Robotic arm tracking palm

1. Introduction

The function of robotic arm tracking palm is to increase the function of controlling the movement of the robotic arm on the basis of the target positioning of the robotic arm palm. According to the coordinate position of the palm in the camera, the joint angle of the robotic arm is controlled in combination with the PID algorithm, so as to realize the function of the robotic arm tracking the palm.

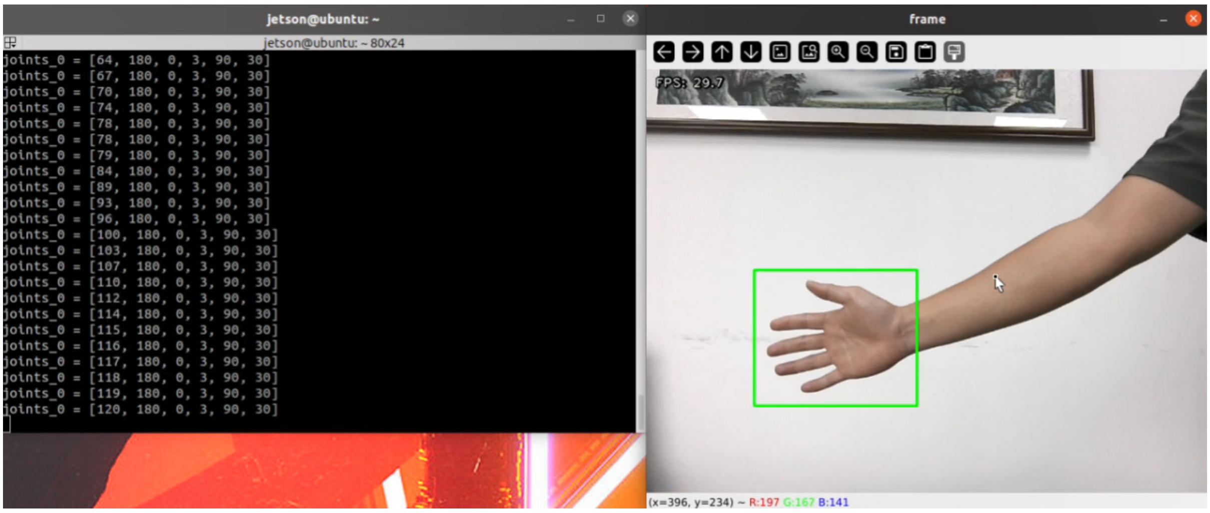
2. Startup

2.1. Preparation before starting the program

Note that when the program is running, the range of motion of the robotic arm is relatively large, and 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 is started, after the camera captures the image, the robotic arm will follow the movement of the palm in the picture. Here**the movement speed of the palm should not be too fast, otherwise the robotic arm cannot keep up.**



```
joints_0 = [64, 180, 0, 3, 90, 30]
joints_0 = [67, 180, 0, 3, 90, 30]
joints_0 = [70, 180, 0, 3, 90, 30]
joints_0 = [74, 180, 0, 3, 90, 30]
joints_0 = [78, 180, 0, 3, 90, 30]
joints_0 = [78, 180, 0, 3, 90, 30]
joints_0 = [79, 180, 0, 3, 90, 30]
joints_0 = [84, 180, 0, 3, 90, 30]
joints_0 = [89, 180, 0, 3, 90, 30]
joints_0 = [93, 180, 0, 3, 90, 30]
joints_0 = [96, 180, 0, 3, 90, 30]
joints_0 = [100, 180, 0, 3, 90, 30]
joints_0 = [103, 180, 0, 3, 90, 30]
joints_0 = [107, 180, 0, 3, 90, 30]
joints_0 = [110, 180, 0, 3, 90, 30]
joints_0 = [112, 180, 0, 3, 90, 30]
joints_0 = [114, 180, 0, 3, 90, 30]
joints_0 = [115, 180, 0, 3, 90, 30]
joints_0 = [116, 180, 0, 3, 90, 30]
joints_0 = [117, 180, 0, 3, 90, 30]
joints_0 = [118, 180, 0, 3, 90, 30]
joints_0 = [119, 180, 0, 3, 90, 30]
joints_0 = [120, 180, 0, 3, 90, 30]
```

2.3. Program startup

- Enter the following command to start the program

```
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:

```
~/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
```

```python
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()
```