

# 目录

前言.....	2
一、项目概述.....	3
二、理论分析.....	3
(1) 项目基础分析 .....	3
(2) 项目移植分析 .....	4
(3) 另需说明 .....	5
三、树莓派与程序设计.....	5
1、树莓派的配置 .....	5
(1) 树莓派 4B 总体框图.....	5
(2) 树莓派 4B 的环境搭建.....	6
(3) 显示屏.....	7
(4) 电源.....	8
2、程序的设计 .....	8
(1) 程序功能描述与设计思路.....	8
(2) 程序流程图.....	10
四、测试方案与测试结果.....	11
1、测试方案 .....	11
2、测试结果及分析 .....	11
(1) 测试结果.....	11
(a) 三维人脸模型.....	11
(b) 电脑综合测试示例.....	12
(c) 树莓派综合测试示例.....	13
(d) 手掌检测.....	14
(e) 手掌距离与三维人脸模型交互.....	14
(2) 测试分析与结论.....	15
五、参考文献.....	16
附录.....	17
源程序: .....	17

## 前言

没有必要使用树莓派，但是我只是想要尝试下，这确实很蠢，而且没有必要。不管怎样，现在这个项目到此为止了。我很遗憾这个项目没有实现很理想的效果，但我已经尝试去做到了最好，这就是我想要说的。

## 一、项目概述

根据已有的三维人脸数据，开发一款基于普通投影仪或显示器和普通相机的人机交互系统，要求实现

1. 实现三维人脸的渲染和显示；
2. 能识别相机前实时采集的面部表情，比如大笑、沮丧等；能根据面部表情变化对应调整三维人脸数据；
3. 同时，此项目设计了一个创新的方向，同样是与三维人脸交互，但是是通过手掌来进行的。通过粗略识别摄像头到手掌的距离来使模型放大或缩小。

## 二、理论分析

### (1) 项目基础分析

最基本要求实现的功能主要有三个，分别是三维人脸的渲染和显示、人脸实时检测与表情识别和表情与三维人脸的交互变换。根据现已有知识储量，可以考虑的实现途径主要有三种，分别是 matlab、c++ 和 python。对于实现这些功能，必不可少的模块一般有 opengl、opencv 和 tensorflow(或 torch)，从实现难易程度来考虑，最终选择了 基于 python 去实现这些功能。

在查阅一定资料了解后，三维人脸的渲染和显示是本项目的最大难题，同时自己也要去收集相应的三维人脸模型。本次使用的数据集来自于 **facescape**，从中申请得到了三维人脸的数据模型。同时，也发现了一个很好用的 python 模块——**pygame**，将其和 **pyopengl** 一起使用，便可以很好地解决对应的三维人脸模型的显示。

其次，是人脸实时检测与表情识别的分析。使用 **opencv** 可以实现摄像头模块的调用和图像处理；同时设计图像处理，使用 **tensorflow** 加载对应的模型和权重。便可实现最基础的人脸实时检测和表情识别。人脸检测非常基础，在很多地方都有对应的模型使用，不再赘述；表情识别在并这里没有用深度学习训练一个这样的模型，个人水平有限，只是对其基础知识略有了解，所以这里采用了一个基于 **tensorflow** 的专门用于人脸特征分析的 python 模块——**deepface**，使用了其中的表情识别部分。由此，便可以实现人脸实时检测和表情识别。

之后要考虑的便是，两个功能的结合使用。关键在于两者的兼容性，在实际测试中，表情识别是要比三维人脸渲染和显示快很多的，主要问题在于三维人脸模型的变



### (3) 另需说明

另外一点想要阐述的是，本次项目其实是使用树莓派去移植原本在电脑上实现的功能，严格来讲在电路设计上的功夫基本没有。但本次项目会介绍树莓派上相应的配置操作。

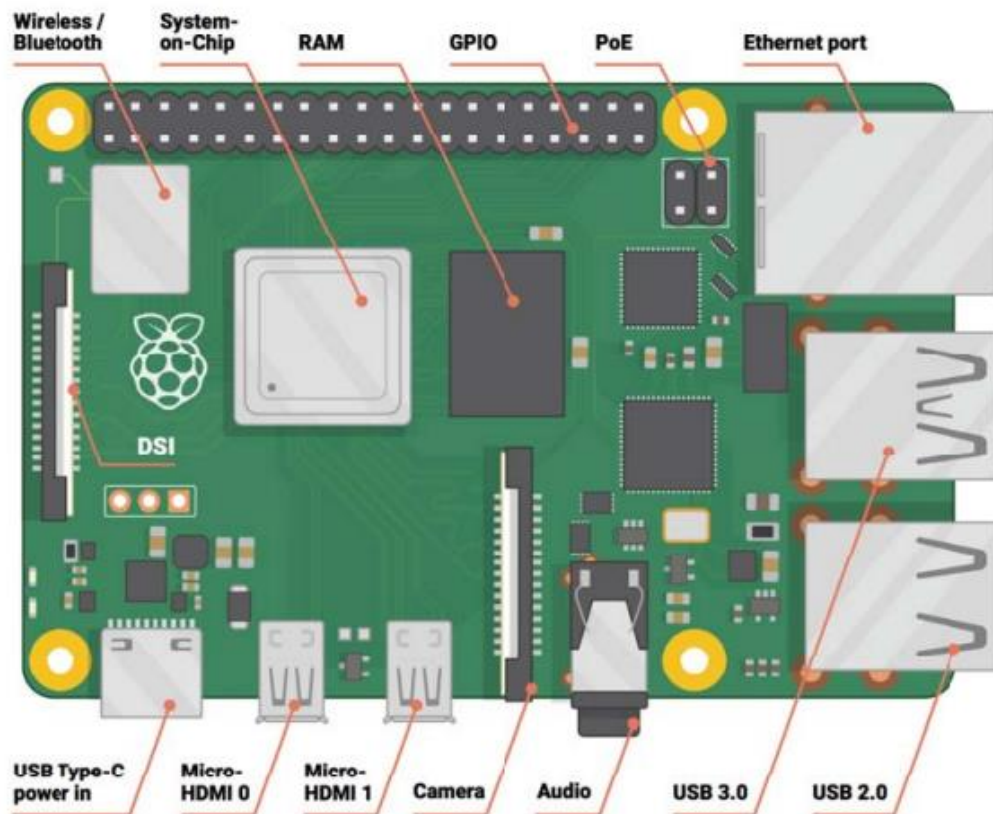
## 三、树莓派与程序设计

### 1、树莓派的配置

这里忽略了树莓派系统的安装。

#### (1) 树莓派 4B 总体框图

树莓派 4B 总体框图如下图所示，最为关键的部分在于 sd 卡读取、屏幕显示和摄像头模块的调用。



图(b) 树莓派 4B 总体框图

## (2) 树莓派 4B 的环境搭建

### 1、基础配置

#### (I) 更换软件源（这里使用的是清华的源）

① `sudo nano /etc/apt/sources.list` 注释掉原来的源，加上

```
deb http://mirrors.tuna.tsinghua.edu.cn/raspbian/raspbian/ buster main non-free contrib rpi
```

```
deb-src http://mirrors.tuna.tsinghua.edu.cn/raspbian/raspbian/ buster main non-free contrib rpi
```

② `sudo nano /etc/apt/sources.list.d/raspi.list` 同样注释掉原来的源，加上

```
deb http://mirrors.tuna.tsinghua.edu.cn/raspberrypi/ buster main
```

```
deb-src http://mirrors.tuna.tsinghua.edu.cn/raspberrypi/ buster main
```

③ 更新软件源 `sudo apt-get update`

#### (II) 无线网络配置

通过修改配置文件的方式实现

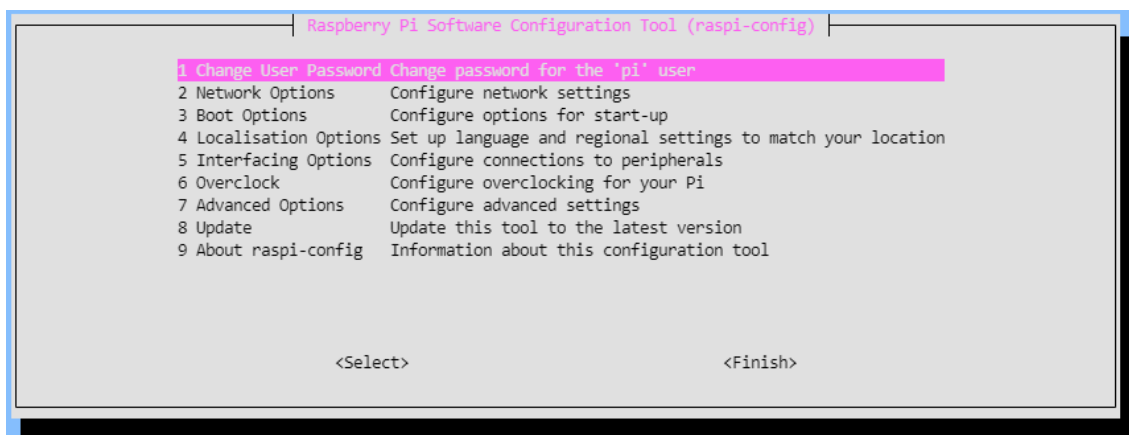
`sudo vim /etc/wpa_supplicant/wpa_supplicant.conf`

加上如下内容

```
network={
    ssid="ssid_name"
    key_mgmt=WPA-PSK
    psk="password"
}
```

#### (III) 其他配置

如打开 vnc 和摄像头，可以通过 `sudo raspi-config` 来实现。如下图(c)所示



图(c) 树莓派基础设置图

## 2、实现环境配置

### (I) 配置 python 环境，下载所需要的第三方库

pip 换源(也可以不换), 依次安装以下的第三方库

```
pip3 install pygame  
pip3 install deepface  
pip3 install PyOpenGL  
pip3 install numpy  
pip3 install opencv-python
```

### (II) 配置 opengl 的环境

```
sudo apt-get install build-essential # 基本编译环境  
sudo apt-get install libgl1-mesa-dev # opengl library  
sudo apt-get install libglu1-mesa-dev # opengl utilities  
sudo apt-get install freeglut3-dev # opengl utilities toolkit
```

### (III) 其它

为提高使用效率，也可以下载 vim 和 ranger

## (3) 显示屏

本项目的显示设置使用的是 3.5 寸屏幕。

驱动安装：

```
sudo rm -rf LCD-show  
git clone https://github.com/goodtft/LCD-show.git  
chmod -R 755 LCD-show  
cd LCD-show/  
sudo ./LCD35-show
```

如果需要改变屏幕方向：

```
cd LCD-show/  
sudo ./rotate.sh 90
```

执行完毕之后，系统会自动重启，然后显示屏就可以旋转 90 度正常显示和触摸

（‘90’也可以改为 0, 90, 180, 270 等数值，分别代表旋转角度 0 度，90 度，180 度，270 度）

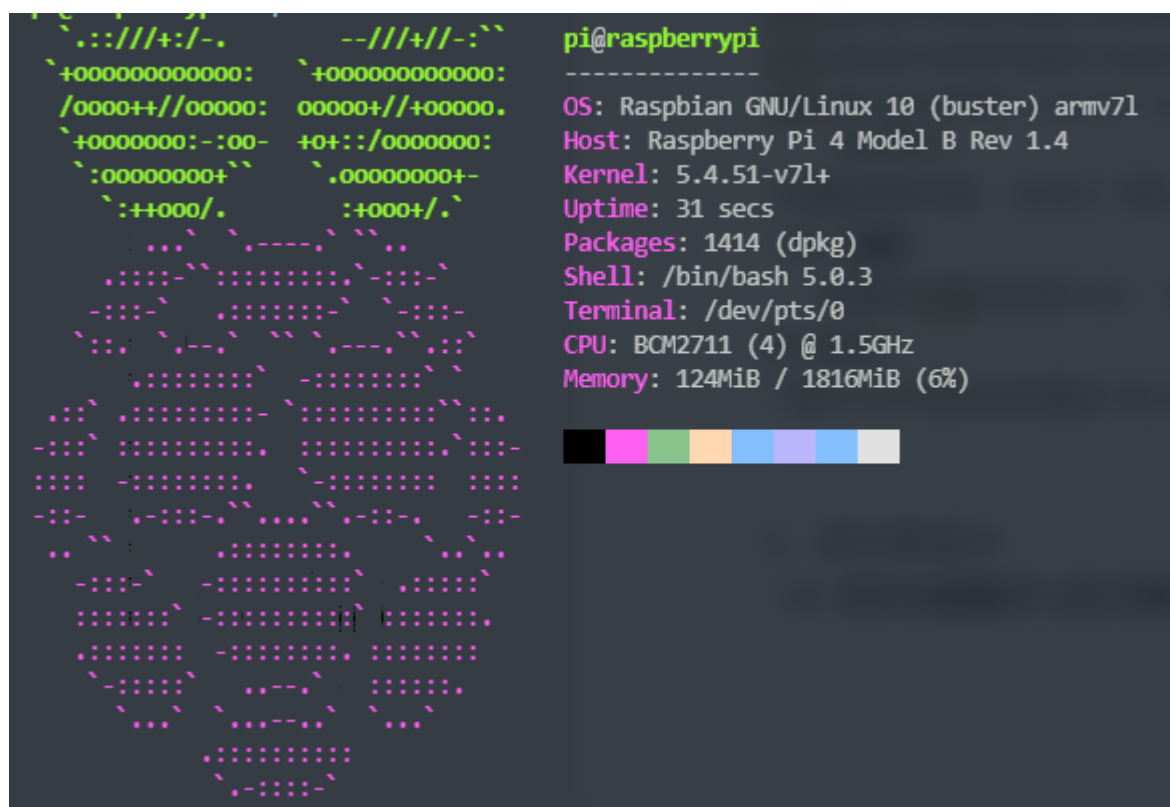
如果提示 rotate.sh 找不到，请回到第一步，安装最新的驱动

如果是 HDMI 接口显示屏使用 Raspberry Pi 4，需要先把 config.txt 文件中的 dtoverlay=vc4-fkms-V3D 注释掉（config.txt 文件位于 Micro SD 卡根目录，即 /boot 中）

#### (4) 电源

电源的考量并没有太多，只能能让树莓派 4B 正常工作就行。因此选择了最为方便的充电宝。

下图(d)为本次使用的树莓派 4B/2G 系统的基本信息



```
pi@raspberrypi
-----
OS: Raspbian GNU/Linux 10 (buster) armv7l
Host: Raspberry Pi 4 Model B Rev 1.4
Kernel: 5.4.51-v7l+
Uptime: 31 secs
Packages: 1414 (dpkg)
Shell: /bin/bash 5.0.3
Terminal: /dev/pts/0
CPU: BCM2711 (4) @ 1.5GHz
Memory: 124MiB / 1816MiB (6%)
```

The terminal window also displays a large heart shape constructed from ASCII characters on the left side of the screen.

图(d) 树莓派系统信息图

## 2、程序的设计

### (1) 程序功能描述与设计思路

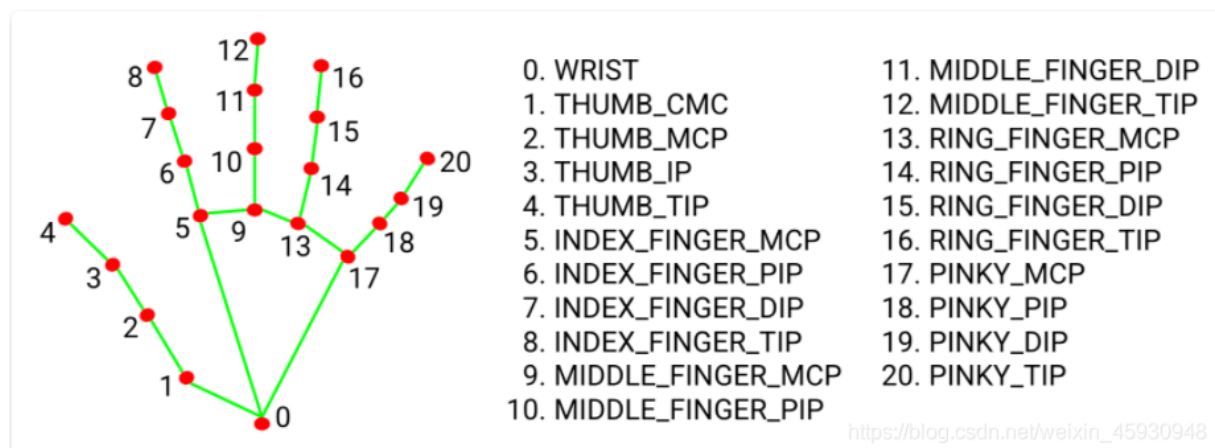
**(a)功能描述：**总体能够实现根据实时捕获到的人脸表情来调整三维人脸模型，并将其渲染并显示在屏幕上。**objloader.py** 用来加载模型和纹理；**light.py** 用来设置光照和相机；**man.py** 和 **main\_rpi.py** 用于显示程序，其中包括三维人脸模型的显示和人脸检测与表情识别。**obj\_show.py** 用于单独的三维人脸模型显示，可以鼠标左键按下旋转，鼠标右键按下将三维模型在空间里移动，鼠标滑轮放大或缩小。**hand\_detector.py** 用于手掌交互，根据手掌距离摄像头的距离放大或缩小。



**(b)设计思路及细节：**人脸检测部分基于已加载的人脸识别模型进行识别，opencv 模块调用摄像头模块并处理实时捕获到的图像，再根据识别结果绘制人脸检测的矩形框图。同时，将捕获到的人脸图像交由 `deepface.Deepface.analyze()` 处理(这个不止可以用来识别表情，还可以分析人脸的其他信息如年龄和性别，但这里只用到了表情分析)，返回表情识别结果，表情识别结果包括 "angry, fear, sad, disgust, happy, surprise, neutal", 最后将识别结果打印在打开的窗口上。

三维人脸模型的渲染和显示主要用的模块为 `pygame`, `pyopengl`, `numpy`, `pygame` 用于创建三维模型显示的窗口，`pyopengl` 用于加载三维人脸模型，同时也能调整光线和颜色。

手掌交互基于 `mediapipe` 实现，具体功能是根据手掌到摄像头的距离使三维模型放大或缩小。基于的原理是根据如手掌 图(e) 下 '5' 与 '17' 两点之间的距离和手掌到摄像头的实际距离拟合一个关于  $x$  与  $y$  的函数，然后基于这个函数去得到这个距离。毋庸置疑，这个函数不够准确，但不管怎样，本次实现也只有一个摄像头，不能实现很精确的相机定位，而且也只是需要一个趋势就够了。

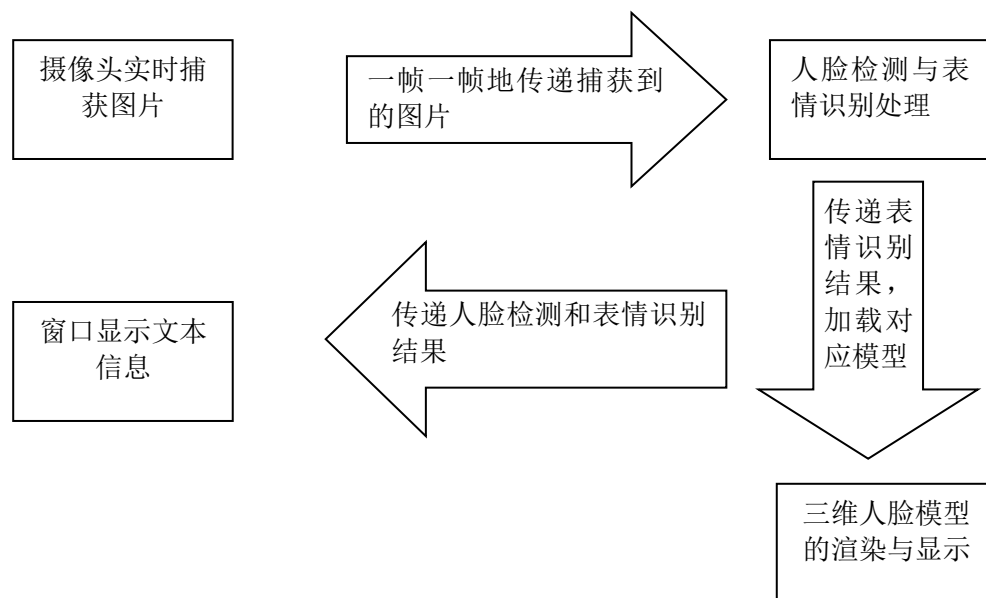


图(e) mediapipe 手掌识别信息图

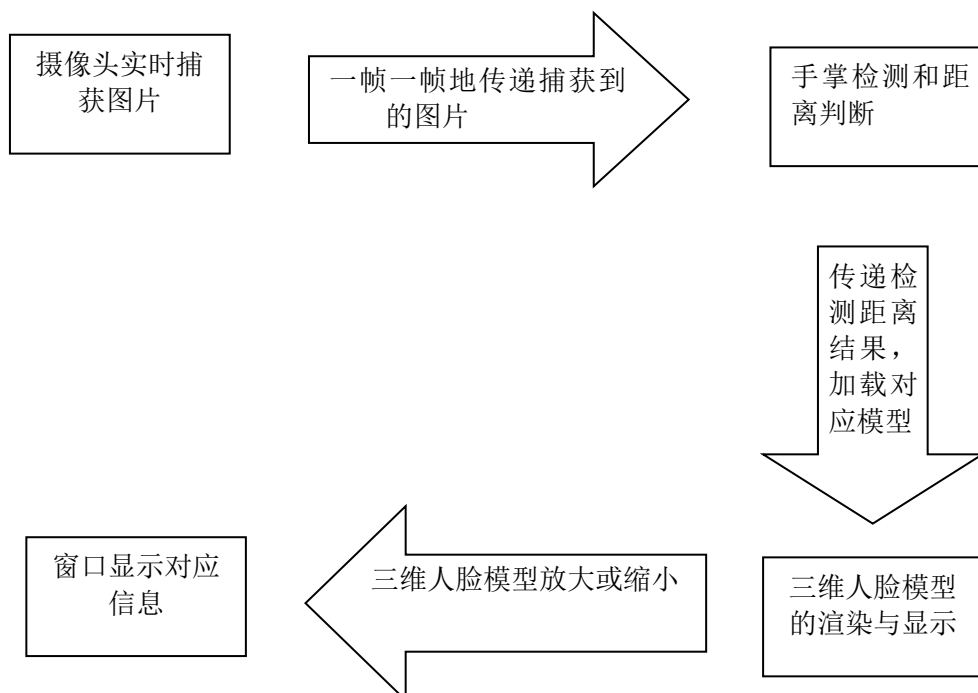
**(c)使用的 python 第三方库：**`pygame`, `pyopengl`, `numpy`, `opencv-python`, `deepface`, `tensorflow`, `mediapipe`, `cvzone`, `os`, `sys`

## (2) 程序流程图

### (a) 主程序流程图



### (b) 手掌交互子程序流程图



## 四、测试方案与测试结果

### 1、测试方案

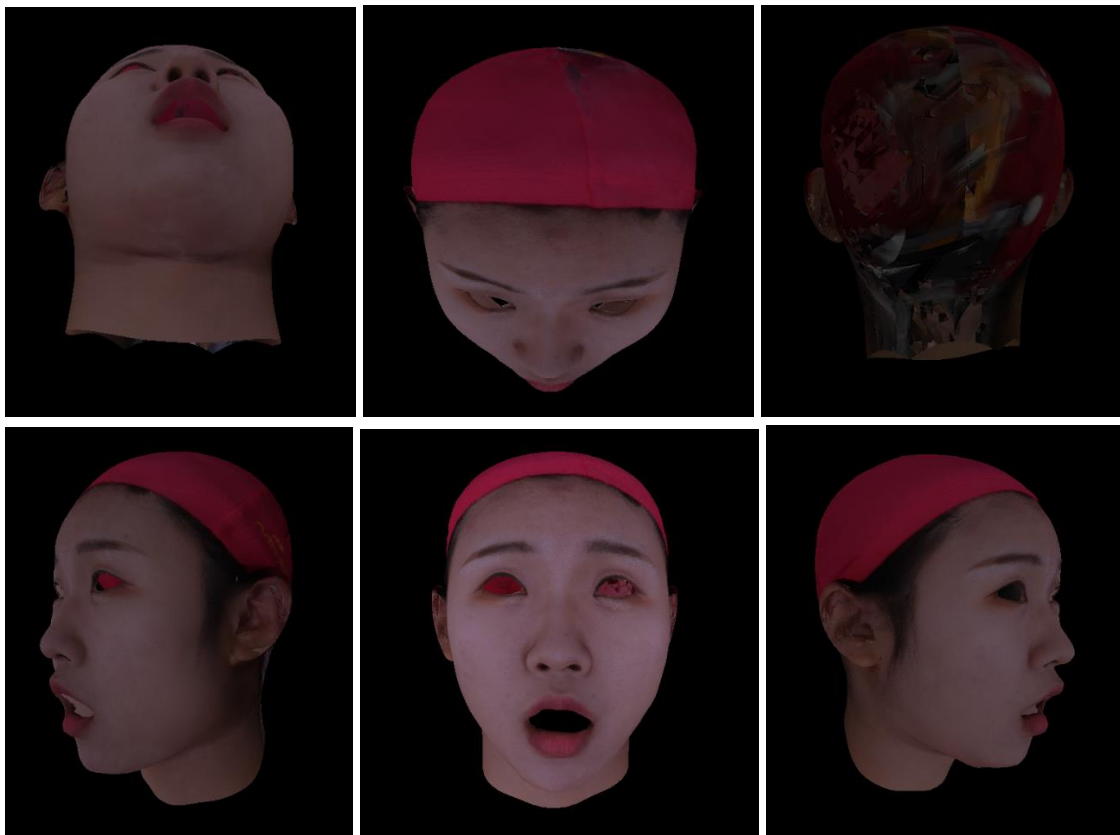
(1) 电脑测试：运行相应程序展示结果

(2) 树莓派测试：运行相应程序展示结果

### 2、测试结果及分析

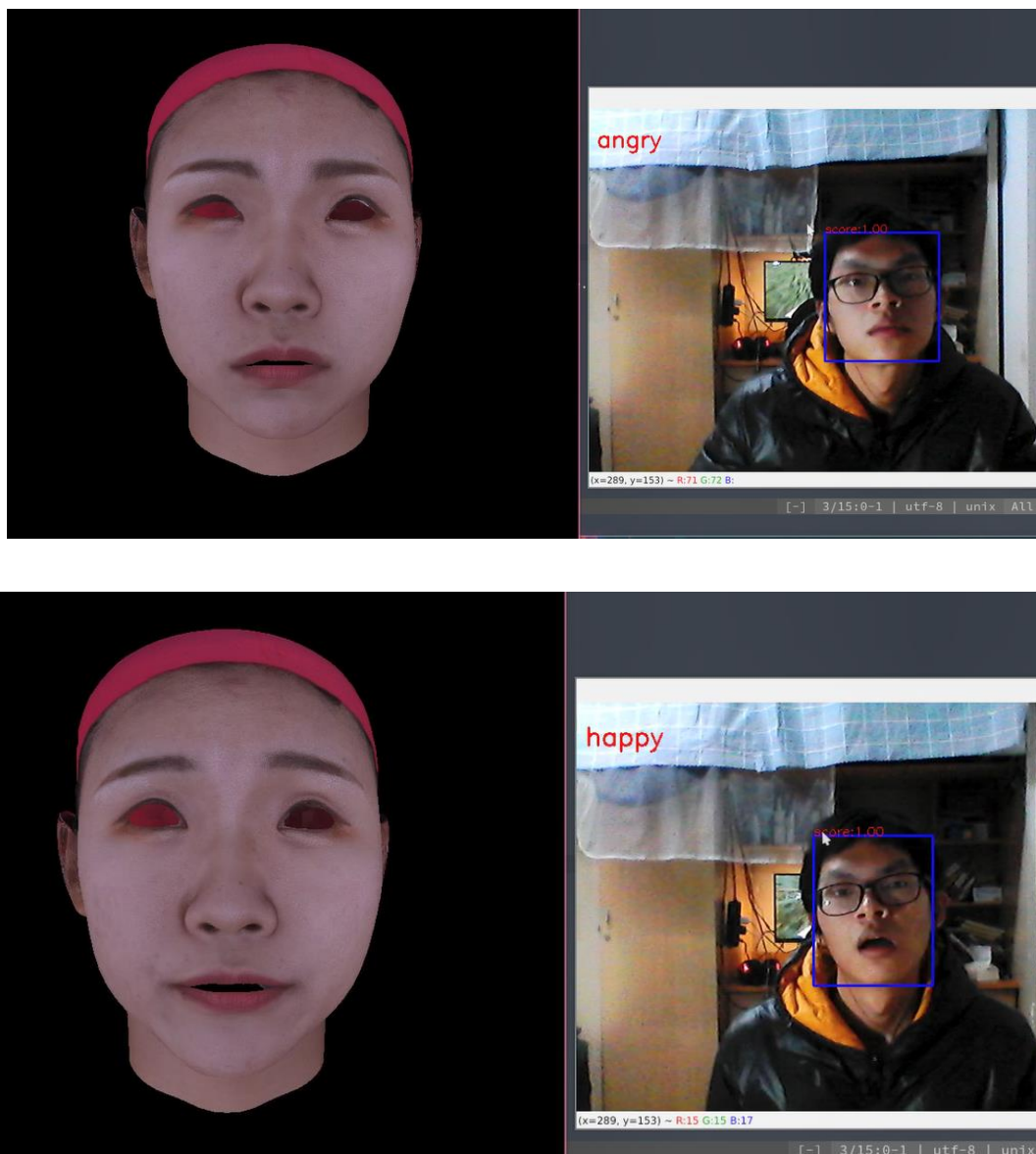
(1) 测试结果

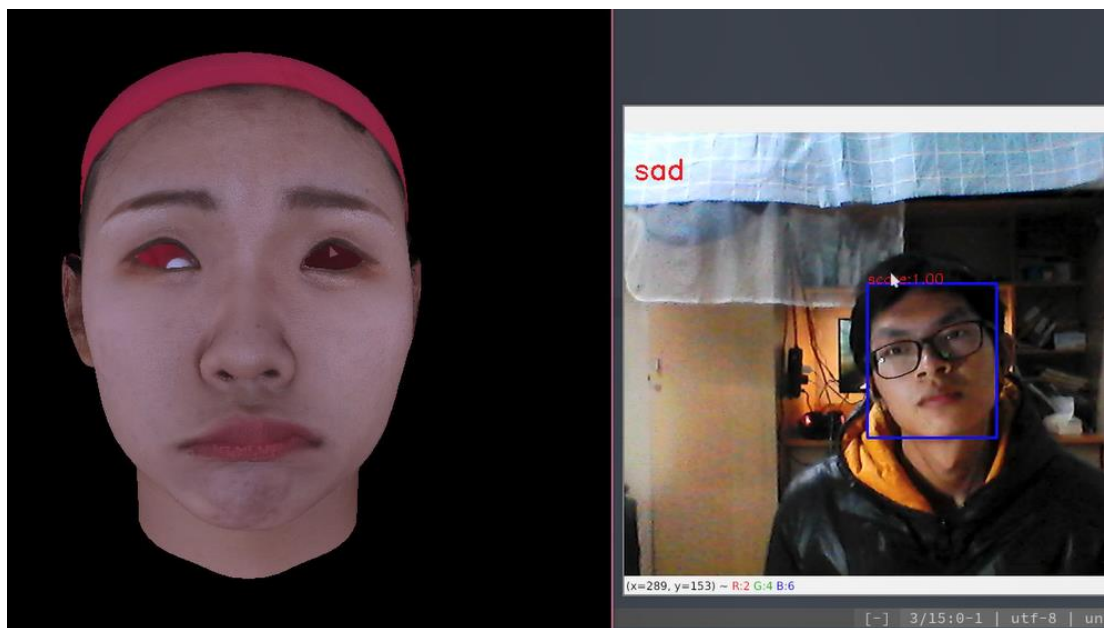
(a) 三维人脸模型



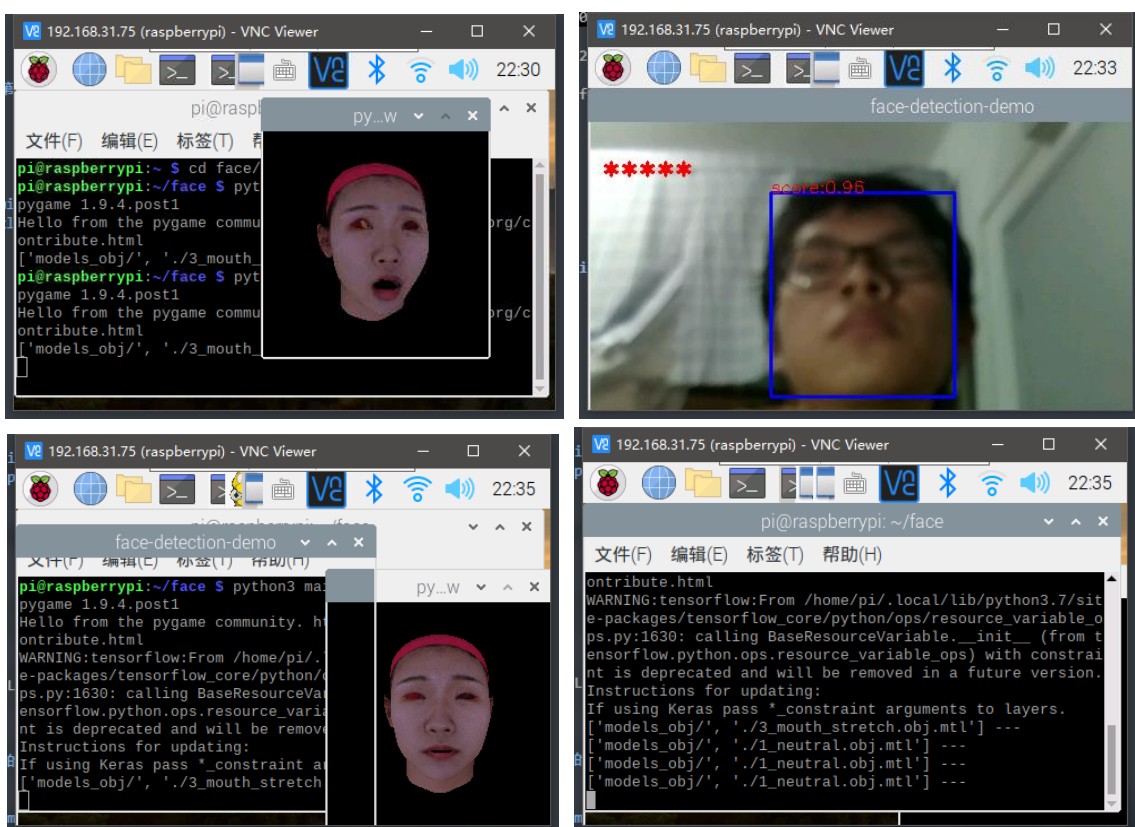


(b) 电脑综合测试示例



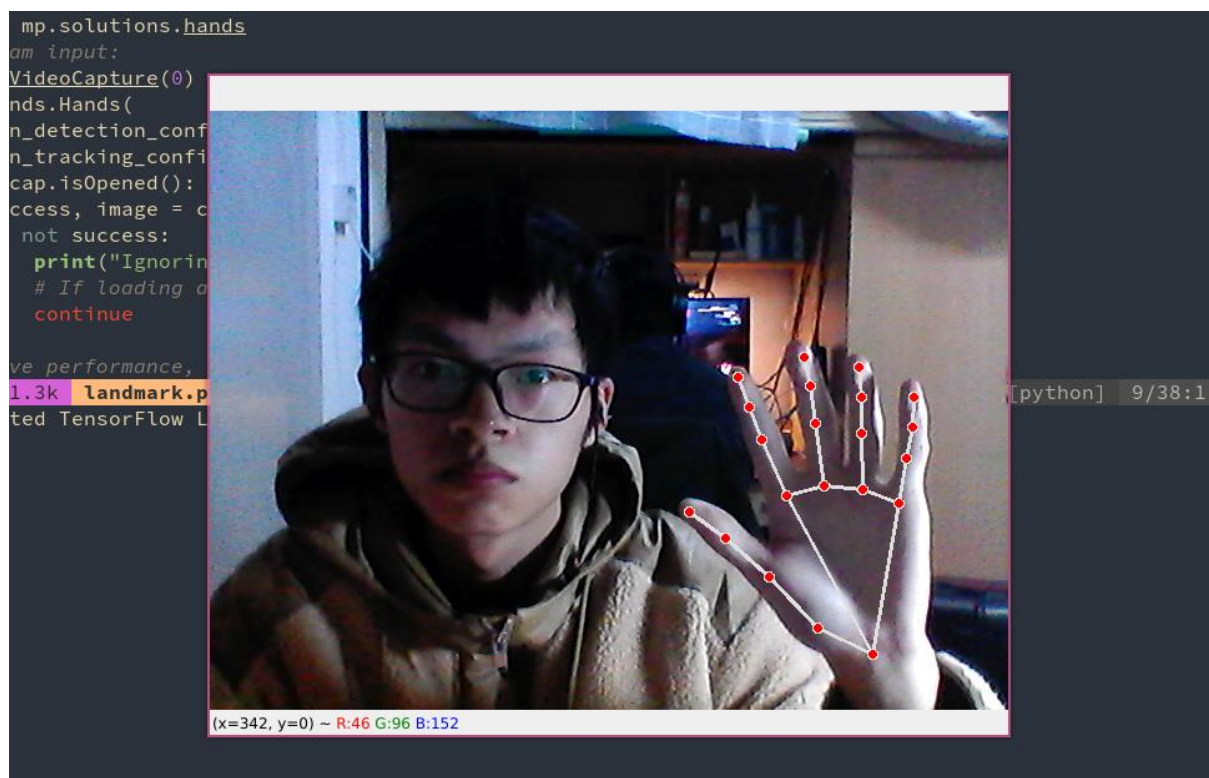


(c) 树莓派综合测试示例

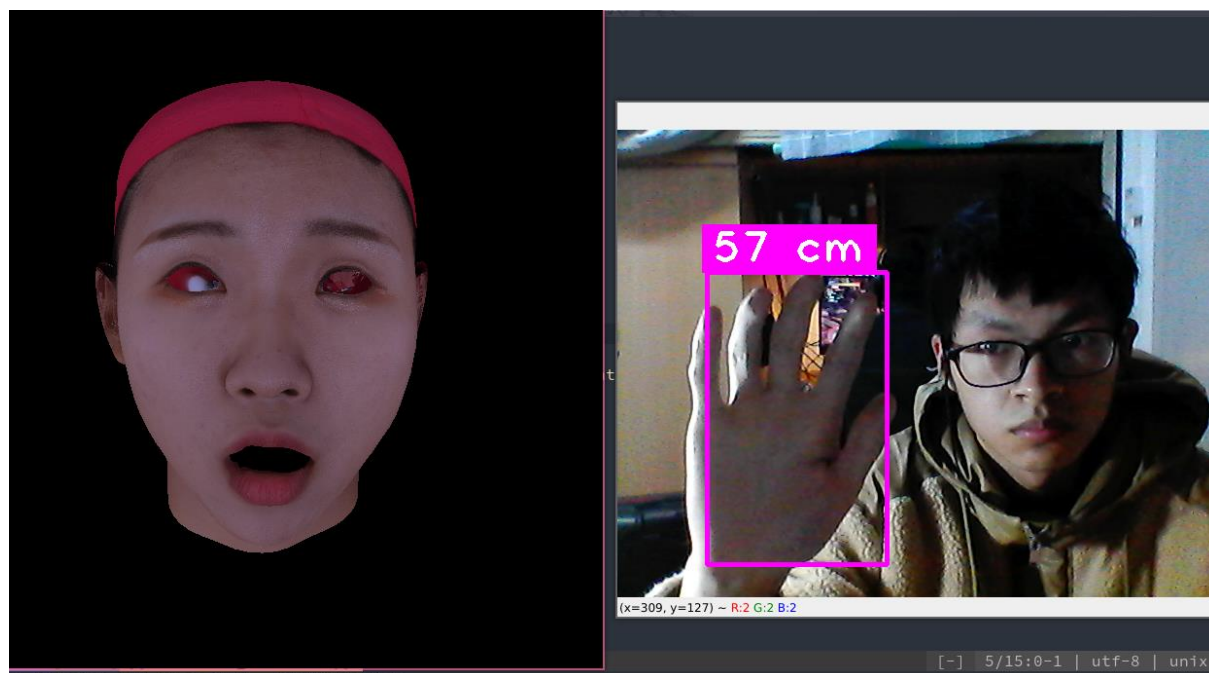




(d) 手掌检测



(e) 手掌距离与三维人脸模型交互





## (2) 测试分析与结论

根据上述测试数据，原系统在加载三维人脸模型和人脸检测上基本没有问题，三维人脸模型可以实现转动和移动。人脸检测基本准确。不仅如此，三维人脸模型根据识别到的人脸表情也能够进行变化。

但人脸表情识别略有误差，存在着表情识别不准，亮度影响等问题。同时，在两个功能结合运行时，由于三维模型渲染显示的速度略慢与表情识别的速率，会导致系统的实时性很差，表情识别和对应的三维模型可能有一定概率不相应；另外由于其而发生的问题还有在于识别的流畅性不行，图片的显示是一帧一帧的。

最后在树莓派上的实现略有不足，如上图所示，当主程序在树莓派运行时，摄像头实时捕获窗口无法显示，但根据终端信息，程序又确实在执行，而且三维人脸模型也能够变化，同样也具有卡顿的问题，这和树莓派的处理速度有关。两个功能分开单独运行效果还行，但依然存在卡顿的问题，而且摄像头的像素或识别精度也存在问题。这是本次项目的不足之处，也许可以通过优化代码加快三维模型的加载，比如使用了 pickle 模块实现 python 对象的持久化存储，当识别到相同表情时，只用加载缓存的文件，这样能比从零加载快不少。但目前未有显著的提升效果，所有树莓派运行该程序依然勉强。

手掌交互程序是独立于主程序进行的，因为主程序要不断加载显示三维人脸模型，所以很难再直接添加更多的操作到上面去。但是因为在手掌交互程序中三维人脸模型只加载了一次，它能够很流畅地运行。可以自由地通过手掌与摄像头之间的距离控制三维人脸模型的放大或缩小。可惜的是，mediapipe 在树莓派上没有对应的版本，所以树莓派上并不能够运行这个程序。

在根据由此可以得出以下结论：

- 1、三维人脸模型的渲染和显示基本实现。
  - 2、人脸检测和表情识别能够实现，但准确度略低。
  - 3、能够根据实时采集到的人脸表情改变三维人脸模型，但流畅度有待优化。
  - 4、手掌交互程序能够很成功的实现，可以基于此设计更多的交互选项。
- 综上所述，本设计达到设计要求。

尽管我们实现了设计的最基本要求同时也有一定创新，但我们依然无法确定是否存在让程序更流畅执行的方法，怎样才能使三维模型的渲染和显示提高速度，占用更少的资源，同时当两个功能结合使用时，是否能够避免摄像头实时识别因为三维模型加载而出现的一帧一帧的卡顿现象，怎样才能能够实现完美的同步呢？这些都是值得我们思考的地方。最后我们想要说的是树莓派是否真的适合这个项目呢？它的运算速度显示是不适合，但是它的广泛兼容性又很难被替代，也许直接在电脑上执行这个项目更加实际，不管怎样，它依然是一个有用的学习工具。这就是我们想说的。

## 五、参考文献

- [1] changhongjian.pygame-show-obj[EB/OL].  
<https://github.com/changhongjian/pygame-show-obj>
- [2] pygame.wiki.OBJFileLoader[EB/OL].  
<https://www.pygame.org/wiki/OBJFileLoader>
- [3] opencv\_ai.opencv\_tutorial\_data[EB/OL].  
[https://gitee.com/opencv\\_ai/opencv\\_tutorial\\_data?\\_from=gitee\\_search](https://gitee.com/opencv_ai/opencv_tutorial_data?_from=gitee_search)



## 附录

### 源程序:

详见 `github`(<https://github.com/Jinx-FX/face>)