Robotic Arm Tracks the Palm (Jetson-Nano)

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.

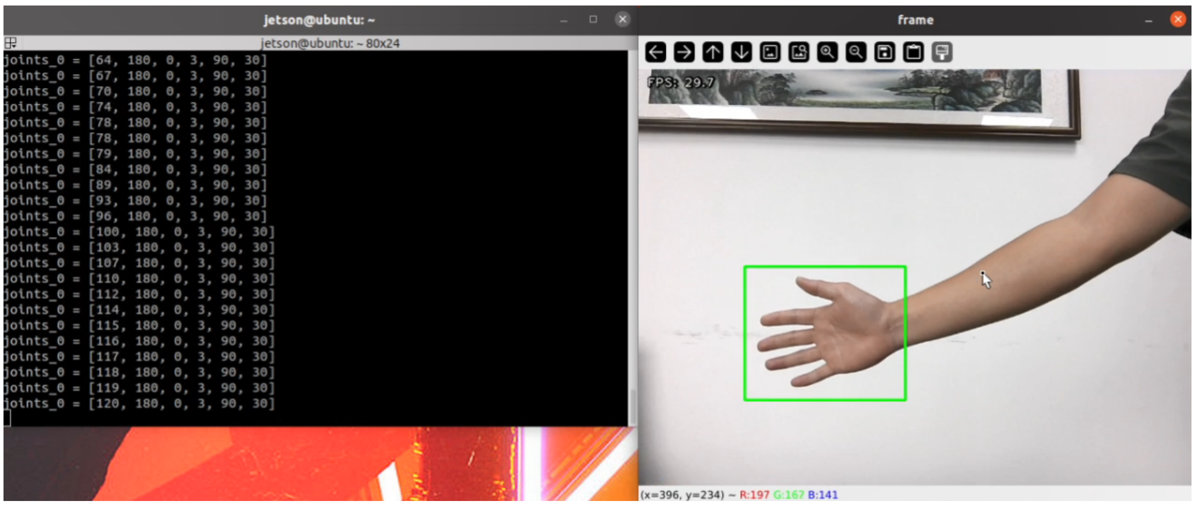
2. Launch

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.**



```
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 Launch

- **This case runs on the host machine**, enter the following command to start the program

```
roscore
rosrun dofbot_hand Hand_Follow.py
```

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

3. Source Code

Code path:

```
~/dofbot_ws/src/dofbot_hand/scripts/Hand_Follow.py
```

```
#!/usr/bin/env python3
# encoding: utf-8
import os
import threading
import numpy as np
from time import sleep, time
from media_library import *
from simple_pid import PID
from dofbot_utils.robot_controller import Robot_Controller
from dofbot_utils.fps import FPS


class HandCtrlArm:
    def __init__(self):
        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)


    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()
        # self.cTime = time()
        # fps = 1 / (self.cTime - self.pTime)
        # self.pTime = self.cTime
        # text = "FPS : " + str(int(fps))
        # cv.putText(frame, text, (20, 30), cv.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0,
255), 1)
        return frame


    def find_hand_threading(self, lmList, bbox):
        hand_x = (bbox[0] + bbox[2]) / 2
        hand_y = (bbox[1] + bbox[3]) / 2
        # print("hand_x: {}, hand_y:{}".format(hand_x, hand_y))
        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]
            # print("output:", output_x, output_y)
            print("joints_0 = {}".format(joints_0))
            self.robot.arm_move_6(joints_0, 1000)


if __name__ == '__main__':
    capture = cv.VideoCapture(0)
    # capture.set(6, cv.VideoWriter.fourcc('M', 'J', 'P', 'G'))
    capture.set(cv.CAP_PROP_FRAME_WIDTH, 640)
    capture.set(cv.CAP_PROP_FRAME_HEIGHT, 480)
    print("capture get FPS : ", capture.get(cv.CAP_PROP_FPS))
    ctrl_arm = HandCtrlArm()
    fps = FPS()
    while capture.isOpened():
        ret, frame = capture.read()
        fps.update_fps()
        action = cv.waitKey(1) & 0xFF
        frame = ctrl_arm.process(frame)
        if action == ord('q'):
            break
        fps.show_fps(frame)
        cv.imshow('frame', frame)
    capture.release()
    cv.destroyAllWindows()
```