Mediapipe Gesture Control Robotic Arm Action Group

Note: There are related running codes on Raspberry Pi and Jetson nano, but due to the difference in motherboard performance, they may not run so smoothly. The supporting virtual machine also has the operating environment and programs installed. If the experience on the motherboard is not good, you can Remove the camera and insert it into the virtual machine. Connect the camera device to the virtual machine to run the image processing program on the virtual machine and run the driver on the motherboard. The premise is that distributed communication between the motherboard and the virtual machine needs to be set up. Refer to the previous tutorial to set it up.

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 for building and using multiple 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 (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 Packet, Stream, Calculator, Graph and Subgraph.

Features of MediaPipe:

- End-to-end acceleration: Built-in fast ML inference and processing accelerates even on commodity hardware.
- Build once, deploy anywhere: Unified solution for Android, iOS, desktop/cloud, web and IoT.
- Ready-to-use solutions: cutting-edge ML solutions that showcase the full capabilities of the framework.
- Free and open source: frameworks and solutions under Apache2.0, fully extensible and customizable.

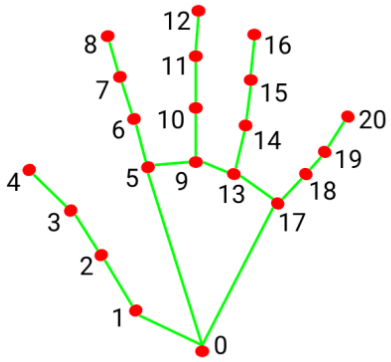
2. Start

After the program is run, the camera will capture images and there are six gestures, as follows,

- Gesture Yes: the robotic arm dances
- Gesture number 1: Robotic arm nods
- Gesture rock (the index finger and little finger are straightened, and the other fingers are bent): the robotic arm is straightened and turned left and right
- Gesture number 5: robotic arm applauds
- Gesture contempt (clench fist, extend thumb, thumb down): The robotic arm kneels down and the clamping claws are tightened
- Gesture OK: The robotic arm shakes its head and the clamping claws tighten

Here, after each gesture is completed, it will return to the initial position and beep, waiting for the next gesture recognition.

MediaPipe Hands infers the 3D coordinates of 21 hand-valued joints from a frame,



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Motherboard runs:

```
roscore
rosrun arm_mediapipe arm_driver.py
rosrun arm_mediapipe FingerCtrl.py
```

3. Source code

3.1, arm_driver.py

Raspberry Pi source code location:/root/dofbot_ws/src/arm_mediapipe/scripts/arm_driver.py

```
#!/usr/bin/env python3
# encoding: utf-8
import rospy
from Arm_Lib import Arm_Device
from yahboomcar_msgs.msg import *
from std_msgs.msg import Bool
import time
Arm = Arm_Device()

joints = [90, 2.0, 60.0, 40.0, 90]
Arm.Arm_serial_servo_write6(90, 90, 90, 90, 90, 90, 1000)
time.sleep(1)
print("init done")
def Armcallback(msg):
if not isinstance(msg, ArmJoint):
print("----------")
return
arm_joint = ArmJoint()
print("msg.joints: ",msg.joints)
print("msg.joints: ",msg.run_time)
if len(msg.joints) != 0:
arm_joint.joints = joints
for i in range(2):
Arm.Arm_serial_servo_write6(msg.joints[0],
msg.joints[1],msg.joints[2],msg.joints[3],msg.joints[4],msg.joints[5],time=msg.
run_time)
```

```
time.sleep(0.01)
else:
arm_joint.id = msg.id
arm_joint.angle = msg.angle
for i in range(2):
Arm.Arm_serial_servo_write(msg.id, msg.angle, msg.run_time)
time.sleep(0.01)

def Buzzercallback(msg):
if not isinstance(msg, Bool):
print("----------")
return
if msg.data==True:
print("beep on")
Arm.Arm_Buzzer_On()
else:
Arm.Arm_Buzzer_Off()
print("beep off")

sub_Arm = rospy.Subscriber("TargetAngle", ArmJoint, Armcallback, queue_size=1000)
sub_Buzzer = rospy.Subscriber("Buzzer", Bool, Buzzercallback,queue_size=1000)

if __name__ == '__main__':
Arm.Arm_serial_servo_write6(90, 90, 90, 90, 90, 90, 1000) #Modify this place to
adjust the initial position of the robot arm
rospy.init_node("arm_driver_node",anonymous=True)
rospy.spin()
```

3.2, FingerCtrl.py

Raspberry Pi source code location:/root/dofbot_ws/src/arm_mediapipe/scripts/FingerCtrl.py

```
#!/usr/bin/env python3
# encoding: utf-8
import threading
import numpy as np
from media_library import *
from time import sleep, time

class HandCtrlArm:
    def __init__(self):
        self.media_ros = Media_ROS()
        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.media_ros.pub_arm([90, 135, 0, 45, 90, 90])
        print("111")
        self.event = threading.Event()
        self.event.set()

    def dance(self):
        print("dance")
```

```
        time_sleep = 0.5
        self.media_ros.pub_arm([90, 90, 90, 90, 90, 90])
        sleep(time_sleep)
        self.media_ros.pub_arm([90, 60, 120, 60, 90, 90])
        sleep(time_sleep)
        self.media_ros.pub_arm([90, 45, 135, 45, 90, 90])
        sleep(time_sleep)
        self.media_ros.pub_arm([90, 60, 120, 60, 90, 90])
        sleep(time_sleep)
        self.media_ros.pub_arm([90, 90, 90, 90, 90, 90])
        sleep(time_sleep)
        self.media_ros.pub_arm([90, 100, 80, 80, 90, 90])
        sleep(time_sleep)
        self.media_ros.pub_arm([90, 120, 60, 60, 90, 90])
        sleep(time_sleep)
        self.media_ros.pub_arm([90, 135, 45, 45, 90, 90])
        sleep(time_sleep)
        self.media_ros.pub_arm([90, 90, 90, 90, 90, 90])
        sleep(time_sleep)
        self.media_ros.pub_arm([90, 90, 90, 20, 90, 150])
        sleep(time_sleep)
        self.media_ros.pub_arm([90, 90, 90, 90, 90, 90])
        sleep(time_sleep)
        self.media_ros.pub_arm([90, 90, 90, 20, 90, 150])
        sleep(time_sleep)
        self.media_ros.pub_arm([0, 90, 90, 90, 0, 90])
        sleep(time_sleep)
        self.media_ros.pub_arm([0, 90, 180, 0, 0, 90])
        sleep(time_sleep)
        self.media_ros.pub_arm([], 6, 180)
        sleep(time_sleep)
        self.media_ros.pub_arm([], 6, 30)
        sleep(time_sleep)
        self.media_ros.pub_arm([90, 90, 90, 90, 90, 90])
        sleep(time_sleep)
        self.media_ros.pub_arm([90, 135, 0, 45, 90, 90])
        sleep(time_sleep)

    def init_pose(self):
        self.media_ros.pub_arm([90.0, 145.0, 0.0, 0.0, 90.0, 31.0])
        sleep(0.5)
        self.media_ros.RobotBuzzer()


    def arm_applaud(self):
        print("applaud")
        self.media_ros.pub_arm([90.0, 145.0, 0.0, 71.0, 90.0, 0.0])
        sleep(0.5)
        self.media_ros.pub_arm([], 6, 30)
        sleep(0.5)
        self.media_ros.pub_arm([], 6, 180)
        sleep(0.5)
        self.media_ros.pub_arm([], 6, 30)
        sleep(0.5)
        self.media_ros.pub_arm([], 6, 180)
        sleep(0.5)
```

```
        #self.media_ros.pub_arm([90.0, 145.0, 0.0, 0.0, 90.0, 31.0])

    def shake(self):
        print("applaud")
        for i in range(3):
            self.media_ros.pub_arm([138.0, 94.0, 92.0, 88.0, 92.0, 172.0])
            sleep(0.5)
            self.media_ros.pub_arm([48.0, 94.0, 92.0, 87.0, 92.0, 172.0])
            sleep(0.5)
        self.media_ros.pub_arm([90.0, 145.0, 0.0, 0.0, 90.0, 31.0])

    def arm_nod(self):
        print("nod")
        for i in range(3):
            self.media_ros.pub_arm([82.0, 89.0, 93.0, 93.0, 89.0, 32.0])
            sleep(0.5)
            self.media_ros.pub_arm([82.0, 89.0, 93.0, 33.0, 89.0, 32.0])
            sleep(0.5)
        self.media_ros.pub_arm([90.0, 145.0, 0.0, 0.0, 90.0, 31.0])



    def process(self, frame):
        frame = cv.flip(frame, 1)
        frame, lmList, bbox = self.hand_detector.findHands(frame)
        if len(lmList) != 0 and self.media_ros.Joy_active:
            threading.Thread(target=self.arm_ctrl_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)
        self.media_ros.pub_imgMsg(frame)
        return frame

    def arm_ctrl_threading(self, lmList,bbox):
        if self.event.is_set():
            self.event.clear()
            fingers = self.hand_detector.fingersUp(lmList)
            self.hand_detector.draw = False
            gesture = self.hand_detector.get_gesture(lmList)
            if gesture == "Yes":
                self.arm_status = False
                self.dance()
                sleep(0.5)
                self.init_pose()
                sleep(1.0)
                self.arm_status = True

            elif gesture == "OK":
                self.arm_status = False
                for i in range(3):
                    sleep(0.1)
                    self.media_ros.pub_arm([80, 135, 0, 15, 90, 180])
```

```
                    sleep(0.5)
                    self.media_ros.pub_arm([110, 135, 0, 15, 90, 90])
                    sleep(0.5)
                self.init_pose()
                sleep(1.0)
                self.arm_status = True
            elif gesture == "Thumb_down":
                self.arm_status = False

                self.media_ros.pub_arm([90, 0, 180, 0, 90, 180])
                sleep(1.0)
                self.media_ros.pub_arm([90, 0, 180, 0, 90, 90])
                sleep(1.0)
                self.init_pose()
                sleep(1.0)
                self.arm_status = True


            elif sum(fingers) == 1:
                self.arm_status = False

                self.arm_nod()
                self.init_pose()
                sleep(1.0)
                self.arm_status = True


            elif fingers[1] == fingers[4] == 1 and sum(fingers) == 2:
                self.arm_status = False

                self.shake()
                self.init_pose()
                sleep(1.0)
                self.arm_status = True

            elif sum(fingers) == 5:
                self.arm_status = False
                self.arm_applaud()
                self.init_pose()
                sleep(1.0)
                self.arm_status = True

            self.event.set()

if __name__ == '__main__':
    rospy.init_node('HandCtrlArm_node', anonymous=True)
    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()
    ctrl_arm.media_ros.pub_arm([90, 135, 0, 45, 90, 90])
    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()
```