Student ID: 406410035
Name: 秦紫頤
E-mail: chinjoyce30@gmail.com

Lab Title: Implementing Pedestrian Detection on Raspberry Pi

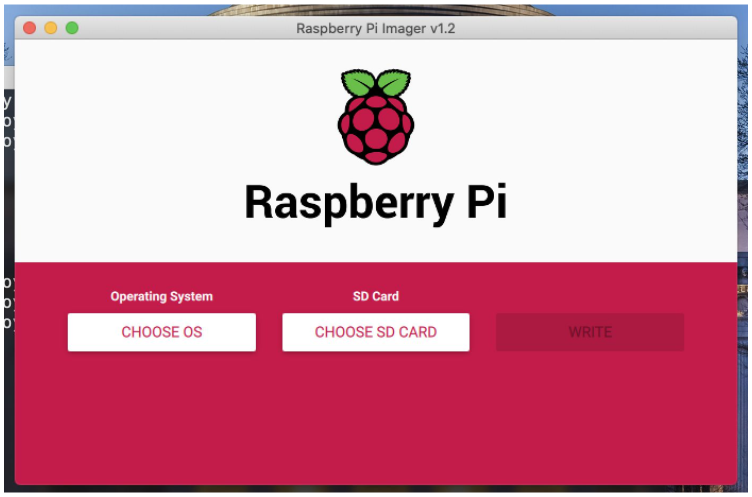
Lab Purpose:

First, let students learn som computer vision techniques. Second, let students understand some image processing techniques. And lastly, learn how to implement OpenCV on Raspberry Pi.

Lab Procedure:

Part 1: Install Tools

1. Download the Raspbian system from Raspberry Pi official site
2. Format the SD card
   a. Delete all the existence partitions
   b. Download Raspberry Pi Imager from Raspberry Pi official site (for macOS system)
   c. Format the SD card as FAT32 using Raspberry Pi Imager

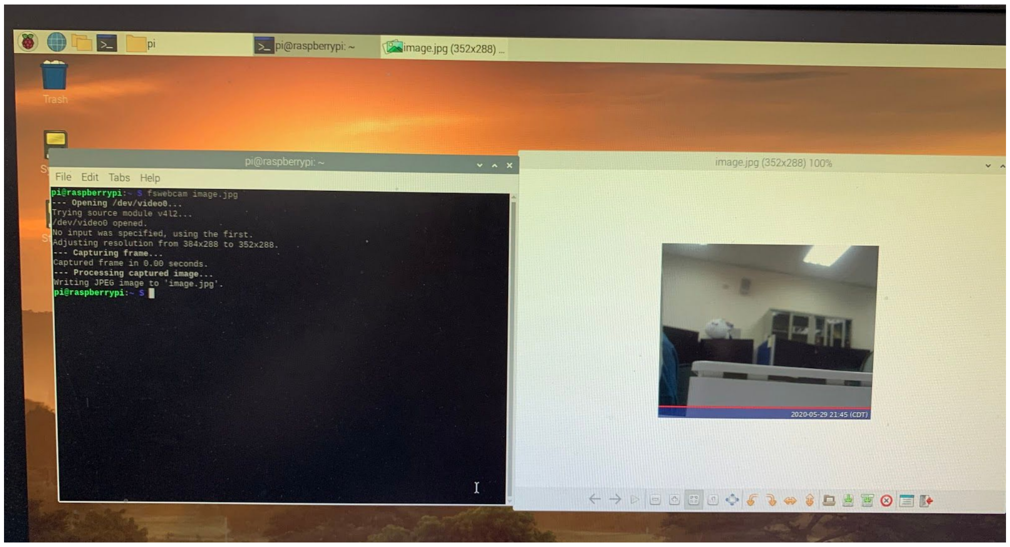


3. Decompress Raspbian zip file and put all the objects into the SD Card
4. Power on Raspberry Pi and install the system

5. After the installation update the system
    a. Update the firmware: `sudo rpi-update`
    b. Update the system packages: `sudo apt-update`
    c. Upgrade the system packages: `sudo apt-upgrade`
6. Reboot Raspberry Pi
7. Install some packages needed for program compilation: `sudo apt-get install build-essential git cmake pkg-config`
8. Install some image I/O packages: `sudo apt-get install libjpeg8-dev libtiff5-dev libjasper-dev libpng12-dev`
9. Install packages need for video I/O: `sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev libxvidcore-dev libx264-dev`
10. Install GTK development library: `sudo apt-get install libgtk2.0-dev`
11. Install packages for matrix computation optimization tools: `sudo apt-get install libatlas-base-dev gfortran`
12. Install OpenCV
    a. Cache OpenCV to find the supporting version currently: `sudo apt-cache search opencv`
    b. Install OpenCV: `sudo apt-get install libopencv-dev, libopencv-highgui3.2`

Part 2: USB Camera

1. Install UV4L
    a. Download UV4L: `curl http://www.linux-projects.org/listing/uv4l_repo/lrk ey.asc | sudo apt-key add -`
    b. Modified `/etc/apt/sources.list` file: `sudo nano /etc/apt/sources.list -> deb http://www.linux-projects.org/listing/uv4l_repo/raspbian/ wheezy main`
    c. Update the system: `sudo apt-get update`
    d. Install uv4l library: `sudo apt-get install uv4l uv4l-raspicam`
    e. If you want to load uv4l when booting up the system, you need to install an extra package: `sudo apt-get install uv4l-raspicam-extras`
    f. Start uv4l service: `sudo service uv4l_raspicam restart`
    g. Update system firmware because we just add a new driver: `sudo rpi-update`
    h. Install v4l2-utils: `sudo apt-get install v4l-utils`
    i. Connect USB camera and reboot the system
    j. Check if the camera is connected to the system: `v4l2-ctl --list-devices`
2. Install fswebcam
    a. Install fswebcam: `sudo apt-get install fswebcam`
    b. Take a picture for testing: `fswebcam image.jpg`

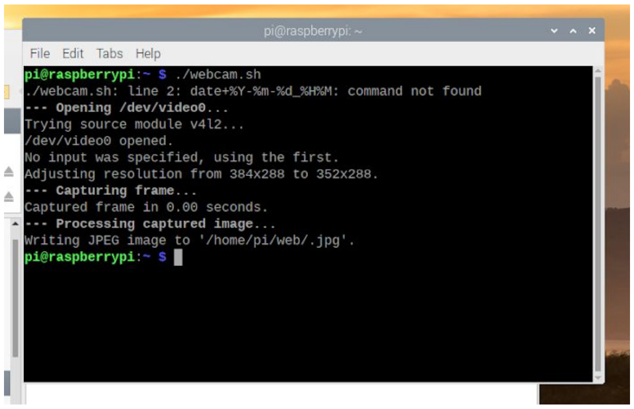


```
pi@raspberrypi:~ $ fswebcam image.jpg
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
Adjusting resolution from 384x288 to 352x288.
--- Capturing frame...
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to 'image.jpg'.
pi@raspberrypi:~ $
```

c. Write a script to take pictures -> **webcam.sh**

d. give the script executable permission: `chmod +x webcam.sh`

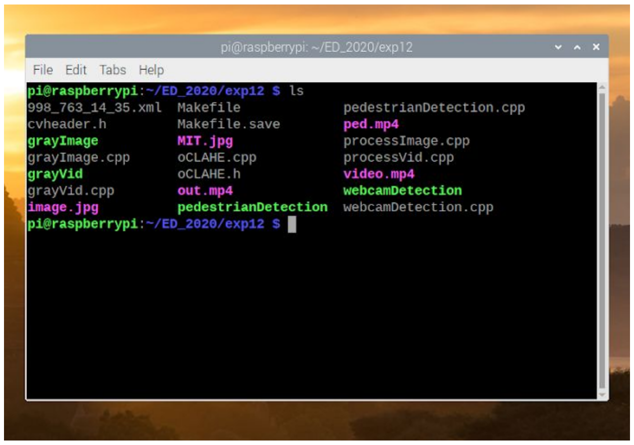
e. Execute the script to take picture: `./webcam.sh` (I got a problem here. The name of the picture should be based on the date and the time the image was taken but the execution result is NULL no image. But in step b I could take pictures with a specific name, so I think the problem here is in the sample script it's given)



```
pi@raspberrypi:~ $ ./webcam.sh
./webcam.sh: line 2: date+%Y-%m-%d_%H%M: command not found
--- Opening /dev/video0...
Trying source module v4l2...
/dev/video0 opened.
No input was specified, using the first.
Adjusting resolution from 384x288 to 352x288.
--- Capturing frame...
Captured frame in 0.00 seconds.
--- Processing captured image...
Writing JPEG image to '/home/pi/web/.jpg'.
pi@raspberrypi:~ $
```

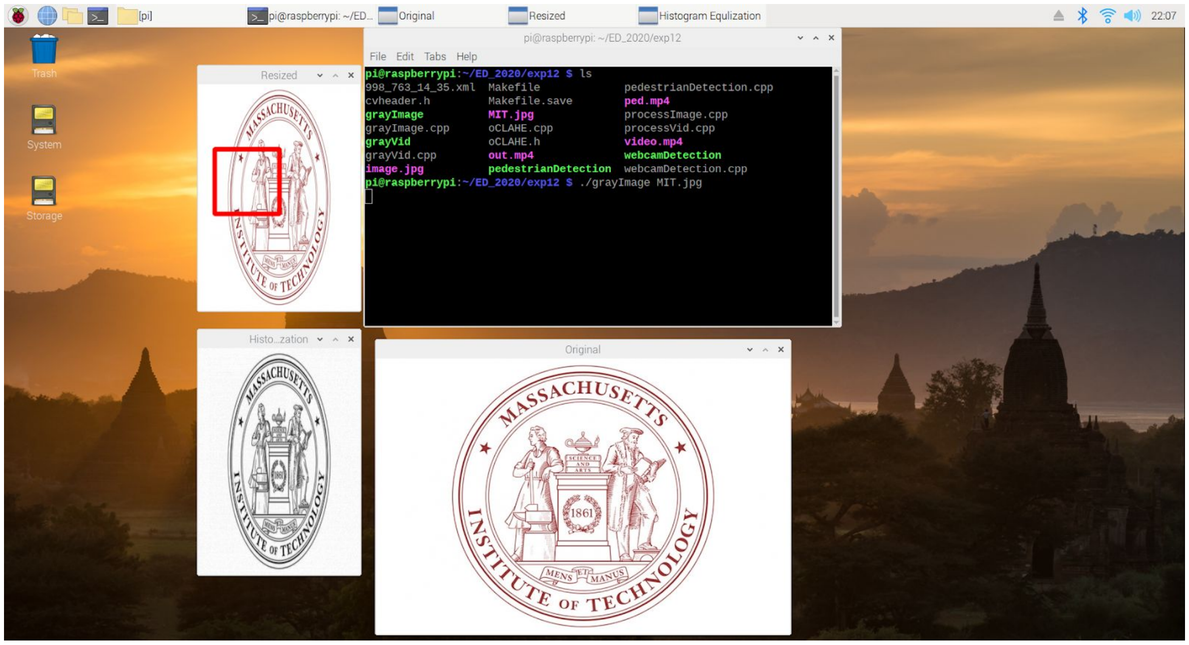
Part 3: OpenCV for basic image and video processing

1. Create two header files include all the C++ header we need in this part
    a. `myOpenCV.h`: OpenCV C++ header files
    b. `OCLAHE.h, OCLAHE.cpp`: This is for the night-image process (gray-scale)



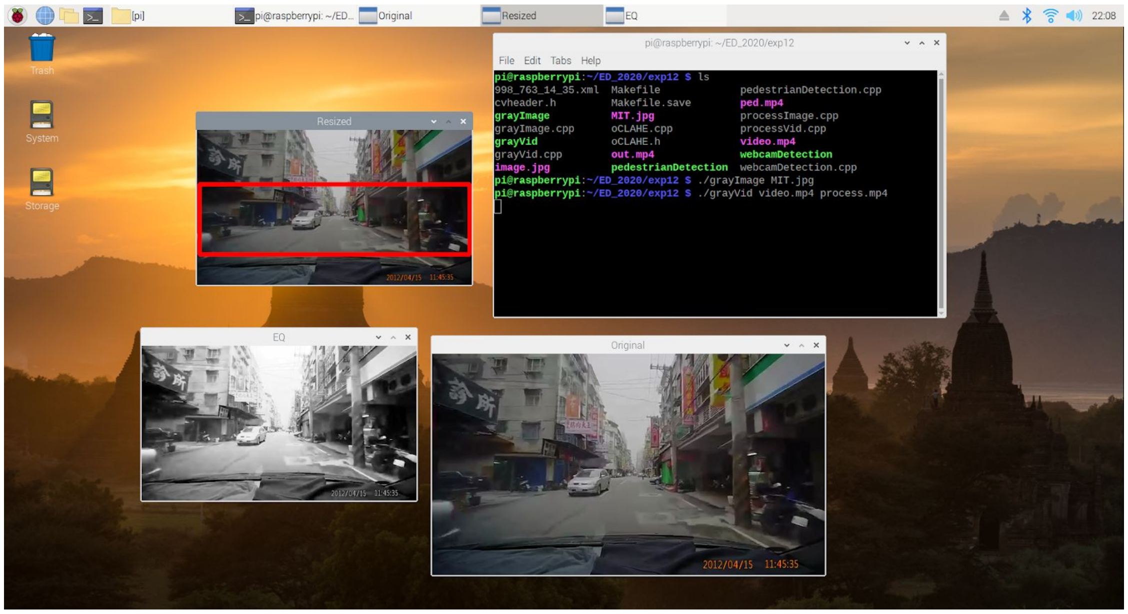
```
pi@raspberrypi:~/ED_2020/exp12 $ ls
998_763_14_35.xml  Makefile            pedestrianDetection.cpp
cvheader.h         Makefile.save       ped.mp4
grayImage          MIT.jpg             processImage.cpp
grayImage.cpp      oCLAHE.cpp          processVid.cpp
grayVid            oCLAHE.h            video.mp4
grayVid.cpp        out.mp4             webcamDetection
image.jpg          pedestrianDetection webcamDetection.cpp
pi@raspberrypi:~/ED_2020/exp12 $
```

2. Image processing: write a program that can open the image, receive the image, ROI pooling, and turn the image to grayscale



```
pi@raspberrypi:~/ED_2020/exp12 $ ls
998_763_14_35.xml  Makefile                pedestrianDetection.cpp
cvheader.h         Makefile.save           ped.mp4
grayImage          MIT.jpg                 processImage.cpp
grayImage.cpp      oCLAHE.cpp              processVid.cpp
grayVid            oCLAHE.h                video.mp4
grayVid.cpp        out.mp4                 webcamDetection
image.jpg          pedestrianDetection     webcamDetection.cpp
pi@raspberrypi:~/ED_2020/exp12 $ ./grayImage MIT.jpg
```

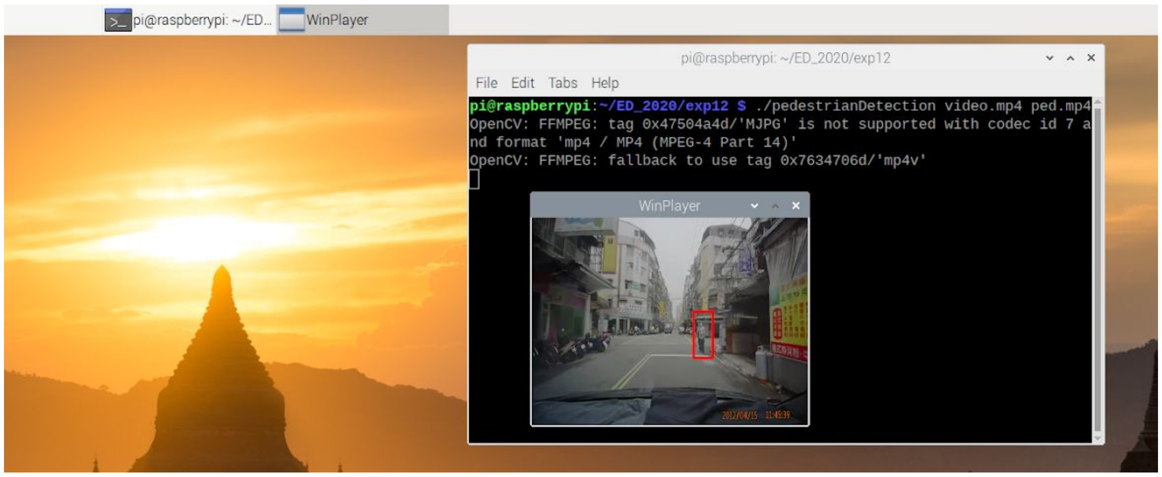
3. Video processing: write a program that can get the video, parse the video to image frame by frame, resize the frames, set ROI pooling, and turn the frame to grayscale



```
pi@raspberrypi:~/ED_2020/exp12 $ ls
998_763_14_35.xml  Makefile                pedestrianDetection.cpp
cvheader.h         Makefile.save           ped.mp4
grayImage          MIT.jpg                 processImage.cpp
grayImage.cpp      oCLAHE.cpp              processVid.cpp
grayVid            oCLAHE.h                video.mp4
grayVid.cpp        out.mp4                 webcamDetection
image.jpg          pedestrianDetection     webcamDetection.cpp
pi@raspberrypi:~/ED_2020/exp12 $ ./grayImage MIT.jpg
pi@raspberrypi:~/ED_2020/exp12 $ ./grayVid video.mp4 process.mp4
```

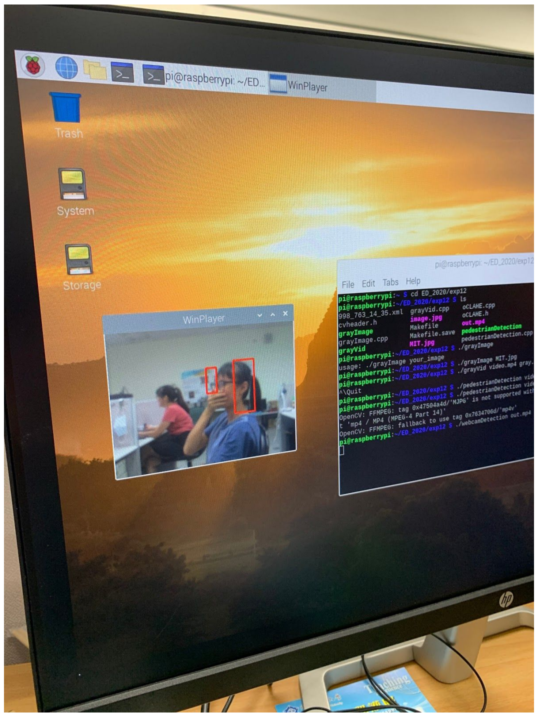
Part4: Pedestrian Detection system

1. Video Detection
    a. Sampling document: 998_763_14_35.xml
    b. Use `draw()` to draw bounding box around detected human
    c. Use the sampling document to recognize the human and place it in the video
    d. Use cvCreate Video Writer to record the analysis (detection) video



```
pi@raspberrypi:~/ED_2020/exp12 $ ./pedestrianDetection video.mp4 ped.mp4
OpenCV: FFMPEG: tag 0x47504a4d/'MJPG' is not supported with codec id 7 a
nd format 'mp4 / MP4 (MPEG-4 Part 14)'
OpenCV: FFMPEG: fallback to use tag 0x7634706d/'mp4v'
```

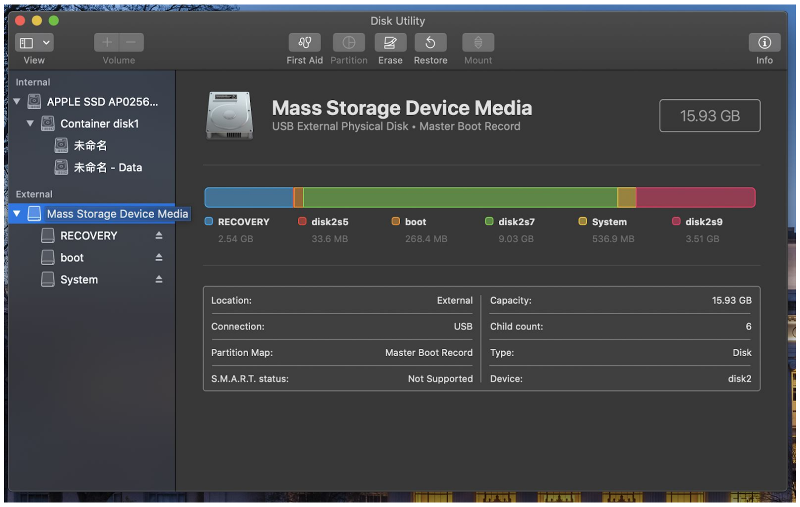
2. Real-time face detection
    a. Using the webcam to capture the video (parse frame by frame to do analysis)
    b. Use sampling document 998_763_14_35.xml
    c. Use `draw()` to draw bounding box around detected human



```
pi@raspberrypi:~ $ cd ED_2020/exp12
pi@raspberrypi:~/ED_2020/exp12 $ ls
998_763_14_35.xml  grayVid.cpp    oCLAHE.cpp
cvheader.h         image.jpg      oCLAHE.h
grayImage          Makefile       out.mp4
grayImage.cpp      Makefile.save  pedestrianDetection
grayVid            MIT.jpg        pedestrianDetection.cpp
pi@raspberrypi:~/ED_2020/exp12 $ ./grayImage
usage: ./grayImage your_image
pi@raspberrypi:~/ED_2020/exp12 $ ./grayImage MIT.jpg
pi@raspberrypi:~/ED_2020/exp12 $ ./grayVid video.mp4 gray.
pi@raspberrypi:~/ED_2020/exp12 $ ./pedestrianDetection vide
^\Quit
pi@raspberrypi:~/ED_2020/exp12 $ ./pedestrianDetection vide
OpenCV: FFMPEG: tag 0x47504a4d/'MJPG' is not supported with
t 'mp4 / MP4 (MPEG-4 Part 14)'
OpenCV: FFMPEG: fallback to use tag 0x7634706d/'mp4v'
pi@raspberrypi:~/ED_2020/exp12 $ ./webcamDetection out.mp4
pi@raspberrypi:~/ED_2020/exp12 $
```

Problems and Discussions

I formatted the SD Card by deleting the partition using `fdisk` command. And after the deletion, my computer detected the SD Card as a damaged device. I first used MacOS disk utility to repair the SD Card and it works. MacOS disk utility is quite powerful. And then I tried to install Raspberry Pi Imager in Ubuntu to format the SD Card. But I didn't know why the Imager installation failed. So I installed Raspberry Pi Imager in my host system (macOS) to format the SD Card to FAT32



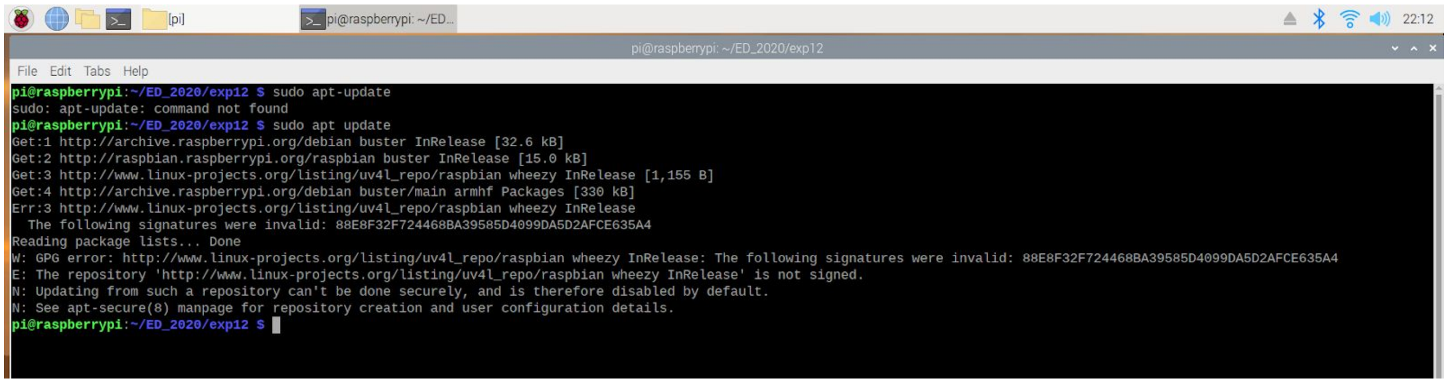
Mass Storage Device Media
USB External Physical Disk • Master Boot Record
15.93 GB

RECOVERY	disk2s5	boot	disk2s7	System	disk2s9
2.54 GB	33.6 MB	268.4 MB	9.03 GB	536.9 MB	3.51 GB

Location:	External	Capacity:	15.93 GB
Connection:	USB	Child count:	6
Partition Map:	Master Boot Record	Type:	Disk
S.M.A.R.T. status:	Not Supported	Device:	disk2

While installing the packages by `sudo apt-get install` command there was some time that the installation failed with no reason. The solution is I don't install many packages via one command, instead, I only install one package at a time, and if still failed I install it again because sometimes the reason to failure is the network problem. This guides me to install all the packages successfully.

When installing UV4L I got many errors. First after `sudo apt update` I get this error **"The following signatures were invalid"**. I searched online and it said maybe the key is expired so I tried to refresh the key but not working. Then I found someone on StackOverflow has the same problem installing the same package as I am so I tried the recommended solution in StackOverflow [Cannot install uv4l raspicam](). I ignore all the errors and warnings this time and finally solve the problem. I still don't know why these errors happen but the solution did solve my problem miraculously.



```
pi@raspberrypi:~/ED_2020/exp12 $ sudo apt-update
sudo: apt-update: command not found
pi@raspberrypi:~/ED_2020/exp12 $ sudo apt update
Get:1 http://archive.raspberrypi.org/debian buster InRelease [32.6 kB]
Get:2 http://raspbian.raspberrypi.org/raspbian buster InRelease [15.0 kB]
Get:3 http://www.linux-projects.org/listing/uv4l_repo/raspbian wheezy InRelease [1,155 B]
Get:4 http://archive.raspberrypi.org/debian buster/main armhf Packages [330 kB]
Err:3 http://www.linux-projects.org/listing/uv4l_repo/raspbian wheezy InRelease
  The following signatures were invalid: 88E8F32F724468BA39585D4099DA5D2AFCE635A4
Reading package lists... Done
W: GPG error: http://www.linux-projects.org/listing/uv4l_repo/raspbian wheezy InRelease: The following signatures were invalid: 88E8F32F724468BA39585D4099DA5D2AFCE635A4
E: The repository 'http://www.linux-projects.org/listing/uv4l_repo/raspbian wheezy InRelease' is not signed.
N: Updating from such a repository can't be done securely, and is therefore disabled by default.
N: See apt-secure(8) manpage for repository creation and user configuration details.
pi@raspberrypi:~/ED_2020/exp12 $ 
```

When doing real-time face detection that part at first my webcam didn't work. The code stopped at `cvCaptureFromCAM(0)` with error said no device detected. I thought this shouldn't happen because I did install uv4l and fswebcam properly and tested the webcam's functionality. The solution for this was I changed my Raspberry Pi and the program executed successfully.

The last problem I encountered happens throughout the whole experiment. Raspberry Pi like any other host computer needs power. Raspberry Pi with Raspbian system needed more power than previous experiments which only has a terminal in it. At first, I use my Macbook Pro as its power source, but Mac system has a mechanism that is when it detected large power-consuming by the device plugged into it, it will disable the USB port temporarily. Then my Raspberry Pi power off abruptly at the moment my Mac disabled the USB port.

Then I thought of another idea is to charge my Raspberry Pi like smartphones plugging into a wall socket connected with a power plug. This method didn't actually solve my problem the screen switch on and off too frequently, I think this is because the wall socket is alternating current. Finally, I find a super long charging cable and connected to the mainframe from the person sitting beside me in the MVL lab, this did provide stable power to my Raspberry Pi to finish this experiment.