10. MediaPipe identifies and tracks the palm to control the robotic arm

10.1. Introduction

MediaPipe is an open-source data stream processing machine learning application development framework developed by Google. It is a graph-based data processing pipeline used to build data sources in various forms, such as video, audio, sensor data, and any time series data.

MediaPipe is cross-platform and can run on embedded platforms (Raspberry Pi, etc.), mobile devices (iOS and Android), workstations and servers, and supports mobile GPU acceleration. MediaPipe provides cross-platform, customizable ML solutions for real-time and streaming media.

The core framework of MediaPipe is implemented in C++ and provides support for languages such as Java and Objective C. The main concepts of MediaPipe include packets, streams, calculators, graphs, and subgraphs.

Features of MediaPipe:

- End-to-end acceleration: built-in fast ML inference and processing can be accelerated even on ordinary hardware.
- Build once, deploy anywhere: unified solution for Android, iOS, desktop/cloud, web and IoT.
- Ready-to-use solution: cutting-edge ML solution that demonstrates the full functionality of the framework.
- Free and open source: framework and solution under Apache2.0, fully extensible and customizable.

10.2, Startup

10.2.1, Preparation before starting the program

Note that when the program is running, the range of motion of the robot arm is relatively large, and there should be no other objects around the robot arm to avoid being hit by the robot arm.

10.2.2, Program Description

After the program is started, after the camera captures the image, the robot 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 robot arm cannot keep up. **

10.3.2, Program Startup

- Enter the following command to start the program

```
ros2 run jetcobot_mediapipe Hand_Ctrl
```

10.3.3, Source Code

Code path: ~/jetcobot_ws/src/jetcobot_mediapipe/jetcobot_mediapipe/Hand_Ctrl.py

```
#!/usr/bin/env python3
# encoding: utf-8
import os
import threading
import numpy as np
from time import sleep, time
from simple_pid import PID
from pymycobot.mycobot import MyCobot
import rclpy
from rclpy.node import Node
from jetcobot_mediapipe.media_library import *

class HandCtrlArm:
    def __init__(self):
        self.target_servox=0
        self.target_servoy=-90
        self.xservo_pid = PID(3.5, 0.1, 0.05)
        self.yservo_pid = PID(2, 0.05, 0.05)

        self.mc = MyCobot('/dev/ttyUSB0', 1000000)
        self.mc.send_angles([0, 0, -90, 90, 0, -45], 50)

        self.hand_detector = HandDetector()
        self.arm_status = True
        self.locking = True
        self.init = True
        self.pTime = 0
        self.add_lock = self.remove_lock = 0
        self.Joy_active = True
        self.event = threading.Event()
        self.event.set()
        self.Joy_active = True


    def process(self, frame):
        frame = cv.flip(frame, 1)
        if self.Joy_active:
            frame, lmList, bbox = self.hand_detector.findHands(frame)
            if len(lmList) != 0 and self.Joy_active:
                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_x)
        hand_x = hand_x / 640
        print("hand_x: ",hand_x)
        if abs(hand_x - 0.5) > 0.02:
            self.xservo_pid.setpoint = 0.5
            output = self.xservo_pid(hand_x, dt=0.1)
            self.target_servox = min(max(self.target_servox - output, -160), 160)
        else:
            self.xservo_pid.reset()

        hand_y = hand_y / 480
        if abs(hand_y - 0.5) > 0.02:
            self.yservo_pid.setpoint = 0.5
            output = self.yservo_pid(hand_y, dt=0.1)
            self.target_servoy = min(max(self.target_servoy + output, -140), 0)
        else:
            self.yservo_pid.reset()

        joints_0 = [self.target_servox, 0, self.target_servoy, -
self.target_servoy, 0, -45]
        print("joints_0 = {}".format(joints_0))
        self.mc.send_angles(joints_0, 50)



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()
    while capture.isOpened():
        ret, frame = capture.read()
        action = cv.waitKey(1) & 0xFF
        frame = ctrl_arm.process(frame)
        if action == ord('q'):
            #ctrl_arm.media_ros.cancel()
            break
        cv.imshow('frame', frame)
    capture.release()
    cv.destroyAllWindows()
```