高级实训设计报告

目录

[第一章 项目概述](#_Toc10960_WPSOffice_Level1) [1](#_Toc10960_WPSOffice_Level1)

[第二章 项目人员组成及分工](#_Toc3454_WPSOffice_Level1) [2](#_Toc3454_WPSOffice_Level1)

[小组分工](#_Toc3454_WPSOffice_Level2) [2](#_Toc3454_WPSOffice_Level2)

[组内互评](#_Toc9618_WPSOffice_Level2) [2](#_Toc9618_WPSOffice_Level2)

[第三章 项目效果](#_Toc9618_WPSOffice_Level1) [2](#_Toc9618_WPSOffice_Level1)

[1. 树莓派输入关机重启](#_Toc21627_WPSOffice_Level2) [2](#_Toc21627_WPSOffice_Level2)

[2. 输入让树莓派显示中文](#_Toc19264_WPSOffice_Level2) [2](#_Toc19264_WPSOffice_Level2)

[3. 用Windows远程桌面连接树莓派](#_Toc24995_WPSOffice_Level2) [3](#_Toc24995_WPSOffice_Level2)

[4. 用树莓派搭建BT下载服务器](#_Toc24279_WPSOffice_Level2) [5](#_Toc24279_WPSOffice_Level2)

[5. 拿树莓派当FTP文件服务器](#_Toc21719_WPSOffice_Level2) [6](#_Toc21719_WPSOffice_Level2)

[6. 使用树莓派实现简易的人脸识别](#_Toc24049_WPSOffice_Level2) [7](#_Toc24049_WPSOffice_Level2)

[第四章 项目开发过程](#_Toc21627_WPSOffice_Level1) [8](#_Toc21627_WPSOffice_Level1)

[一、 下载、安装操作系统](#_Toc9602_WPSOffice_Level2) [8](#_Toc9602_WPSOffice_Level2)

[二、 开箱上手教程](#_Toc31139_WPSOffice_Level2) [8](#_Toc31139_WPSOffice_Level2)

[1、让树莓派显示中文](#_Toc3454_WPSOffice_Level3) [8](#_Toc3454_WPSOffice_Level3)

[2、用Windows远程桌面连接树莓派](#_Toc9618_WPSOffice_Level3) [10](#_Toc9618_WPSOffice_Level3)

[三、网络、通讯相关模块 1、用树莓派搭建BT下载服务器](#_Toc17007_WPSOffice_Level2) [12](#_Toc17007_WPSOffice_Level2)

[2、使用SFTP远程传输树莓派文件](#_Toc21627_WPSOffice_Level3) [14](#_Toc21627_WPSOffice_Level3)

[四、人脸检测](#_Toc1903_WPSOffice_Level2) [15](#_Toc1903_WPSOffice_Level2)

[主要工具](#_Toc19829_WPSOffice_Level2) [15](#_Toc19829_WPSOffice_Level2)

[具体思路](#_Toc8896_WPSOffice_Level2) [15](#_Toc8896_WPSOffice_Level2)

[实现步骤](#_Toc14716_WPSOffice_Level2) [17](#_Toc14716_WPSOffice_Level2)

[步骤一：安装OpenCV](#_Toc19295_WPSOffice_Level2) [17](#_Toc19295_WPSOffice_Level2)

[步骤二：安装Picamera](#_Toc1443_WPSOffice_Level2) [17](#_Toc1443_WPSOffice_Level2)

[步骤三：编写python代码](#_Toc28803_WPSOffice_Level2) [18](#_Toc28803_WPSOffice_Level2)

[第五章 项目总结](#_Toc19264_WPSOffice_Level1) [19](#_Toc19264_WPSOffice_Level1)

# 第一章 项目概述

工具：树莓派系统、显示屏、键盘、鼠标、网卡

开箱上手

功能一：[树莓派如何安全关机重启](http://shumeipai.nxez.com/2013/08/25/raspberry-pi-how-to-safely-shutdown-restart.html" \t "http://shumeipai.nxez.com/_blank)

功能二：[如何让树莓派显示中文](http://shumeipai.nxez.com/2016/03/13/how-to-make-raspberry-pi-display-chinese.html" \t "http://shumeipai.nxez.com/_blank)

功能三：[用Windows远程桌面连接树莓派的方法](http://shumeipai.nxez.com/2013/10/06/windows-remote-desktop-connection-raspberry-pi.html" \t "http://shumeipai.nxez.com/_blank)

网络有关的功能：

功能一：[用树莓派搭建BT下载服务器](http://shumeipai.nxez.com/2013/09/08/raspberry-pi-bt-download-servers.html" \o "用树莓派搭建BT下载服务器" \t "http://shumeipai.nxez.com/_blank)

功能二：[拿树莓派当FTP文件服务器](http://shumeipai.nxez.com/2013/09/07/use-the-remote-sftp-file-transfer-raspberry-pi.html" \o "使用SFTP远程传输树莓派文件" \t "http://shumeipai.nxez.com/_blank)

创新功能：使用树莓派进行简易人脸识别

# 第二章 项目人员组成及分工

## 小组分工

|  |  |  |
| --- | --- | --- |
| **姓名** | **学号** | **分工** |
| 刘勤富 | 15352235 | 搭建树莓派系统、完成level3功能、参与撰写设计报告 |
| 刘益智 | 15352243 | 实现人脸检测功能、参与撰写设计报告 |
| 刘子立 | 15352245 | 完成level3功能、录制视频、参与撰写设计报告 |
| 卢杨 | 15352248 | 实现“开箱上手”前三个功能、参与撰写设计报告 |

## 组内互评

|  |  |
| --- | --- |
| **成员** | **分数** |
| 刘勤富 | 100 |
| 刘子立 | 100 |
| 刘益智 | 100 |
| 卢杨 | 100 |

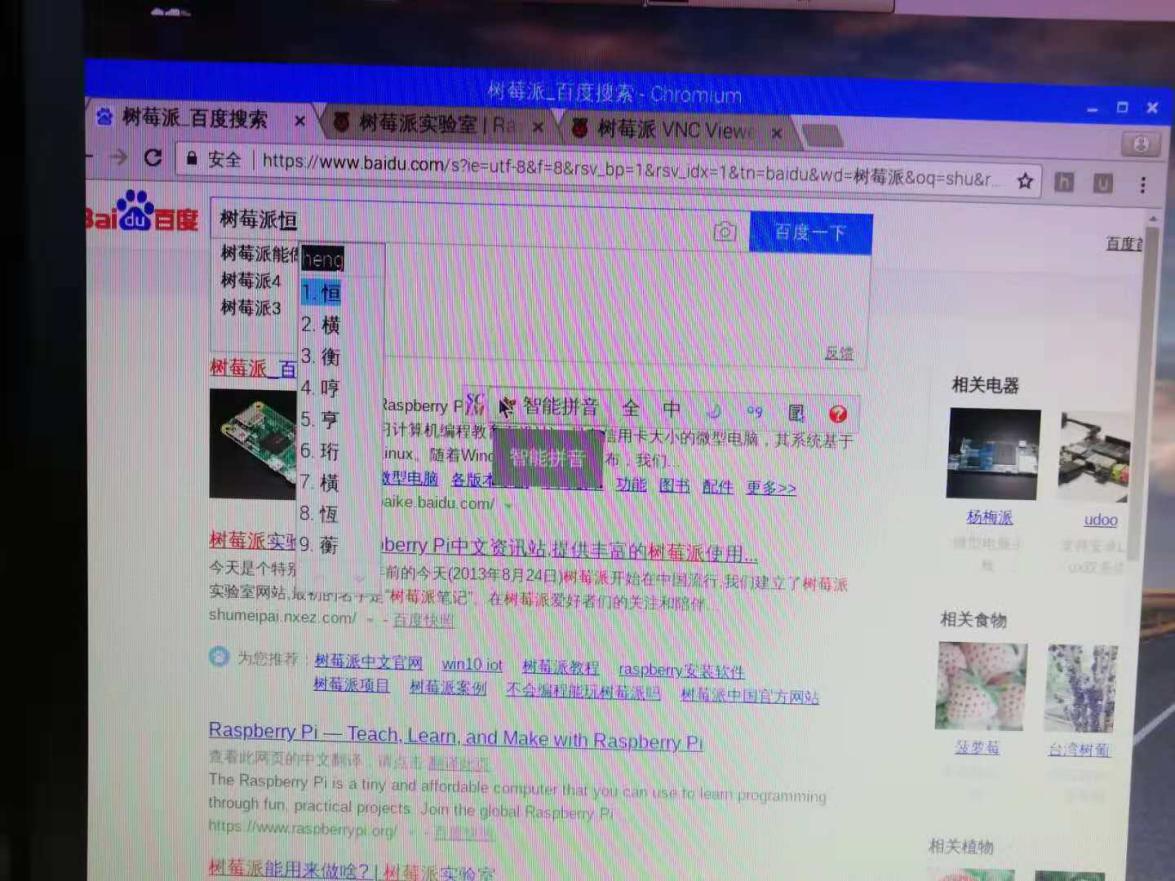
# 第三章 项目效果

1. 树莓派输入关机重启

略

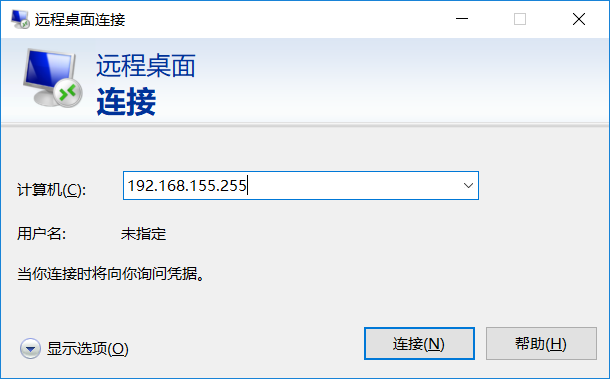
1. 输入让树莓派显示中文

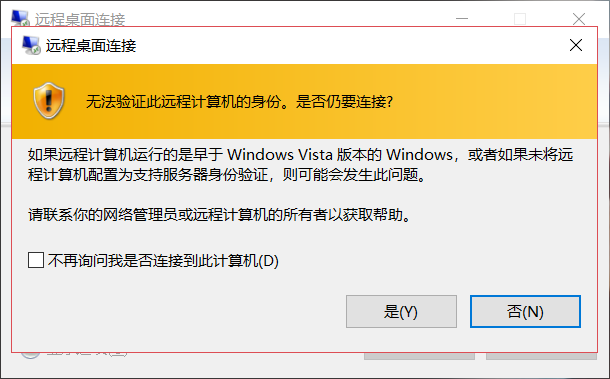
重启完成好就可以在VNC连接上去后使用中文显示和中文输入法了，切换中文输入法一样也是ctrl+space。然后在输入是可以切换输入法，选择中文输入



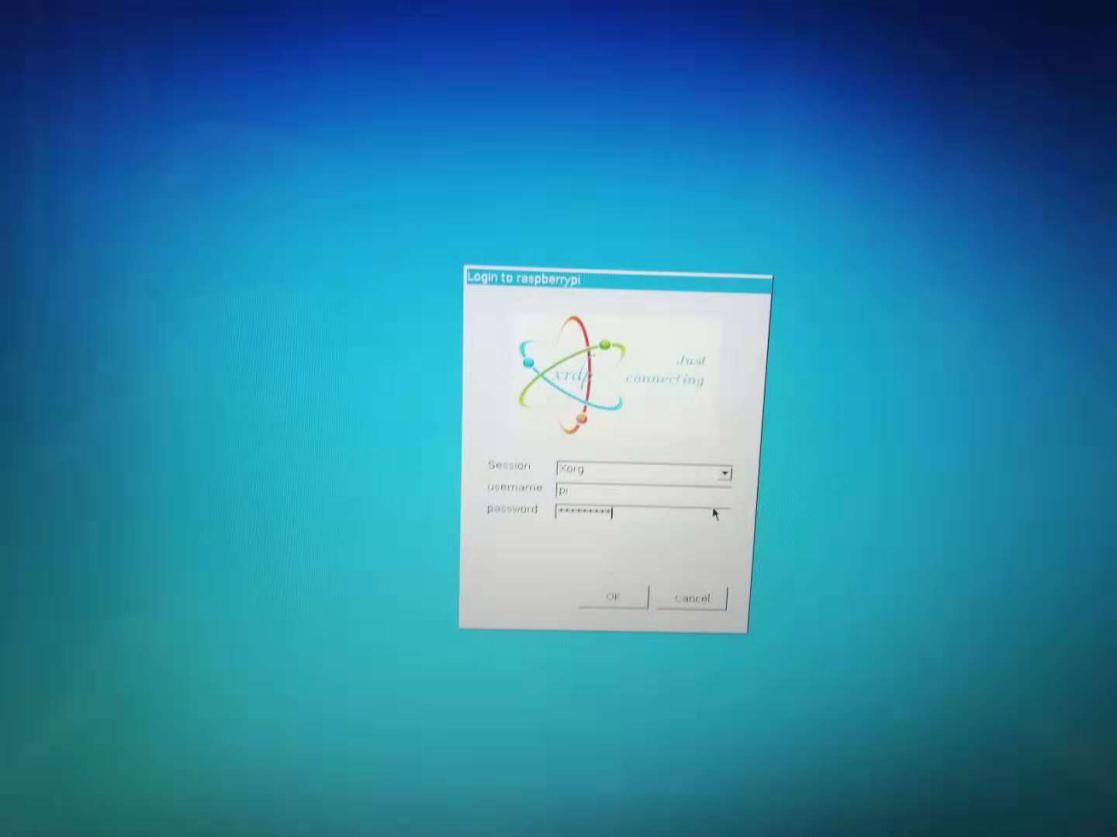
1. [用Windows远程桌面连接树莓派](http://shumeipai.nxez.com/2013/10/06/windows-remote-desktop-connection-raspberry-pi.html" \t "http://shumeipai.nxez.com/_blank)

树莓派通过终端输入指令：ifconfig 可知树莓派的主机地址为192.168.155.255，然后再Windows电脑使用远程连接程序键入IP地址

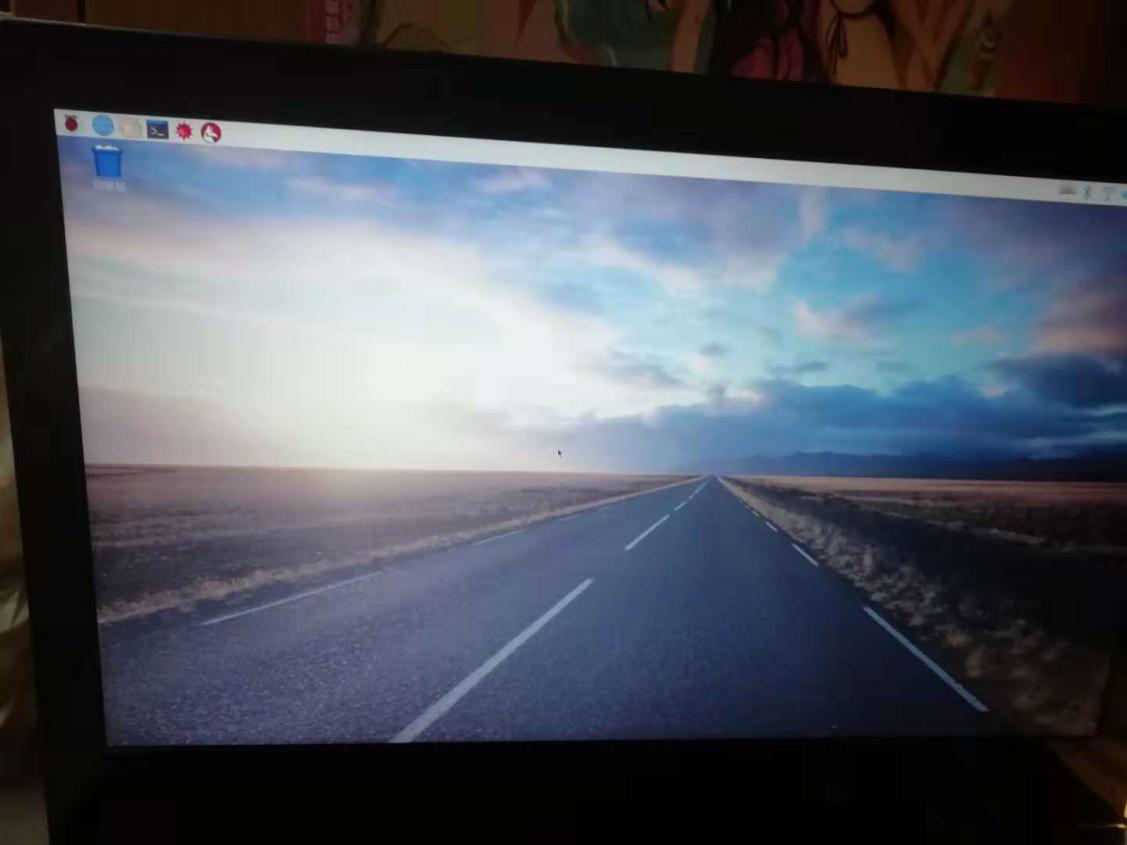




选择是。然后，连接成功，输入登录用户名和密码：如果未修改过密码，默认用户名为pi，密码为：raspberry

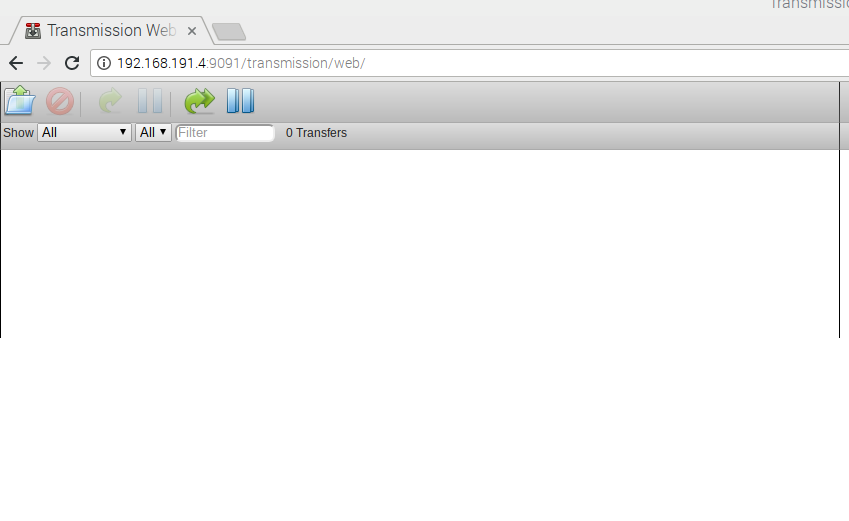


登录成功，实现远程控制

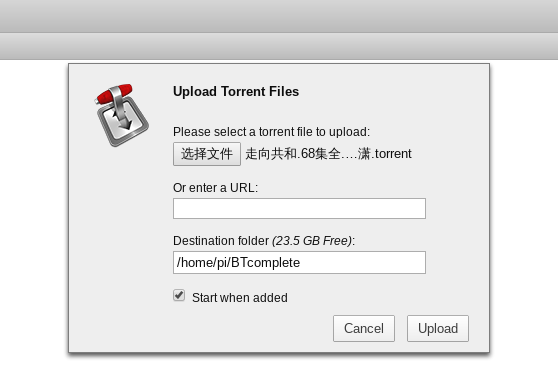


1. 用树莓派搭建BT下载服务器

在浏览器中访问 IP 加 9091端口：比如： http://192.168.191.4:9091/ 。访问时输入用户名和密码，默认都是：transmission 。进入以下界面



选择要下载的文件，并且选定下载的种子文件



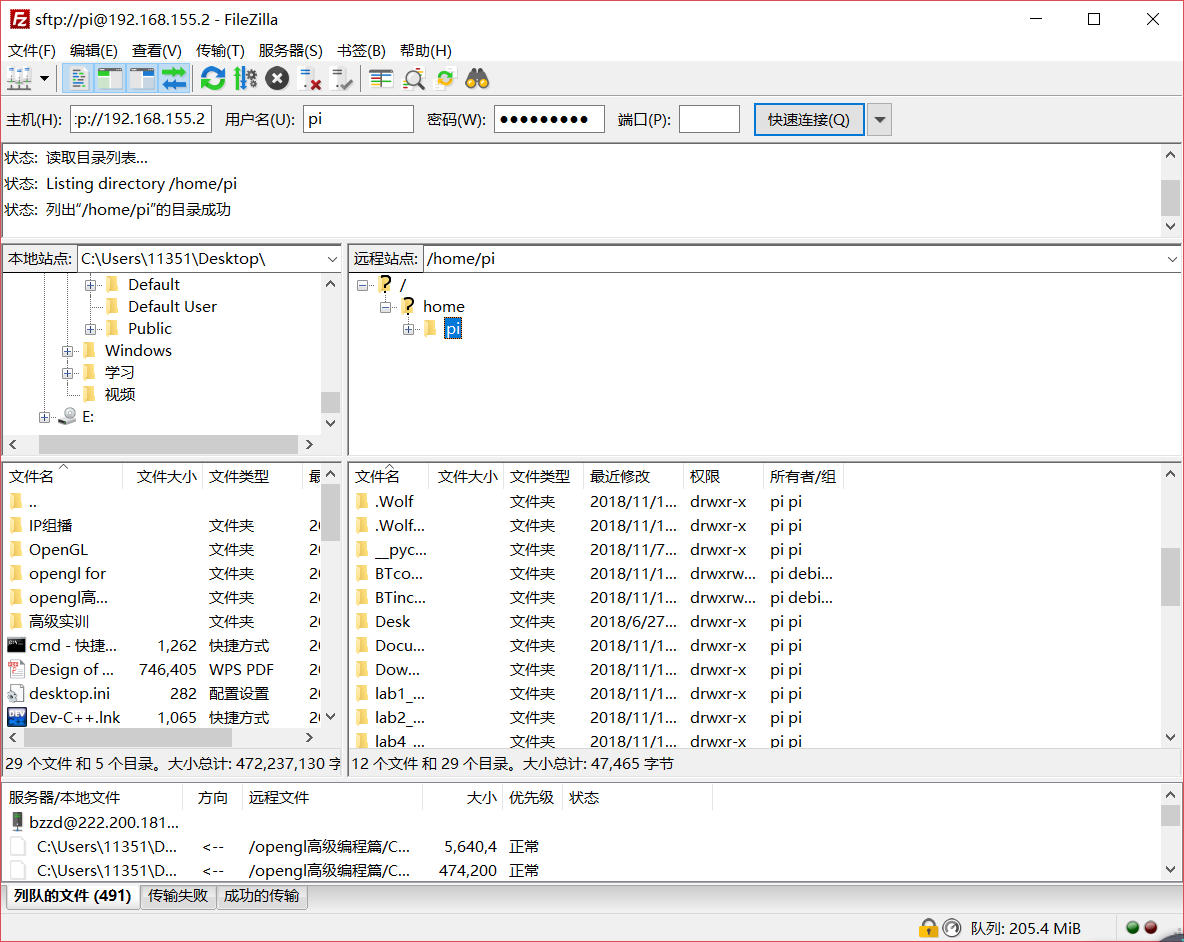
开始下载：



1. 拿树莓派当FTP文件服务器

这里我们使用FlieZilla进行实验：

只需在“快速连接”中输入：  
主机：sftp://192.168.155.2 （前面的sftp://一定要加）  
用户名和密码照实填。（Raspbian默认是pi/raspberry）

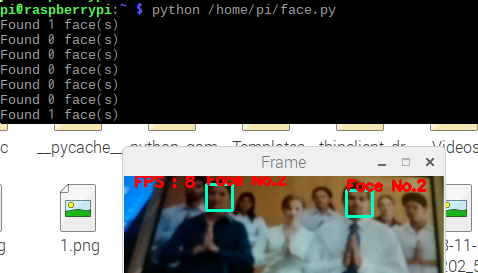


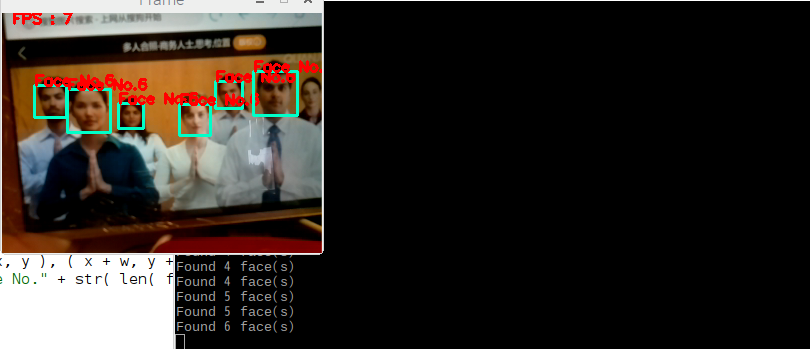
成功读取出树莓派文件。

1. 使用树莓派实现简易的人脸识别

终端运行face.py代码：python /home/pi/face.py

即可实现动态人脸检测，并在终端显示检测到的人脸数目。





# 项目开发过程

## 下载、安装操作系统

此处加500字+

1. 开箱上手教程

1、让树莓派显示中文

树莓派默认是采用英文字库的，而且系统里没有预装中文字库，所以即使你在locale中改成中文，也不会显示中文，只会显示一堆方块。因此需要我们手动来安装中文字体。好在有一个中文字体是免费开源使用的。ssh中输入以下命令：

sudo apt-get install ttf-wqy zenhei

安装过程中如果碰到(Y/n)，都选择y  
中文字库安装完成之后，还需要安装一个中文输入法。输入如下命令

sudo apt-get install scim-pinyin

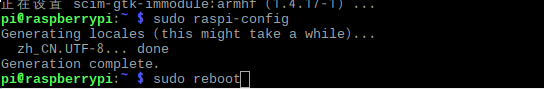


一样的安装过程，安装完毕后输入如下指令：

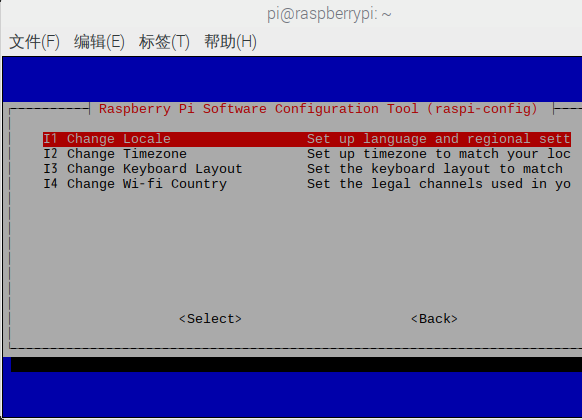
sudo raspi-config

然后选择change\_locale，在Default locale for the system environment:中选择zh\_CN.UTF-8,配置完成之后，输入命令

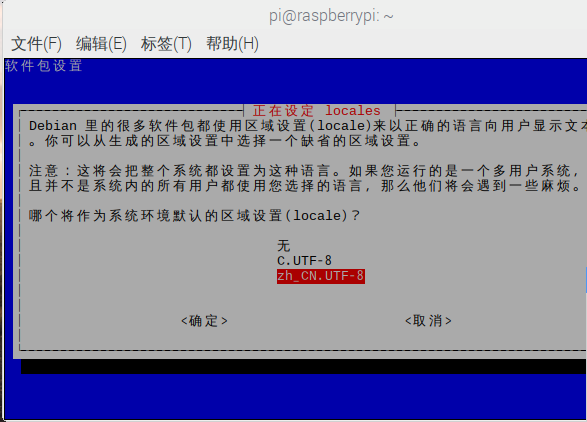
sudo reboot



Change locale:



设定locales, 选择zh\_CN.UTF-8



### 2、用Windows远程桌面连接树莓派

这个实验要求Windows计算机和树莓派在同一局域网上，实验分两步，第一步先在树莓派上开放远程连接的许可，再利用Windows电脑自带的远程桌面控制程序（mstsc.exe）连接树莓派。

第一步，先在树莓派raspbian文件下安装一个服务xdrp。

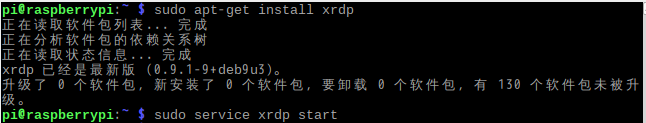
输入如下命令安装：

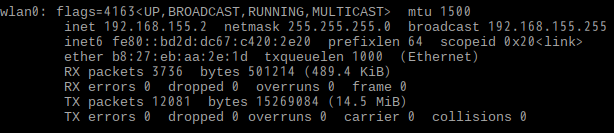
sudo apt-get install xrdp

再用如下命令激活该安装：

Sudo service xrdp start

在命令提示行中输入“ifconfig”查找树莓派的IP地址并记下。



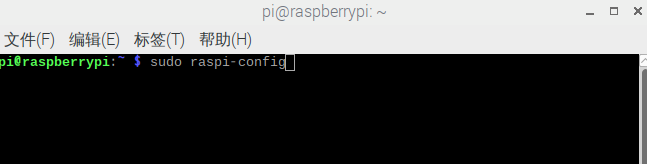


第二步，在自己的Windows系统的计算机上进行远程连接。

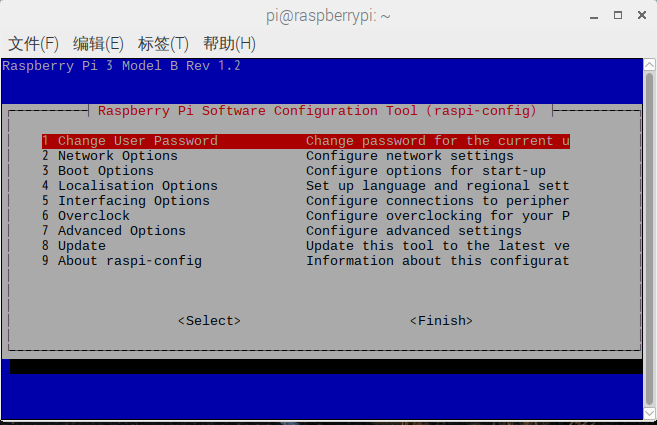
打开 开始-运行-输入mstsc，对话框弹出后输入树莓派的IP地址即可连接。

**另外，我们在做该功能时遇到一个小问题,就是忘记了用户名的密码，使得无法完成远程登陆验证，以下是解决的办法：**

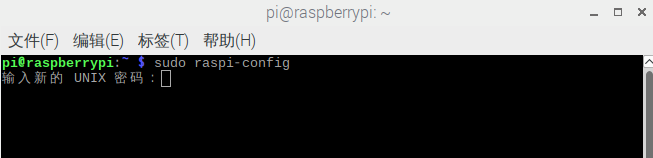
（1）输出指令：sudo raspi-config #该指令用来更改系统的设置，非常好用



（2）进入以下界面，选择第一项change User Password

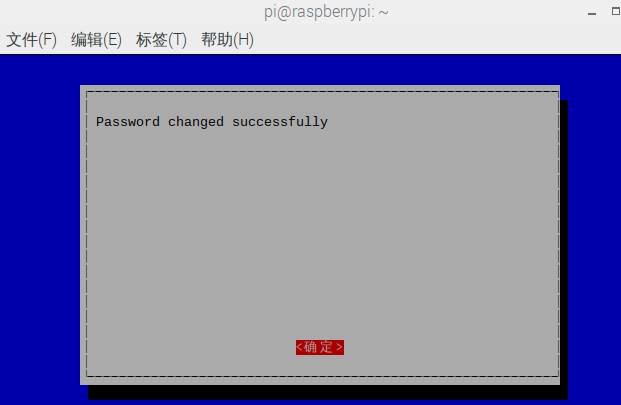


(3)重新更改密码



此处截图无法看见密码，因为该更改密码的界面输入后只会显示和背景颜色一样，只要两次输入的密码一致，修改即可通过。

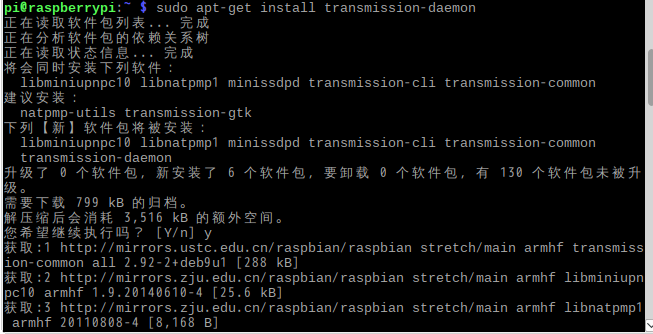
（4）出现以下界面说明更改成功。当然如果没有作过特别更改，第一次安装树莓派使系统默认用户名为pi,密码为raspberry



三、网络、通讯相关模块  
1、用树莓派搭建BT下载服务器

首先在树莓派上安装transmission：

sudo apt-get install transmission-daemon



然后我们要创建两个下载目录，一个是下载完的目录，一个是未完成的目录：

mkdir -p /home/pi/BTincomplete

mkdir /home/pi/BTcomplete

配置目录的权限：

sudo usermod -a -G debian-transmission pi

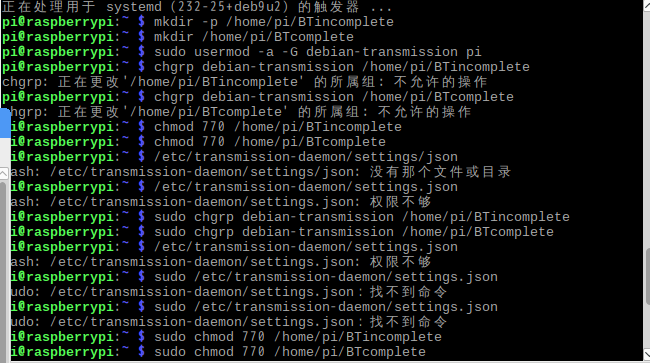
sudo chgrp debian-transmission /home/pi/BTincomplete

sudo chgrp debian-transmission /home/pi/BTcomplete

sudo chmod 770 /home/pi/BTincomplete

sudo chmod 770 /home/pi/BTcomplete

配置过程截图如下:

另外，在配置目录权限时，有几行代码呗提示权限不够，在代码前全都加上“sudo”授予权限即可执行。

接着修改配置文件：

sudo nano /etc/transmission-daemon/settings.json

这个json文件我们按照指导修改了下列项：

下载目录位置：

"download-dir": "/home/pi/BTcomplete",

未完成目录：

"incomplete-dir": "/home/pi/BTincomplete",

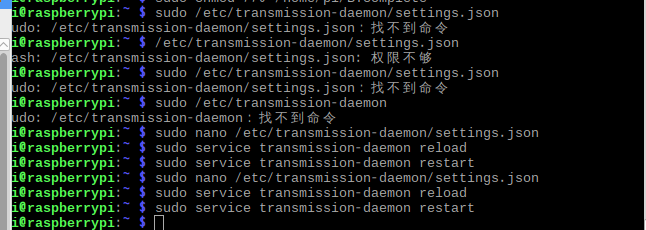
允许Web访问的白名单地址

"rpc-whitelist": "192.168.155.2\*",

最后，配置好之后重启 transmission按顺序执行下列两个指令，单独 restart 的话配置不会保存：

sudo service transmission-daemon reload

sudo service transmission-daemon restart



现在在浏览器中访问 IP 加 9091端口：比如： http://192.168.155.2:9091/（IP是树莓派的IP） 。访问时输入用户名和密码，默认都是：transmission。

如果想要修改用户名和密码，先停止服务：

sudo service transmission-daemon stop  
然后修改配置上面提到的json文件，下面两项分别是默认的用户名和密码：  
“rpc-username”: “transmission”,  
“rpc-password”: “{2dc2c41724aab07ccc301e97f56360cb35f8ba1fGVVrdHDX”

将两项改为新用户名和新密码即可  
再此启动服务：

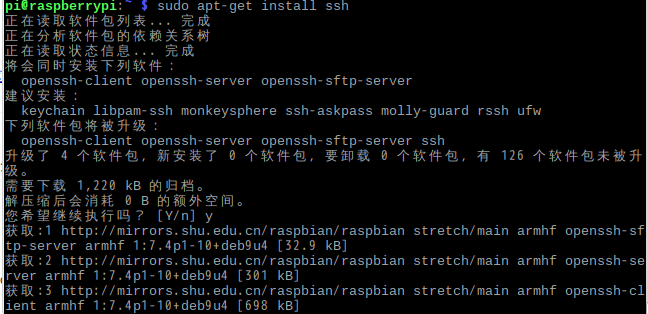
sudo service transmission-daemon start

新的用户名和密码即可生效。

### 2、使用SFTP远程传输树莓派文件

树莓派可用于实现SFTP功能，和FTP类似，实现上传/下载/管理文件的操作。很多FTP软件就“顺便”支持SFTP。  
推荐FileZilla。理由和PuTTY用汉化版一样——无需设置直接UTF-8编码，中文名文件上传树莓派不乱码。

在树莓派上输入指令进行配置：





现在可以在其他电脑通过Filezile实现快速连接，并读取树莓派系统下的文件

## 四、人脸检测

### 主要工具

软件：OpenCV、Picamera、python

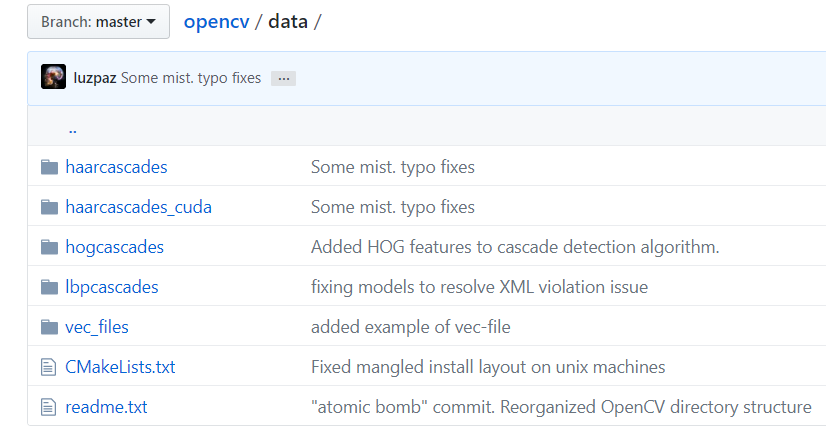
硬件：usb摄像头、树莓派系统

环境：python平台下运行

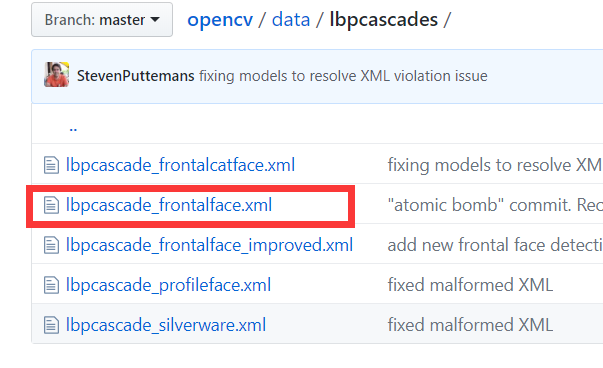
### 具体思路

人脸识别，是基于人的脸部特征信息进行身份识别的一种生物识别技术。用摄像机或摄像头采集含有人脸的图像或视频流，并自动在图像中检测和跟踪人脸，进而对检测到的人脸进行脸部的一系列相关技术，通常也叫做人像识别、面部识别。人脸检测大体分为三个步骤：1 人脸图像采集及检测 2 人脸图像预处理 3 人脸图像特征提取以及匹配与识别；在OpenCV中主要使用了两种特征（即两种方法）进行人脸检测，Haar特征和LBP特征，并使用已经训练好的XML格式的分类器进行人脸检测，在python语言中OpenCV的调用方式是**import cv2**，使用xml分类器的函数是**cv2.CascadeClassifier(“xml文件路径”)**。

在树莓派系统安装好OpenCV后，我们需要从GitHub库<https://github.com/opencv/opencv/tree/master/data>里下载需要的xml文件，如下图



文件夹的名字“haarcascades”、“hogcascades”和“lbpcascades”分别表示通过“haar”、“hog”和“lbp”三种不同的特征而训练出的分类器："haar"特征主要用于人脸检测，“hog”特征主要用于行人检测，“lbp”特征主要用于人脸识别。这里我们只下载lbpcascade\_frontalface.xml文件，事关该实验只实现人脸的检测



### 实现步骤

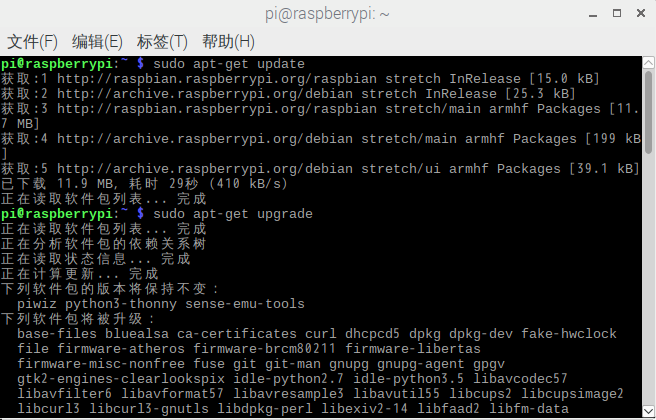
步骤一：安装OpenCV

**sudo apt-get update** #更新所有的软件版本

**sudo apt-get upgrade** #升级所有的软件版本

##以上两个操作是防止安装的OpenCV版本过低

**sudo apt-get install python-opencv**  #安装openCV



步骤二：安装Picamera

**sudo apt-get install python-dev**



**sudo apt-get install python-pip**

**sudo pip install picamera**



步骤三：编写python代码

|  |
| --- |
| **from** picamera**.**array **import** PiRGBArray  **from** picamera **import** PiCamera  **import** cv2  **import** os  **import** pygame  ### Setup #####################################################################  os**.**putenv**(** 'SDL\_FBDEV'**,** '/dev/fb1' **)**  # Setup the camera  camera **=** PiCamera**()**  camera**.**resolution **=** **(** 320**,** 240 **)**  camera**.**framerate **=** 30  rawCapture **=** PiRGBArray**(** camera**,** size**=(** 320**,** 240 **)** **)**  fcounter **=** 0  facefind **=** 0  # Load a cascade file for detecting faces  face\_cascade **=** **cv2.CascadeClassifier( '/home/pi/lbpcascade\_frontalface.xml' )**  t\_start **=** time**.**time**()**  fps **=** 0  ### Main ###########  # Capture frames from the camera  **for** frame **in** camera**.**capture\_continuous**(** rawCapture**,** format**=**"bgr"**,** use\_video\_port**=True** **):**  image **=** frame**.**array  # Run the face detection algorithm every four frames  **if** fcounter **==** 3**:**  fcounter **=** 0  # Look for faces in the image using the loaded cascade file  gray **=** cv2**.**cvtColor**(** image**,** cv2**.**COLOR\_BGR2GRAY **)**  faces **=** face\_cascade**.**detectMultiScale**(** gray **)**  **print** "Found " **+** str**(** len**(** faces **)** **)** **+** " face(s)"  **if** str**(** len**(** faces **)** **)** **!=** 0**:**  facefind **=** 1  facess **=** faces  **else:**  facefind **=** 0  # Draw a rectangle around every face  **for** **(** x**,** y**,** w**,** h **)** **in** faces**:**  cv2**.**rectangle**(** image**,** **(** x**,** y **),** **(** x **+** w**,** y **+** h **),** **(** 200**,** 255**,** 0 **),** 2 **)**  cv2**.**putText**(** image**,** "Face No." **+** str**(** len**(** facess **)** **),** **(** x**,** y **),** cv2**.**FONT\_HERSHEY\_SIMPLEX**,** 0.5**,** **(** 0**,** 0**,** 255 **),** 2 **)**  facess **=** faces  **else:**  **if** facefind **==** 1 **and** str**(** len**(** facess **)** **)** **!=** 0**:**  # Continue to draw the rectangle around every face  **for** **(** x**,** y**,** w**,** h **)** **in** facess**:**  cv2**.**rectangle**(** image**,** **(** x**,** y **),** **(** x **+** w**,** y **+** h **),** **(** 200**,** 255**,** 0 **),** 2 **)**  cv2**.**putText**(** image**,** "Face No." **+** str**(** len**(** facess **)** **),** **(** x**,** y **),** cv2**.**FONT\_HERSHEY\_SIMPLEX**,** 0.5**,** **(** 0**,** 0**,** 255 **),** 2 **)**    fcounter **+=** 1  # Calculate and show the FPS  fps **=** fps **+** 1  sfps **=** fps **/** **(** time**.**time**()** **-** t\_start **)**  cv2**.**putText**(** image**,** "FPS : " **+** str**(** int**(** sfps **)** **),** **(** 10**,** 10 **),** cv2**.**FONT\_HERSHEY\_SIMPLEX**,** 0.5**,** **(** 0**,** 0**,** 255 **),** 2 **)**  cv2**.**imshow**(** "Frame"**,** image **)**  cv2**.**waitKey**(** 1 **)**  # Clear the stream in preparation for the next frame  rawCapture**.**truncate**(** 0 **)** |

# 项目总结

本次的高级实训的内容是跟树莓派相关的实验内容，树莓派是一款功能强大的微型电脑，可以实现众多功能。我们小组选择了“在树莓派上搭建BT下载器”“在树莓派上搭建FTP文件服务器”以及“简易人脸识别”这几个实验。

**Windows远程连接的实现：**

在用Windows远程连接树莓派的实验过程中，我们遇到了第一个问题，在安装好软件激活它后用Windows的电脑进行连接，始终卡在远程连接输入用户名和密码的界面。组员们将教程中的指令多次从头输入运行，没有得到解决；然后考虑我们通过“ifconfig”查询的IP地址是否正确，根据所学的计算机网络知识和树莓派网站上的指导，又排除了该问题；还考虑过是否是防火墙阻隔的问题，尝试按照网上的解答修改了一些端口配置，无果；最后重新设置了新的树莓派密码，才得以连接。在设置新的树莓派密码时，我们发现新密码的设置并不需要验证旧密码，这其实说明树莓派这点在安全性上是不如PC计算机的，但对开发者来说是个便利。

**树莓派做FTP服务器：**

拿树莓派当FTP服务器时，直接在FileZilla输入sftp://IP地址,即使用户名和密码正确，有些树莓派系统也是连接不上的。原因有很多，我们这里遇到是ssh没有安装造成，需要安装完ssh,并重启一遍方可连接成功。此外还有以下几类原因，都是从网上总结摘抄下来的，希望以后遇到类似的问题不用再焦头烂额：

1. 防火墙未开启：在防火墙把filezilla的　Filazilla server interface.exe　和Filezilla server.exe　二个文件都加入防火墙的例外。
2. 站点管理器中，传输设置选择主动，不过这个可能和本来只是连接传输文件的目的不一样。
3. 如果 1，2 无效再选择传输协议为FTP，加密选 只使用普通FTP
4. 如果1,2,3仍无效再把传输设置勾选限制并发连接数。这条的参考：http://www.jb51.net/softjc/326331.html。意思是说：FileZilla 无法连接多线程FTP 服务器。这也作为一个参考方法

4.以上方法都试过了之后，还不行，莫名其妙是不是，那就换一个吧：比如WinSCP，我之前用的贼好用，我觉得使用布局比fileZilla简化的多，使用方便。现在就是想要解决这个问题所以就用这个fileZilla。其他的也行。总之，这只是一个传输文件的工具，安全、易用就行。

**BT下载器的搭建：**

在“在树莓派上搭建BT下载器”的实验当中，代码中的好几条指令都被提示“权限不足”，无法运行指令。我们尝试在指令前加上“sudo”，问题就得到了解决。“sudo”这条指令其实给了我们开发者很大的权限和便利，能让很多指令顺利地运行。树莓派对开发者e确实非常友好，在代码中也只需调用“sudo”指令就可以解决很多问题，而不必修改复杂的配置，新手是容易学习的。

**人脸检测：**

我们在做人脸检测实验时遇到最大的困难是关于**cv2.CascadeClassifier(“xml文件路径”)**的使用，由于该实验并不需要添加任何关于人脸识别的算法，所有的算法实现都在xml格式的分类器上，也就是安装完openCV后默认在

../opencv-2.4.9/data/lbpcascades/lbpcascade\_frontalface.xml的路径上，**只要调用CascadeClassifier（）函数就可以获取摄像头内每一帧能检测到的人脸。**但我重装了好几次OpenCV都没有找到lbpcascade\_frontalface.xml文件路径，后来急中生智索性在GitHub中下载一份 OpenCV的安装包（当然里面包含了haar”、“hog”和“lbp等各种特征的xml文件），并将lbpcascade\_frontalface.xml放在/home/pi下，方便最快效率检索到xml分类器文件，实时检测到摄像头范围内所有的人脸，从而完成了实验的基本功能。

其次是关于人脸检测效果的研究，在准确率、识别完整度上，HaarCascade的xml文件远没有lbpcascade的xml文件的好，这里我也整理了一下haar、hog、lbp三类分类器文件夹的差别，"haar"特征主要用于人脸检测，“hog”特征主要用于行人检测，“lbp”特征主要用于人脸识别，且“lbp”优于“haar”,但“lbp”没有“haar”聚焦那么精确。通俗来讲“lbp”检测一张人脸能划出的面积2如果是25cm^2，那么“haar”检测的面积可以做到是前者的1/2，也就是说后者能更准确聚焦到人脸的中心点，这样带来的优劣之分是：如果一张图片很多人挨得很近，“lbp”可能检测不出所有的人脸，或者说没有“haar”检测的人脸数得多。另一方面，事实上检测的效果是刚好相反的，实时去检测一个人的脸，“lbp”可以很快检测出人脸的方位，并随着人脸位置的移动而移动，而“haar”则需要更长的时间才能检测出人脸，甚至有时候检测不出来。这点不难看出，理论上讲“haar”更侧重检测人脸的“量”，而“lbp”更侧重的是“质”，但是事实上只有在画质清晰下才会成立，当画质差或者画面内目标对象四处晃动的时候，“lbp”检测的效果远比“haar”要好。当然“haar”也不是一无是处，“lbp”只能识别人脸，“haar”却还能识别人眼、笑容等属性，这不是前者能相比的

总的来说，高级实训的过程是非常有挑战性的，但在实验的难度上，树莓派网站设置了阶梯性，让我们的自主研发探究过程循序渐进，逐渐地累积成就感，给人继续探索的乐趣。实验指导简明易懂，可操作性非常强，步骤清晰，对于我们这些新手来说十分友好。有时我们会遇到较为棘手的问题，可能需要从树莓派网站以外的地方获得答案,但这样的地方不多，也说明了树莓派功能的强大和开发者友好度。树莓派还有众多强大而实用的功能，希望能继续探索和研究。

（包括遇到的问题，解决过程及结论。未解决的问题，给出你们可能的建议。项目进一步完善的建议）

参考文献