8. MediaPipe arm gesture control robot

8.1. 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.

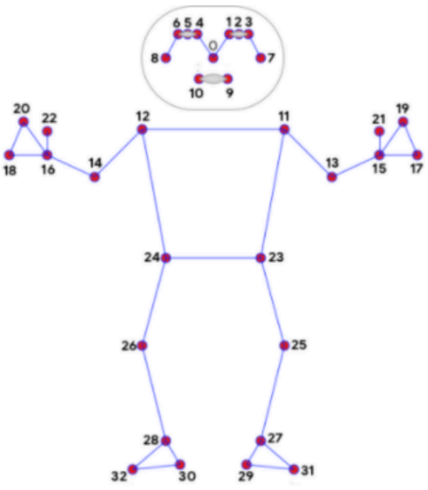
The core framework of MediaPipe is implemented in C++ and provides support for languages such as Java and Objective C. The main concepts of MediaPipe include packets, streams, calculators, graphs, and subgraphs.

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.

8.2, Implementation principle

The landmark model in MediaPipe Pose predicts the position of 33 pose coordinates (see 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

In this program, we need the coordinates of the body and arm parts, and control the movement of the robotic arm by calculating the angle formed by these coordinates.

8.3, Startup

8.3.1, Preparation before program startup

Note that the robot arm has a large range of motion when the program is running. Do not place other objects around the robot arm to avoid being hit by the robot arm.

8.3.2, Program description

After the program is started, stand in front of the camera so that the entire upper part of the body appears in the picture. There are a total of five body movements that can control the movement of the robot arm. The five movements are [lower left hand, raise right hand] [lower right hand, raise left hand] [both hands up] [hands on waist] [hands form a triangle].

8.3.2, Program startup

- Enter the following command to start the program

```
ros2 run jetcobot_mediapipe PoseArm
```

8.3.3, Source code

Code path: ~/jetcobot_ws/src/jetcobot_mediapipe/jetcobot_mediapipe/PoseArm.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 PoseCtrlArm:
    def __init__(self):
        self.mc = MyCobot('/dev/ttyUSB0', 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()

def main(args=None):
    rclpy.init(args=args)
    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))
    try:
        while capture.isOpened() and rclpy.ok():
            ret, frame = capture.read()
            if ret:
                frame = pose_ctrl_arm.process(frame)
                cv.imshow('frame', frame)
            if cv.waitKey(1) & 0xFF == ord('q'):
                break
    finally:
        capture.release()
        cv.destroyAllWindows()
        pose_ctrl_arm.destroy_node()
        rclpy.shutdown()

if __name__ == '__main__':
    main()
```