self.process(frame)
            processed_image_msg = self.bridge.cv2_to_imgmsg(frame, "bgr8")
            self.publisher_.publish(processed_image_msg)
            cv.imshow('frame', frame)
            if cv.waitKey(1) & 0xFF == ord('q'):
                self.capture.release()
                cv.destroyAllWindows()
                rclpy.shutdown()

def main(args=None):
    rclpy.init(args=args)
    hand_ctrl_arm_node = HandCtrlArmNode()
    rclpy.spin(hand_ctrl_arm_node)
    hand_ctrl_arm_node.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
```

```
    main()
```