

## 7. MediaPipe gesture control robot arm action group

### 7.1. 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 commodity 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 capabilities of the framework.
- Free and open source: framework and solution under Apache2.0, fully extensible and customizable.

### 7.2, source code

Code path: ~/jetcobot\_ws/src/jetcobot\_mediapipe/jetcobot\_mediapipe/scripts/FingerCtrl.py

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

class HandCtrlArm:
    def __init__(self):
        self.mc = MyCobot('/dev/ttyUSB0', 1000000)
        self.media_ros = Media_ROS()
        self.hand_detector = handDetector(detectorCon=0.75)
        self.arm_status = True
        self.locking = True
        self.init = True
        self.pTime = 0
        self.add_lock = self.remove_lock = 0
        self.init_pose()
        self.event = threading.Event()
```

```

self.event.set()

def init_pose(self):
    # 机械臂初始位置
    self.mc.send_angles([0, 0, 0, 0, 0, -45], 50)
    sleep(0.5)
    # 闭合夹爪（第一个参数取值0-100，100是张开夹爪）
    self.mc.set_gripper_value(0, 50)
    sleep(3)

# 调酒
def blending(self):
    self.mc.send_radians([0.002131119865439069, -1.0102805575559934,
2.3596385151646215,
-1.349273883919728, -0.0022642357843590923,
-0.7853981633974483], 50)
    sleep(1.5)
    self.mc.send_radians([1.2282398532608345, 1.4220789519337802,
-0.573522918085324,
-0.8492764075464502, -1.2280747135108614,
-0.7853981633974483], 50)
    sleep(1.5)
    self.mc.send_radians([1.8983076765171147, -1.1608903330406473,
-0.02815422147205849,
1.1894923277112577, -1.8982241236262254,
-0.7853981633974483], 50)
    sleep(1.5)
    self.mc.send_radians([0.0018565210807136886, 1.0498746153676226,
-2.5332261556430495,
1.4832699746780278, -0.0018452938303413079,
-0.7853981633974483], 50)
    sleep(1.5)
    self.mc.send_angles([0, 0, 0, 0, 0, -45], 50)
    sleep(1.5)
    self.mc.send_radians([0.0009039212435389698, 0.8169237917361073,
-1.8101769275985953,
0.9931414011909611, -0.0010044489822263628,
-0.7853981633974483], 50)
    sleep(1.5)
    for i in range(2):
        self.mc.send_radians([0.000499411329404627, 0.40249854856005807,
-1.1807172919935842,
1.6691055316386183, -4.044151356057693e-05,
-0.7853981633974483], 50)
        sleep(1)

        self.mc.send_radians([0.0005371606450911302, -0.2684392226957434,
1.073647324612679,
-0.6767929115155326, -0.000118386158952169,
-0.7853981633974483], 50)
        sleep(1.5)
        self.mc.send_radians([0.0004003356301867094, -1.130551290420075,
-0.5353317406930522,
1.7938983073605428, 0.00011283072801522158,
-0.7853981633974483], 50)
        sleep(3)

```

```

self.mc.set_gripper_value(100, 50)
sleep(3)
self.init_pose()

# 炒菜颠锅
def stir_fry(self):
    self.mc.send_radians([0.0014636672906520056, 1.0537558918638636,
-2.572182834305464,
                                1.5183325476997855, -0.0015675503153531013,
-0.7853981633974483], 50)
    sleep(1.5)
    for i in range(5):
        if i == 0:
            self.mc.send_radians([0.6293343429206779, -1.8832064966666389,
1.6517030826936279,
                                0.23107245540775312, -0.6292105600283662,
-0.7853981633974483], 50)
        else:
            self.mc.send_radians([0.6293343429206779, -1.8832064966666389,
1.6517030826936279,
                                0.23107245540775312, -0.6292105600283662,
-0.7853981633974483], 100)
            sleep(0.5)
            self.mc.send_radians([0.6292897676468308, -1.2906457048295161,
1.0699517665022527,
                                0.22024767705303563, -0.6291465563096521,
-0.7853981633974483], 100)
            sleep(0.5)
            self.mc.send_radians([0.6294020002224259, -2.0738592273794385,
1.752136963521392,
                                0.321093603910445, -0.6292819663588981,
-0.7853981633974483], 100)
            sleep(0.5)
    sleep(1)
    self.init_pose()

# 倒茶
def pour_tea(self):
    self.mc.send_radians([0.00013987280202047583, -2.266577029102527,
1.2770636656552903,
                                0.9893391358006595, -0.00020463473184286916,
-0.7853981633974483], 50)
    sleep(1.5)
    self.mc.send_angle(Angle.J6.value, -120, 50)
    sleep(1.5)
    self.mc.send_angle(Angle.J6.value, -45, 50)
    sleep(1.5)
    self.mc.set_gripper_value(100, 50)
    sleep(2)
    self.mc.send_radians([0.0009339507816224792, -2.1308889421103827,
2.527321381024112,
                                -0.39624178824381895, -0.001131203673988256,
-0.7853981633974483], 50)
    sleep(1.5)
    self.init_pose()

```

```
# 招财猫
def caicai_cat(self):
    num = 3
    while num > 0:
        self.mc.send_angle(Angle.J3.value, -90, 50)
        sleep(1.5)
        self.mc.send_angle(Angle.J3.value, 0, 50)
        sleep(1.5)
        num -= 1
    self.mc.set_gripper_value(100, 50)
    sleep(2)
    self.init_pose()

# 点头
def nod(self):
    num = 3
    while num > 0:
        self.mc.send_angle(Angle.J4.value, 50, 50)
        sleep(1.5)
        self.mc.send_angle(Angle.J4.value, -20, 50)
        sleep(1.5)
        num -= 1
    self.init_pose()

# 摇头
def shake(self):
    for i in range(3):
        self.mc.send_angle(Angle.J5.value, 50, 100)
        sleep(1)
        self.mc.send_angle(Angle.J5.value, -50, 100)
        sleep(1)
    self.init_pose()

# 向前扑倒
def pounce_down(self):
    self.mc.send_angle(Angle.J2.value, -90, 50)
    sleep(2)
    self.init_pose()

# 机械臂做出马姿态等待主人上车
def horse_pose(self):
    self.mc.send_angles([0, -90, 0, 90, 0, -45], 50)
    sleep(2)
    self.init_pose()

# 主人骑上马儿开始奔跑
def horse_run(self):
    self.mc.send_angles([0, -90, 0, 90, 0, -45], 50)
    sleep(1)
    for i in range(5):
        self.mc.send_angle(Angle.J2.value, -95, 100)
        sleep(0.5)
        self.mc.send_angle(Angle.J2.value, -85, 100)
        sleep(0.5)
    sleep(2)
    self.init_pose()
```

```

# 机械臂俯身观察前方敌情
def bowdown_observe(self):
    self.mc.send_angles([0, -90, 0, 90, 0, -45], 50)
    sleep(1)
    for i in range(3):
        self.mc.send_angle(Angle.J1.value, -30, 20)
        sleep(2)
        self.mc.send_angle(Angle.J1.value, 30, 20)
        sleep(2)
    sleep(2)
    self.init_pose()

def process(self, frame):
    frame = cv.flip(frame, 1)
    frame, lmList = self.hand_detector.findHands(frame, draw=False)
    if len(lmList) != 0:
        threading.Thread(target=self.arm_ctrl_threading).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 arm_ctrl_threading(self):
    if self.event.is_set():
        self.event.clear()
        gesture = self.hand_detector.get_gesture()
        print("gesture = ", gesture)
        if gesture == "Zero":
            self.arm_status = False
            sleep(1.0)
            self.blending()
            self.arm_status = True

        elif gesture == "One":
            self.arm_status = False
            sleep(1.0)
            self.stir_fry()
            self.arm_status = True

        elif gesture == "Two":
            self.arm_status = False
            sleep(1.0)
            self.shake()
            self.arm_status = True

        elif gesture == "Three":
            self.arm_status = False
            sleep(1.0)
            self.nod()
            self.arm_status = True

```

```

        elif gesture == "Four":
            self.arm_status = False
            sleep(1.0)
            self.caicai_cat()
            self.arm_status = True

        elif gesture == "Five":
            self.arm_status = False
            sleep(1.0)
            self.pour_tea()
            self.arm_status = True

        elif gesture == "Six":
            self.arm_status = False
            sleep(1.0)
            self.pounce_down()
            self.arm_status = True

        elif gesture == "Seven":
            self.arm_status = False
            sleep(1.0)
            self.horse_pose()
            self.arm_status = True

        elif gesture == "Eight":
            self.arm_status = False
            sleep(1.0)
            self.horse_run()
            self.arm_status = True

        elif gesture == "OK":
            self.arm_status = False
            sleep(1.0)
            self.bowdown_observe()
            self.arm_status = True

    self.event.set()

def main(args=None):
    rclpy.init(args=args)
    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))

    ctrl_arm = FingerCtrlArm()

    try:
        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)
    rclpy.spin_once(ctrl_arm, timeout_sec=0.001)
finally:
    capture.release()
    cv.destroyAllWindows()
    ctrl_arm.destroy_node()
    rclpy.shutdown()

if __name__ == '__main__':
    main()

```

## 7.3, Start

- Enter the following command to start the program

```
ros2 run jetcobot_mediapipe FingerCtrl
```

After the program runs, the camera will capture images. There are 9 gestures, as follows,

- Gesture numbers 1~8 and gesture OK: correspond to 9 action groups.

Here, when each gesture is completed, it will return to the initialization position and wait for the next gesture recognition.

