15.Mediapipe recognition and tracking palm control

Note: There are relevant running codes on Raspberry Pi and Jetson nano, but due to differences in motherboard performance, they may not run so smoothly. The supporting virtual machines also have running environments and programs installed.If the experience on the motherboard is not good, you can remove the camera and plug 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 needs to be set up between the motherboard and the virtual machine. Please refer to the previous tutorial for setting up.

15.1. 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.

15.2. Start

15.2.1. Preparation before starting the program

If you have a notebook, use the camera on the notebook and unplug the connection between the robotic arm camera and the PC. If there is no camera on the PC, remove the camera from the robotic arm and fix it on the PC. **Note that when this program is running, the range of motion of the robotic arm is relatively large. There should be no other objects around the robotic arm to avoid being hit by the robotic arm.**

15.2.2. Program description

After the program is started and the camera captures the image, the robotic arm will follow the movement of the palm in the screen. Here it is important not to move the palm too fast, otherwise the image processing cannot keep up, causing the robotic arm to misrecognize and move randomly. **

15.3.2. Program startup

Method 1: Run on separate motherboard/virtual machine,

```
roscore
rosrun arm_mediapipe arm_driver.py
rosrun arm_mediapipe ArmCtrl.py
```

Method 2: Motherboard + virtual machine (needs to be equipped with a camera and set up multi-machine communication),

```
roscore #Host
rosrun arm_mediapipe arm_driver.py #The motherboard starts the underlying drive robotic arm
rosrun arm_mediapipe ArmCtrl.py #Virtual machine starts image processing
```

15.3.3.Source code

Source code location：/home/yahboom/dofbot_ws/src/arm_mediapipe/scripts/ArmCtrl.py

jetson-nano source code
location：/home/jetson/dofbot_ws/src/arm_mediapipe/scripts/ArmCtrl.py

Raspberry Pi source code location:
/home/dofbot/dofbot_ws/src/arm_mediapipe/scripts/ArmCtrl.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.Joy_active = True
        #self.media_ros.pub_arm([90, 135, 0, 45, 90, 90])
        self.event = threading.Event()
        self.event.set()
```

```
    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.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):
        fingers = self.hand_detector.fingersUp(lmList)
        self.hand_detector.draw = True

        angle = self.hand_detector.ThumbTOforefinger(lmList)
        value = np.interp(angle, [0, 70], [185, 20])
        indexX = (bbox[0] + bbox[2]) / 2
        indexY = (bbox[1] + bbox[3]) / 2
        print("indexX: ",indexX)
        if indexY > 400: indexY = 400
        elif indexY < 200: indexY = 200
        joint2 = -0.4 * indexY + 170
        joint3 = 0.05 * indexY + 25
        joint4 = -0.125 * indexY + 85
        if 300 < indexX < 340: joint1 = 90
        else: joint1 = -0.3 * indexX + 200
        print("joint1: ",joint1)
        #self.media_ros.pub_arm([joint1, joint2, joint3, joint4, 90, value])
        self.media_ros.pub_arm([],1,joint1,200)
        sleep(0.1)

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()
    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()
```