11. Gesture Recognition

11.1. 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 (such as Raspberry Pi), 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 capabilities of the framework.
- Free and open source: framework and solution under Apache2.0, fully extensible and customizable.

11.2. Gesture Recognition

Gesture recognition designed for the right hand, which can be accurately recognized when certain conditions are met. The recognizable gestures are: [Zero, One, Two, Three, Four, Five, Six, Seven, Eight, Ok, Rock, Thumb_up (Like), Thumb_down (Thumbs down), Heart_single (Single-hand heart)], a total of 14 categories. The recognized gesture results will also be displayed on the oled.

Source code location:

/home/pi/project_demo/07.AI_Visual_Recognition/mediapipe/11.gesture_recognition/11_gesture_recognition.ipynb

```
#导入相关的模块   Import related modules
import threading
import cv2
import time
import math
from time import sleep
import ipywidgets.widgets as widgets
# 导入oled屏幕库 Import oled screen library
```

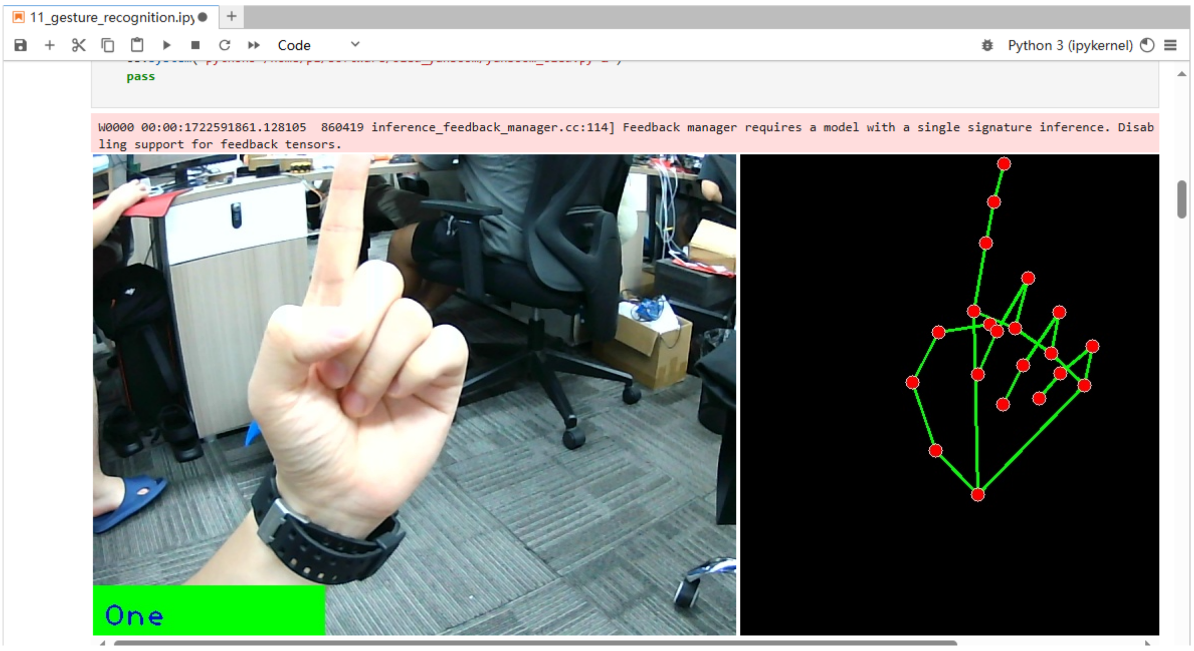
```python
import sys
sys.path.append('/home/pi/software/oled_yahboom/')
from yahboom_oled import *
# 创建oled对象 Create an oled object
oled = Yahboom_OLED(debug=False)

from gesture_action import handDetector
```

```python
g_camera = cv2.VideoCapture(0)
g_camera.set(3, 320)
g_camera.set(4, 240)
g_camera.set(5, 30)  #设置帧率 Setting the frame rate
# g_camera.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter.fourcc('M', 'J', 'P', 'G'))
# g_camera.set(cv2.CAP_PROP_BRIGHTNESS, 40) #设置亮度 -64 - 64  0.0 Set Brightness
-64 - 64 0.0
# g_camera.set(cv2.CAP_PROP_CONTRAST, 50) #设置对比度 -64 - 64  2.0 Set Contrast
-64 - 64 2.0
# g_camera.set(cv2.CAP_PROP_EXPOSURE, 156) #设置曝光值 1.0 - 5000  156.0 Set the
exposure value 1.0 - 5000 156.0
```

```python
hand_detector = handDetector(detectorCon=0.75)
image_original = widgets.Image(format='jpeg', width=640, height=480)
image_result = widgets.Image(format='jpeg', width=640, height=480)
image_widget = widgets.HBox([image_original, image_result])
```

```python
#bgr8转jpeg格式 bgr8 to jpeg format
def bgr8_to_jpeg(value, quality=75):
    return bytes(cv2.imencode('.jpg', value)[1])
```

```python
display(image_widget)
try:
    oled.init_oled_process() #初始化oled进程 Initialize oled process
    while True:
        ret, frame = g_camera.read()
        frame, img = hand_detector.findHands(frame, draw=False)
        if len(hand_detector.lmList) != 0:
            finger_number = hand_detector.get_gesture()
            cv2.rectangle(frame, (0, 430), (230, 480), (0, 255, 0), cv2.FILLED)
            cv2.putText(frame, str(finger_number), (10, 470),
cv2.FONT_HERSHEY_PLAIN, 2, (255, 0, 0), 2)
            finger_str=f"Number:{finger_number}"
            print(finger_str)
            oled.clear()
            oled.add_line("Gesture_Reco",1)
            oled.add_line(finger_str, 3)
            oled.refresh()
        else:
            oled.clear()
            oled.add_line("Gesture_Reco",1)
            oled.add_line("Number:None",3)
            oled.refresh()
        image_original.value = bgr8_to_jpeg(frame)
        image_result.value = bgr8_to_jpeg(img)
except:
    print(" Program closed! ")
    # 恢复屏幕基础数据显示 Restore basic data display on screen
    os.system("python3 /home/pi/software/oled_yahboom/yahboom_oled.py &")
    pass
```

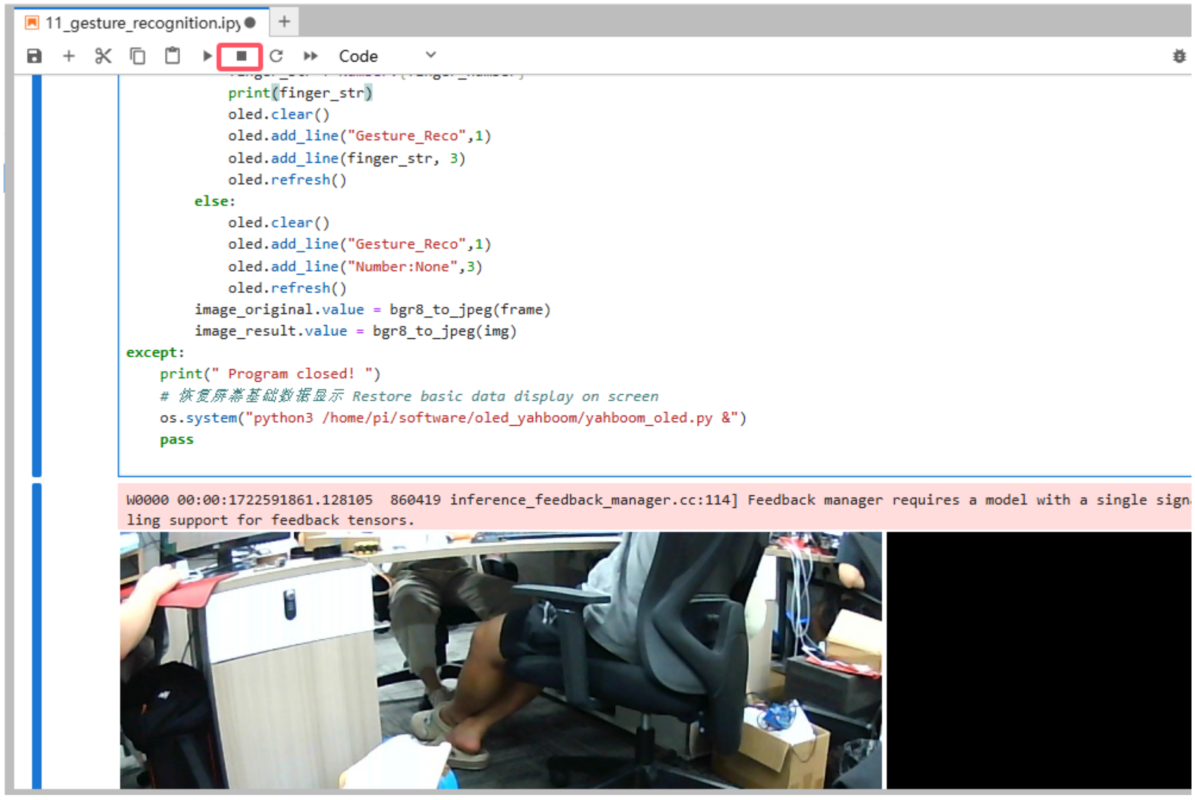


```
    pass
```

```
W0000 00:00:1722591861.128105  860419 inference_feedback_manager.cc:114] Feedback manager requires a model with a single signature inference. Disab
ling support for feedback tensors.
```

```
#使用完成对象记住释放掉对象，不然下一个程序使用这个对象模块会被占用，导致无法使用  Release
resources
g_camera.release()
```

You need to stop the program before releasing resources.



```
                print(finger_str)
                oled.clear()
                oled.add_line("Gesture_Reco",1)
                oled.add_line(finger_str, 3)
                oled.refresh()
            else:
                oled.clear()
                oled.add_line("Gesture_Reco",1)
                oled.add_line("Number:None",3)
                oled.refresh()
            image_original.value = bgr8_to_jpeg(frame)
            image_result.value = bgr8_to_jpeg(img)
except:
    print(" Program closed! ")
    # 恢复屏幕基础数据显示 Restore basic data display on screen
    os.system("python3 /home/pi/software/oled_yahboom/yahboom_oled.py &")
    pass
```

```
W0000 00:00:1722591861.128105  860419 inference_feedback_manager.cc:114] Feedback manager requires a model with a single sign
ling support for feedback tensors.
```