Mediapipe arm posture control robotic arm

1.Introduction

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

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

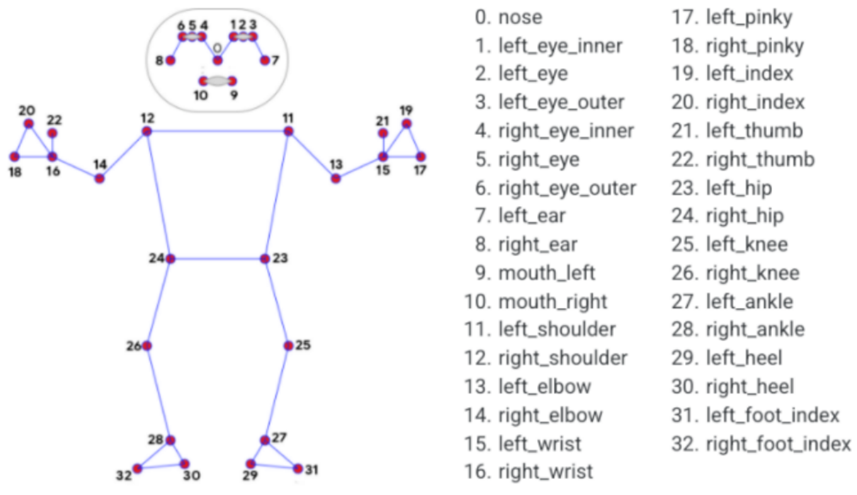
The core framework of MediaPipe is implemented in C++and provides support for languages such as Java and Objective C. The main concepts of MediaPipe include packets, streams, calculators, graphs, and subgraphs.

Features of MediaPipe:

- End to end acceleration: Built in fast ML inference and processing can accelerate even on regular hardware.
- Build once, deploy anytime, anywhere: A unified solution suitable for Android, iOS, desktop/cloud, web, and IoT.
- Ready to use solution: a cutting-edge ML solution that showcases all the functionalities of the framework.
- Free and open-source: frameworks and solutions under Apache 2.0, fully extensible and customizable.

2. Working principle

The landmark model in MediaPipe Pose predicted the positions of 33 pose coordinates (as shown in the figure below).



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

3. Start

3.1 Preparation

Note: when the program is running, the robotic arm has a large range of movement. Do not place other objects around the robotic arm to avoid being hit by the robotic arm.

3.2 Code analysis

After the program is started, stand in front of the camera so that the upper part of the body appears in the picture. There are a total of five body movements that can control the movement of the robotic arm.

The five movements are [lower left hand, raise right hand] [lower right hand, raise left hand] [both hands up] [hands on waist] [hands forming a triangle].

3.3 Start program

- If you are using Jetson Orin NX/Jetson Orin Nano board. You need to enter the Docker environment using the following command.

```
sh ~/start_docker.sh
```

- Input following command to start the program

```
roscore
rosrun jetcobot_mediapipe PoseArm.py
```

3.4 About code

Code path： ~/jetcobot_ws/src/jetcobot_mediapipe/scripts/PoseArm.py

```
#!/usr/bin/env python3
# encoding: utf-8
import os
import threading
import cv2 as cv
import numpy as np
from media_library import *
from time import sleep, time
from pymycobot.mycobot import MyCobot
from pymycobot.genre import Angle

class PoseCtrlArm:
    def __init__(self):
        self.mc = MyCobot(str(os.getenv('MY_SERIAL')), 1000000)
        self.car_status = True
        self.stop_status = 0
        self.locking = False
        self.pose_detector = Holistic()
        self.hand_detector = HandDetector()
        self.pTime = self.index = 0
        self.media_ros = Media_ROS()
        self.reset_pose()
        self.event = threading.Event()
        self.event.set()

    def reset_pose(self):
        self.mc.send_angles([0, 0, 0, 0, 0, -45], 50)
        sleep(1.5)

    def process(self, frame):
        frame = cv.flip(frame, 1)
        frame, pointArray, lhandptArray, rhandptArray =
self.pose_detector.findHolistic(frame)
        threading.Thread(target=self.arm_ctrl_threading, args=(pointArray,
lhandptArray, rhandptArray)).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)
        self.media_ros.pub_imgMsg(frame)
        return frame

    def get_angle(self, v1, v2):
        angle = np.dot(v1, v2) / (np.sqrt(np.sum(v1 * v1)) * np.sqrt(np.sum(v2 *
v2)))
        angle = np.arccos(angle) / 3.14 * 180
        cross = v2[0] * v1[1] - v2[1] * v1[0]
        if cross < 0:
            angle = - angle
        return angle


    def get_pos(self, keypoints):
        str_pose = ""
        # 计算左臂与水平方向的夹角
        keypoints = np.array(keypoints)
        v1 = keypoints[12] - keypoints[11]
        v2 = keypoints[13] - keypoints[11]
        angle_left_arm = self.get_angle(v1, v2)
        #计算右臂与水平方向的夹角
        v1 = keypoints[11] - keypoints[12]
        v2 = keypoints[14] - keypoints[12]
        angle_right_arm = self.get_angle(v1, v2)
        #计算左肘的夹角
        v1 = keypoints[11] - keypoints[13]
        v2 = keypoints[15] - keypoints[13]
        angle_left_elow = self.get_angle(v1, v2)
        # 计算右肘的夹角
        v1 = keypoints[12] - keypoints[14]
        v2 = keypoints[16] - keypoints[14]
        angle_right_elow = self.get_angle(v1, v2)

        if 90<angle_left_arm<120 and -120<angle_right_arm<-90:
            str_pose = "NORMAL"
        elif 90<angle_left_arm<120 and 90<angle_right_arm<120:
            # 左手放下，举起右手
            str_pose = "RIGHT_UP"
        elif -120<angle_left_arm<-90 and -120<angle_right_arm<-90:
            # 右手放下，举起左手
            str_pose = "LEFT_UP"
        elif -120<angle_left_arm<-90 and 90<angle_right_arm<120:
            # 手上向上
            str_pose = "ALL_HANDS_UP"
        elif 130<angle_left_arm<150 and -150<angle_right_arm<-130 and 
90<angle_left_elow<120 and -120<angle_right_elow<90:
            # 双手叉腰
            str_pose = "AKIMBO"
        elif -150<angle_left_arm<-120 and 120<angle_right_arm<150 and 
-85<angle_left_elow<-55 and 55<angle_right_elow<85:
            # 双手合成三角形
            str_pose = "TRIANGLE"
        # print("str_pose = ",str_pose)
        # print("angle_left_arm = ",angle_left_arm,"\tangle_right_arm = 
",angle_right_arm)
        # print("angle_left_elow = ",angle_left_elow,"\tangle_right_elow = 
",angle_right_elow)
        return str_pose

    def arm_ctrl_threading(self, pointArray, lhandptArray, rhandptArray):
        keypoints = ['' for i in range(33)]
        if self.event.is_set():
            self.event.clear()
            if len(pointArray) != 0:
                for i in range(len(pointArray)):
                    keypoints[i] = (pointArray[i][1],pointArray[i][2])

                str_pose = self.get_pos(keypoints)
                if str_pose:
                    print("str_pose = ",str_pose)
                if str_pose=="RIGHT_UP":
                    self.RIGHT_UP()
                elif str_pose=="LEFT_UP":
                    self.LEFT_UP()
                elif str_pose=="ALL_HANDS_UP":
                    self.ALL_HANDS_UP()
                elif str_pose=="TRIANGLE":
                    self.TRIANGLE()
                elif str_pose=="AKIMBO":
                    self.AKIMBO()
                self.event.set()
            else:
                self.event.set()

    def RIGHT_UP(self):
        self.mc.send_angles([90, 80, 0, -90, -90, -45], 50)
        sleep(3)
        self.reset_pose()

    def LEFT_UP(self):
        self.mc.send_angles([-90, 80, 0, -90, 90, -45], 50)
        sleep(3)
        self.reset_pose()

    def ALL_HANDS_UP(self):
        self.mc.send_angles([0, 0, 0, 80, 0, -45], 50)
        sleep(3)
        self.reset_pose()

    def TRIANGLE(self):
        self.mc.send_angles([0, 90, -120, 50, 0, -45], 50)
        sleep(3)
        self.reset_pose()

    def AKIMBO(self):
        self.mc.send_angles([0, 90, -120, -20, 0, -45], 50)
        sleep(3)
        self.reset_pose()

if __name__ == '__main__':
    rospy.init_node('PoseCtrlArm_node', anonymous=True)
    pose_ctrl_arm = PoseCtrlArm()
    capture = cv.VideoCapture("/dev/video0")
    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))
    while capture.isOpened():
        ret, frame = capture.read()
        frame = pose_ctrl_arm.process(frame)
        if cv.waitKey(1) & 0xFF == ord('q'): break
        cv.imshow('frame', frame)
    capture.release()
    cv.destroyAllWindows()
```