

Raspberry Pi and MATLAB Drone Detector

Installation and Troubleshooting Instructions

Mohamed Abdel Bagi Ahmed 1037372

Ahmed Ibrahim Elbenawi 1059431

Bader Eddin Ahmed Chayeb 1058952

Mohamed Fathy Salem 1061169

SUPERVISED BY: DR. MOHAMMED GHAZAL,
DR. MONTASIR QASYMEH



Submitted: September 12, 2020

Contents

1	MATLAB Instructions	4
1.1	Setting Up MATLAB	4
1.2	MATLAB Codes Instructions and Troubleshooting Tips	15
1.2.1	CNN Detector	15
1.2.2	RCNN Detector	15
1.2.3	YOLOv2 Detector	16
2	Raspberry Pi Installation Instructions	18
2.1	Before Booting The Raspberry Pi	18
2.2	Booting the Raspberry Pi	21
2.2.1	Requirements	21
2.2.2	Instructions	21
2.2.3	Troubleshooting	22
2.3	Setting Up the Raspberry Pi	23
2.3.1	Enabling camera and SSH	23
2.3.2	Expand SD card	23
2.3.3	Install the OpenCV Dependencies	24
2.3.4	Installing Numpy and creating the virtual environment	25
2.3.5	Compiling OpenCV 4 from source	26
2.3.6	Test OpenCV Installation	32
2.4	Create the Custom Haarcascade	32
2.4.1	Requirements	32
2.4.2	Instructions	33
2.4.3	Troubleshooting	35
2.5	Running the Python Codes	36
2.5.1	Requirements	36
2.5.2	Instructions	36
2.5.3	Troubleshooting	37

List of Figures

1.1	MATLAB Download Link.	4
1.2	Matlab Installation Process 1.	5
1.3	Matlab Installation Process 2.	6
1.4	Matlab Installation Process 3.	7
1.5	Matlab Installation Process 4.	8
1.6	Matlab Installation Process 5.	9
1.7	Matlab Installation Process 6.	10
1.8	Matlab Installation Process 7.	11
1.9	Matlab Installation Process 9.	12
1.10	Matlab Installation Process 9.	13
1.11	Matlab Installation Process 10.	14
2.1	Downloading Raspberry Pi Imager.	19
2.2	Using the Raspberry Pi Imager.	20
2.3	Cmake Configuration Output.	29
2.4	Cmake Configuration Output of Non-free Algorithms.	30
2.5	Testing The Installation of OpenCV.	32

Chapter 1

MATLAB Instructions

1.1 Setting Up MATLAB

1. Firstly, to use MATLAB you need to download and Install MATLAB from <https://www.mathworks.com/products/matlab.html> as in figure 1.1. You must have a MathWorks account and a licence key which can be provided by your institution, bought by the use, or having the file that contains the licence before installing the software.

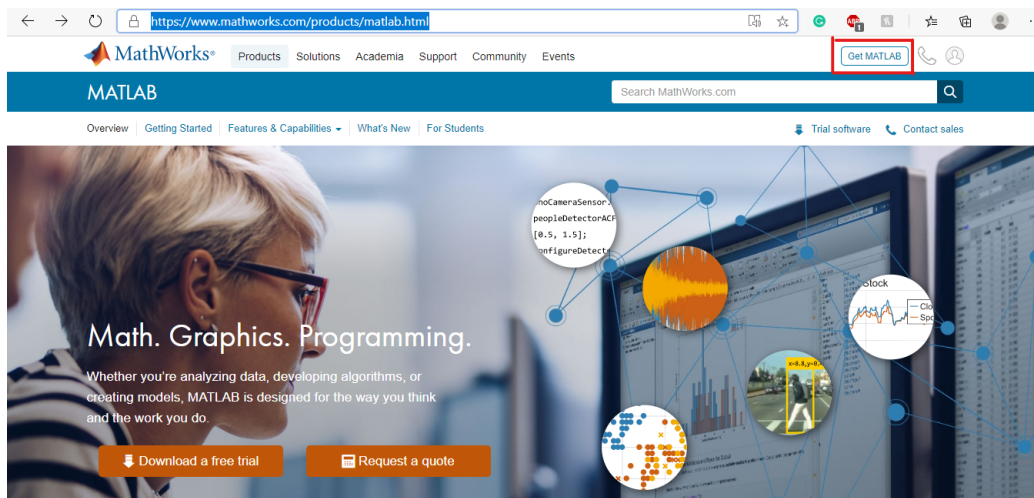


Figure 1.1: MATLAB Download Link.

2. Then you choose the operating system of your computer and the install setup file will start downloading. 1.2 **NOTE: For quicker booting time and faster experience, please install MATLAB on your SSD especially if you will use many products from MathWorks**

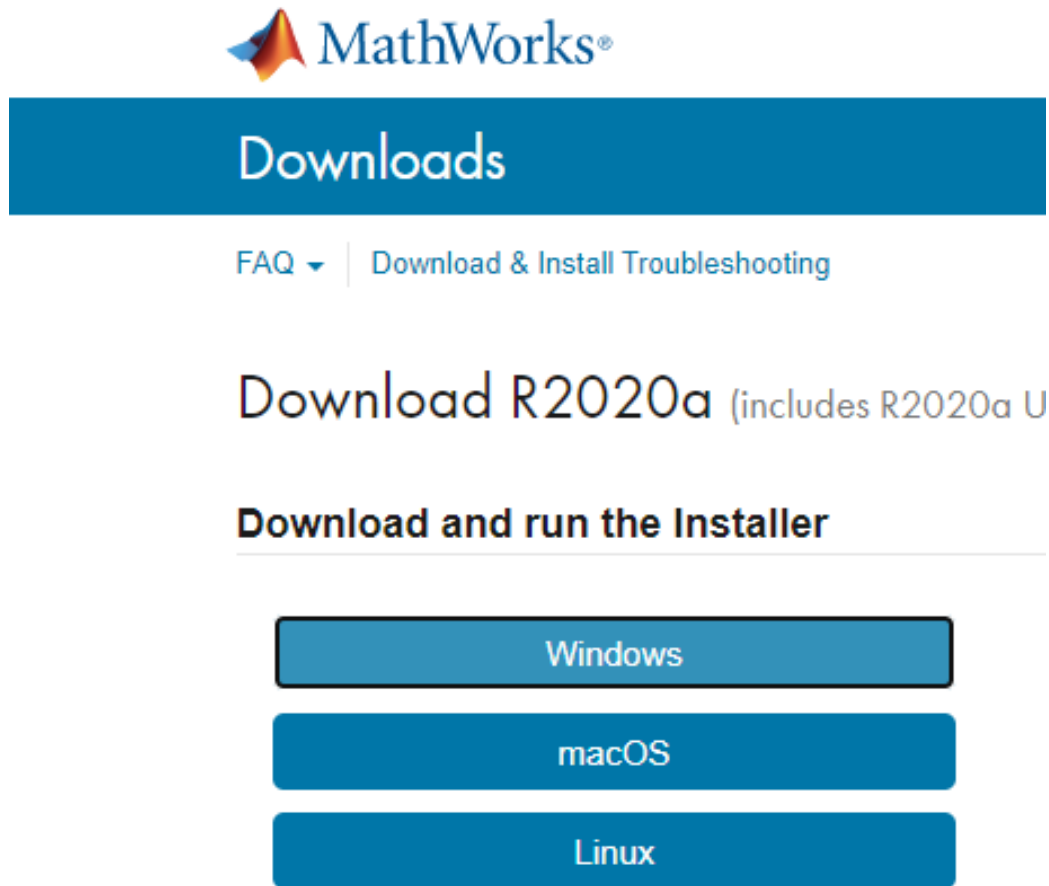


Figure 1.2: Matlab Installation Process 1.

3. When MATLAB is downloaded and the setup file is running, Enter the email address of your as in figure 1.3.

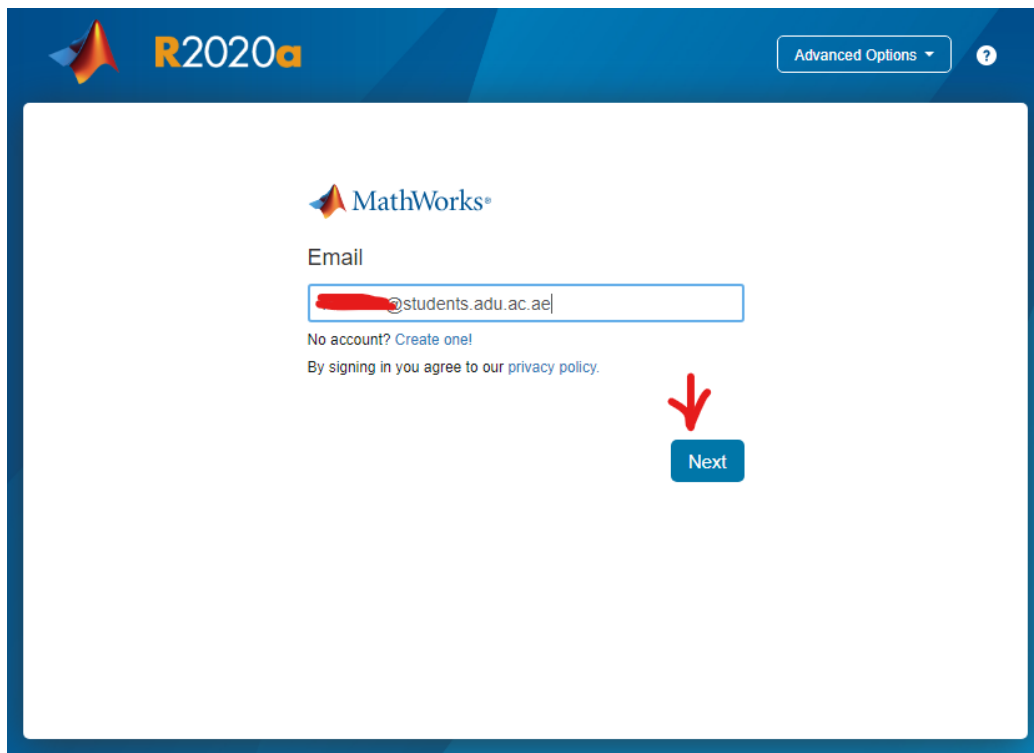


Figure 1.3: Matlab Installation Process 2.

4. Then enter the account's password as displayed in figure 1.4 and choose **Sign In**.

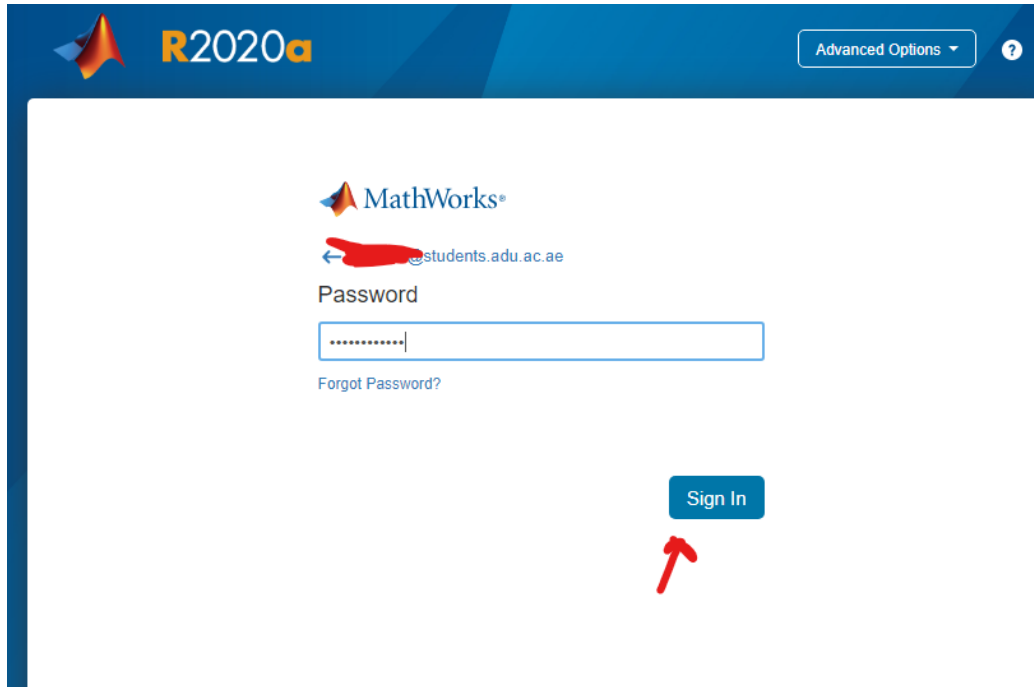


Figure 1.4: Matlab Installation Process 3.

5. Next, accept the license agreement and choose **Next**. 1.5

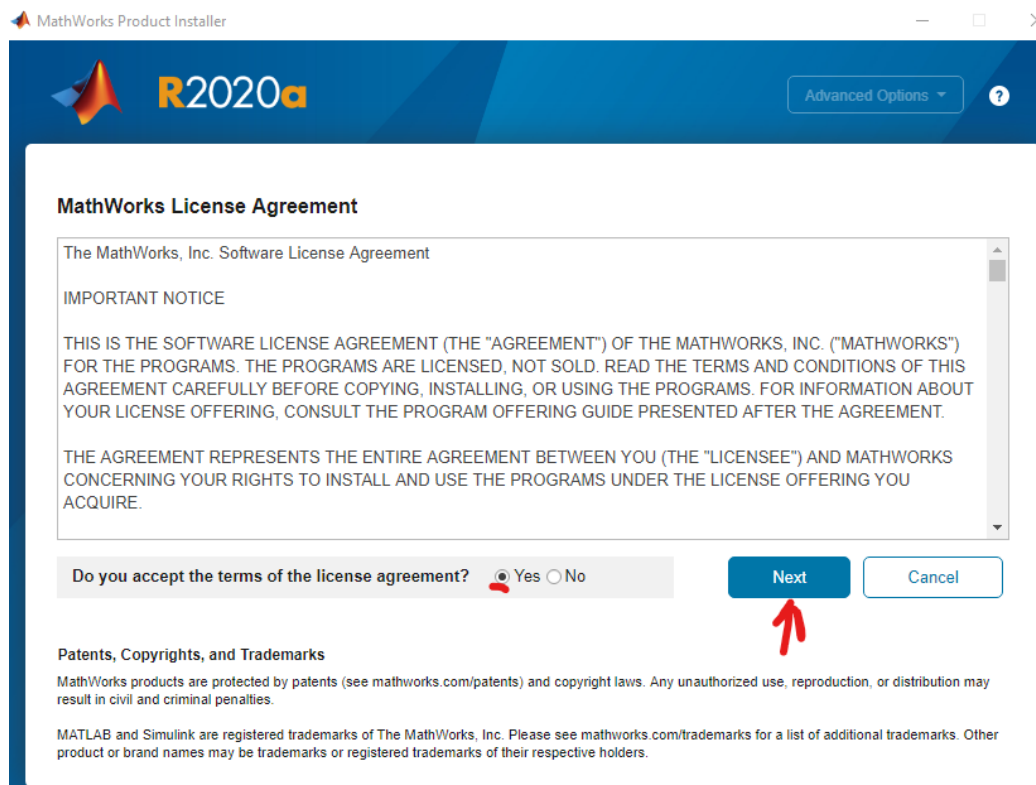


Figure 1.5: Matlab Installation Process 4.

6. After agreeing and logging in, choose the correct license key that is valid and working. if your license key is not showing, manually enter the key and press **Next**. 1.6

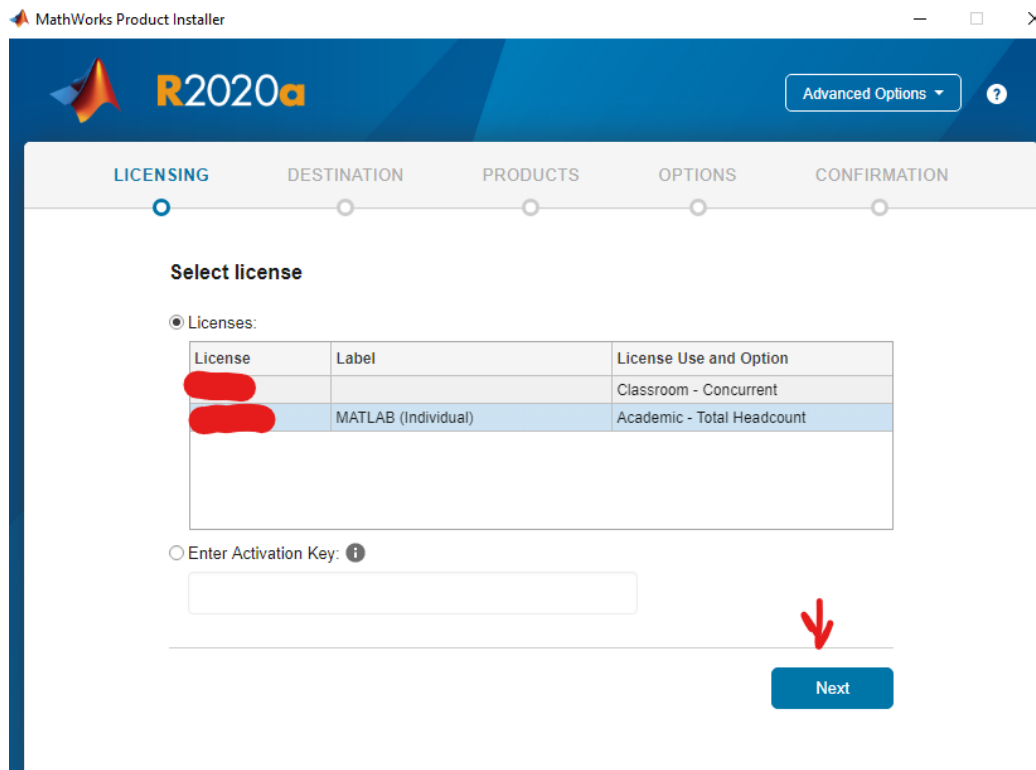


Figure 1.6: Matlab Installation Process 5.

7. Confirm the account's information and select **Next** as in figure 1.7

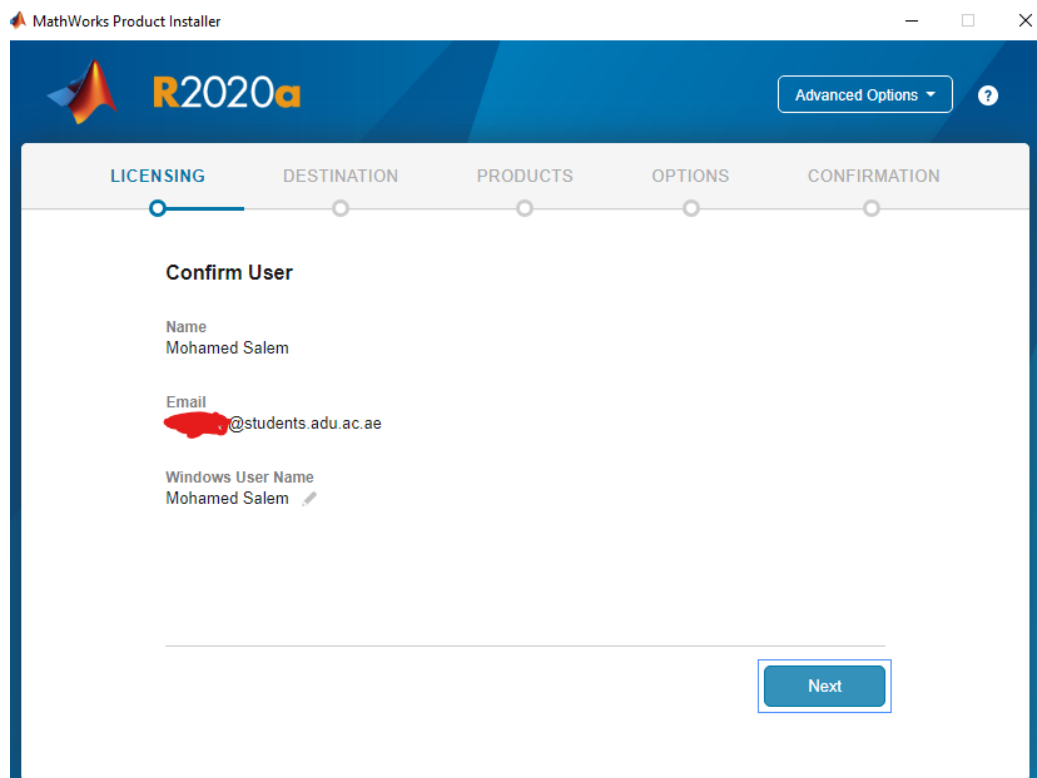


Figure 1.7: Matlab Installation Process 6.

8. Select the Destination folder where all the files and codes related to MATLAB will be stored in like in figure 1.8.

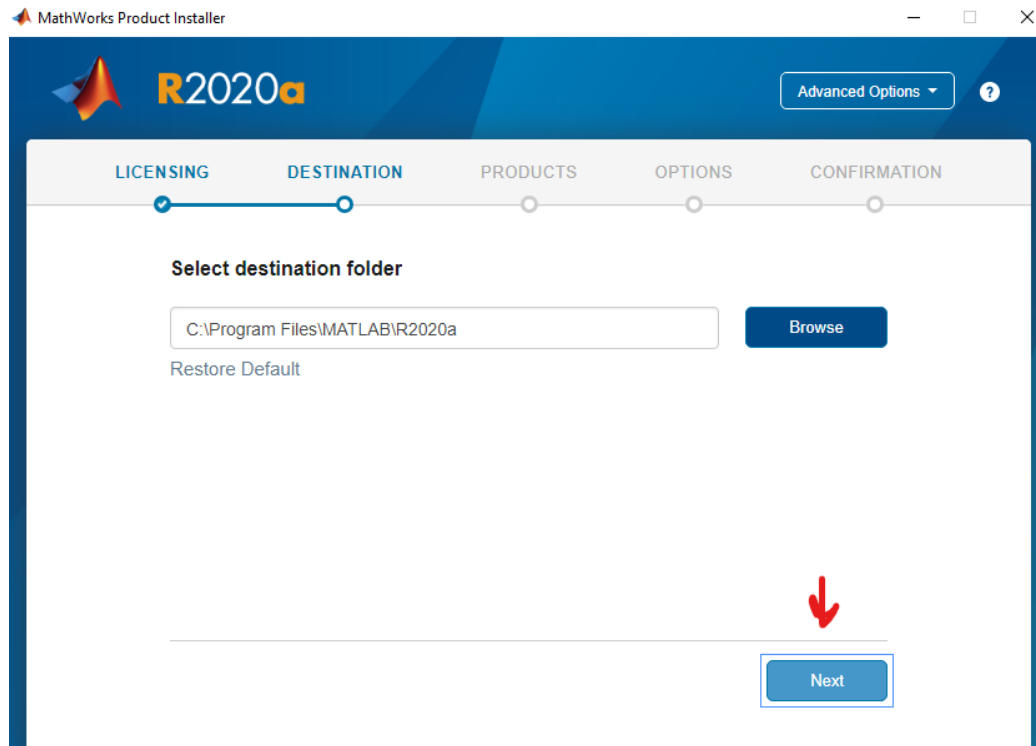


Figure 1.8: Matlab Installation Process 7.

9. Next step is to choose the products or toolboxes that you will need when using MATLAB. The default toolboxes will be selected from the beginning. **NOTE : To work on the project you must install *Deep Learning Toolbox*, *Image Processing Toolbox*, and *Computer Vision Toolbox* to be able to use the image and video labelers in addition to alexnet and other features**

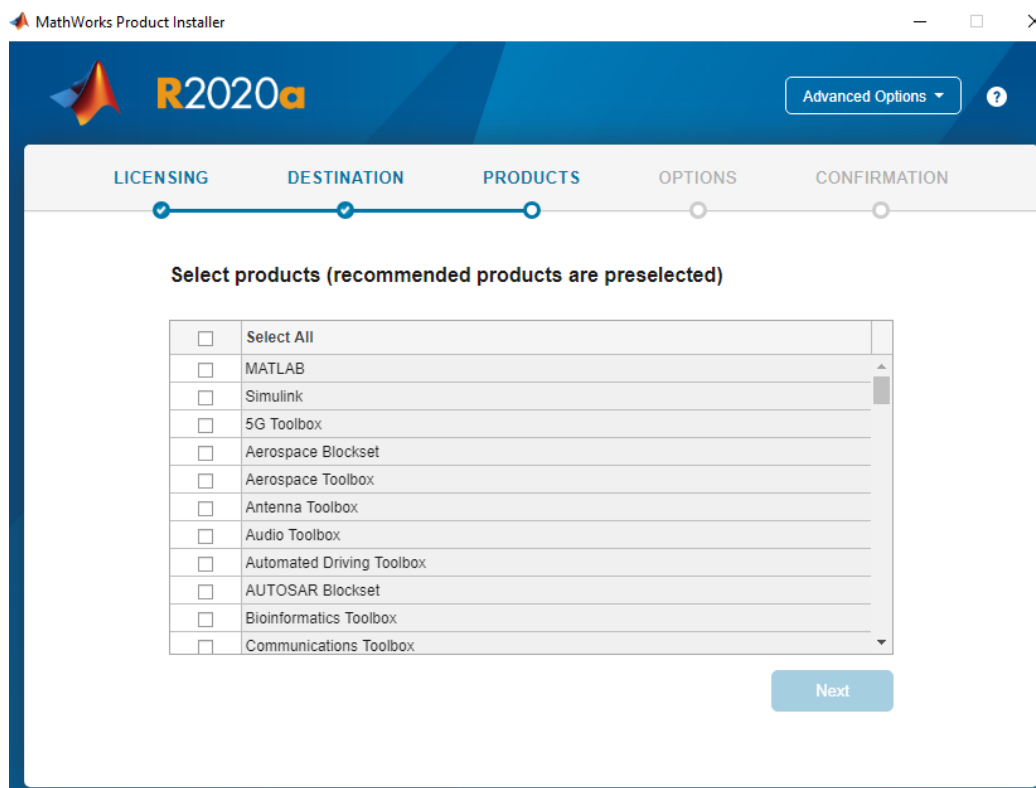


Figure 1.9: Matlab Installation Process 9.

10. After choosing the products of MATLAB, modify the installation option to your liking and choose **Next** as done in figure 1.10 **NOTE: If you want to find the application faster, add MATLAB to your desktop**

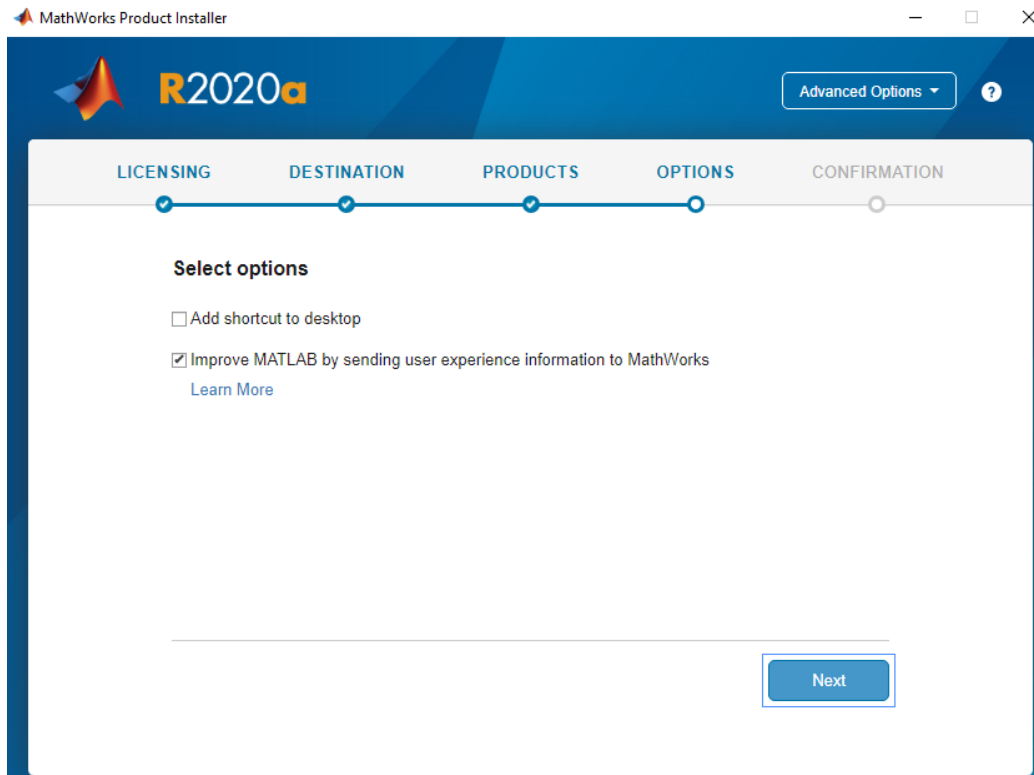


Figure 1.10: Matlab Installation Process 9.

11. Finally, begin the installing process and wait till the installation is done. If more products and toolboxes are installed the longer the booting time and installation time as in 1.11.

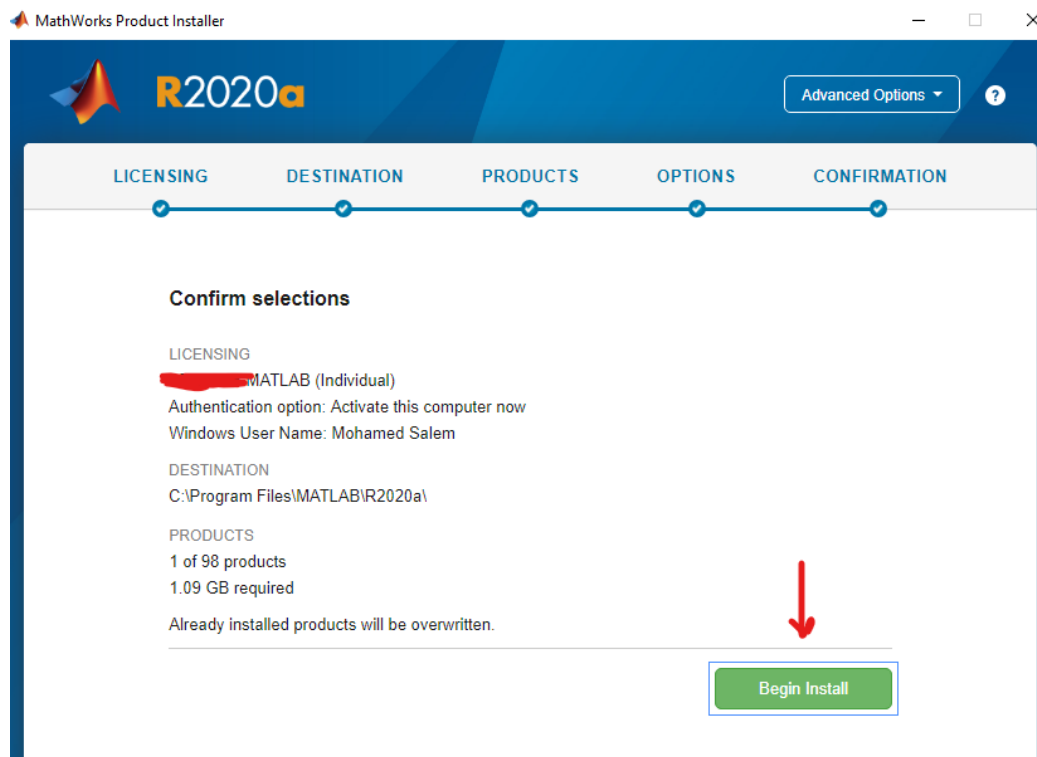


Figure 1.11: Matlab Installation Process 10.

1.2 MATLAB Codes Instructions and Troubleshooting Tips

1.2.1 CNN Detector

Requirements

- MATLAB code **cnnDetector.m**.
- Image Datastore file **ds** (you can use your own data as well).
- MATLAB that supports Deep Learning.

Instructions

1. Open **cnnDetector.m** and make sure that the full path of imagesDatastore is correct or the Image DataStore **ds** is in the same directory of the MATLAB code.

Troubleshooting

1. If the error is while reading the image, make sure that the file names are correct.
2. If the error is in alexnet, make sure that the Deep Learning toolbox is installed. If not, go to **HOME** panel, **Add-Ons**, then search for **Deep Learning Toolbox** and install.
3. If the error is an incorrect input size, make sure that the images are resize to $[227\ 227\ 3]$ or $[227\ 227]$.
4. if the error is in drawing the figure or the bar graph, make sure that there are no typos.

1.2.2 RCNN Detector

Requirements

- MATLAB Code **rcnnDetector.m** for detection from image & **rcnnDetectorVid.m** for deteting from video.

- Ground Truth Table.
- MATLAB that supports Deep Learning, Image Processing, and Computer Vision.
- test image or test video.

Instructions

1. Create a ground truth table using **Image Labeler** for images, or **Video Labeler** for videos. Or use our own ground truth table **newGtruth.mat** for labeled images, and **gtruth.mat** for labeled videos.
2. Use our test image or video by adding the video/image to the same directory of the code, or use the full path of the video/image. You can also use your own image by changing the variable *img* in **rcnnDetector.m** or *vidReader* in **rcnnDetectorVid.m**
3. Finally run the code file. **NOTE:If you will use the video detector in rcnnDetectorVid.m, you MUST run the rcnnDetector.m file**

Troubleshooting

1. Firstly, make sure that the ground truth table, test image, or test video are in the directory or the path of the files is correct.
2. If the error is from the **trainRCNNObjectDetector**, make sure that the arguments and parameters are correct.
3. If the system is out of memory, it is because MATLAB requires a lot of the performance of the computer. So close any unnecessary programs, and I prefer using any browser except Google Chrome since it takes so much RAM especially if a lot of tabs are opened.

1.2.3 YOLOv2 Detector

Requirements

- Ground Truth Tables **gtruth.mat** & **gTruthTable.mat**, or your own ground truth tables.

- MATLAB Code **designYOLO.m**
- MATLAB that supports Deep Learning, Image Processing, and Computer Vision.
- MATLAB Code **AnchorBoxes.m**

Instructions

1. Load the Ground Truth Tables **gtruth.mat** & **gTruthTable.mat**, or your own ground truth tables if you have.
2. Run the **designYOLO.m** code.

Troubleshooting

1. Firstly, make sure that the ground truth tables are in the directory or the path of the tables is correct.
2. If the error is from the **trainYOLOv2ObjectDetector**, make sure that the arguments and parameters are correct.
3. If the system is out of memory, it is because MATLAB requires a lot of the performance of the computer. So close any unnecessary programs, and I prefer using any browser except Google Chrome since it takes so much RAM especially if a lot of tabs are opened.

Chapter 2

Raspberry Pi Installation Instructions

2.1 Before Booting The Raspberry Pi

1. Firstly, you get a pre installed NOOBS micro SD Card from Raspberry Pi, or install the operating system on your own micro SD Card which is recommended and better. a 32 GB SD Card is recommended for operating the Raspberry Pi with OpenCV.
2. Go to <https://www.raspberrypi.org/downloads/> and download the **Raspberry Pi Imager** for your computer's operating system as in figure 2.1

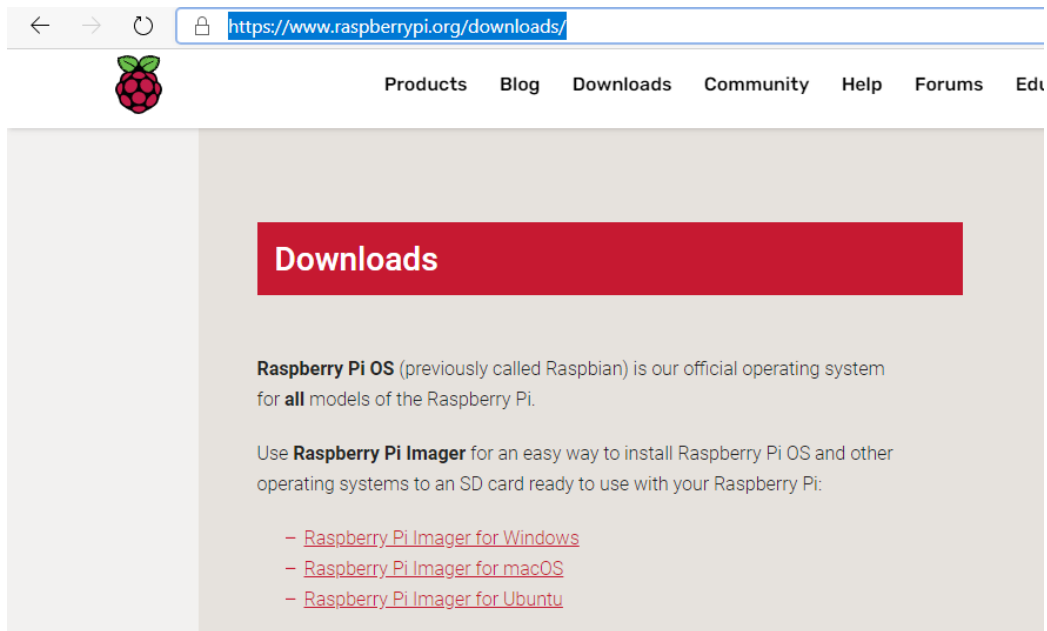


Figure 2.1: Downloading Raspberry Pi Imager.

3. Install and run the the Imager. You will choose the operating system you want to run your Raspberry Pi, choose the SD Card, then choose **write** to write the operating system on the micro SD Card as shown in figure 2.2 **NOTE: Please use a non important SD card or create a backup on your computer since the Imager will overwrite all the old files.**

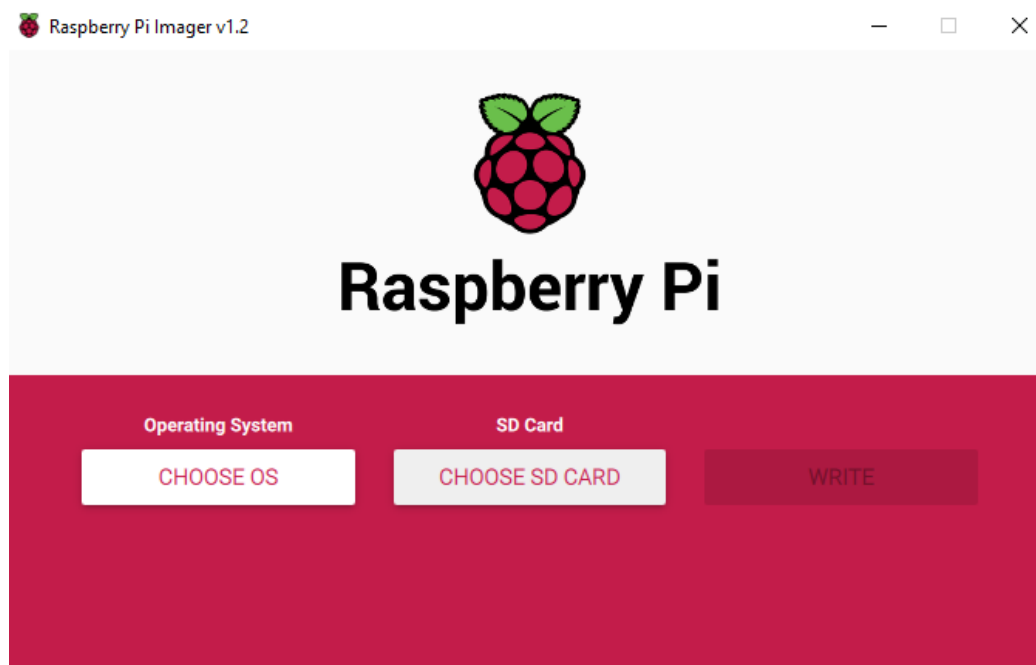


Figure 2.2: Using the Raspberry Pi Imager.

4. Now your micro SD card will be unmounted and ready to be used in the Raspberry Pi.

2.2 Booting the Raspberry Pi

2.2.1 Requirements

To boot your Raspberry Pi you will need the following:

- Micro SD Card with the Raspbian operating system.
- USB mouse
- USB Keyboard
- Micro HDMI cable / HDMI Cable and HDMI to Micro HDMI adapter
- 5v 3A USB type C
- Raspberry Pi 4 model B (Preferred RPIs for this project are 3B, 3B+, and 4)

2.2.2 Instructions

The boot is done in the following steps:

1. Connect the USB mouse and keyboard to the USB slots at the back of the Raspberry Pi.
2. Connect the micro HDMI into HDMI slot 0 since it is the default.
3. Insert the micro SD card into the micro SD card slot.
4. Connect the type C USB into the Raspberry Pi. **NOTE: This step is the last step because the booting of the Raspberry Pi is direct and cannot be controlled. Therefore, as soon as the Pi is connected, the RPi will be booted instantly.** [1]

2.2.3 Troubleshooting

1. Firstly, if you boot the Raspberry Pi and there is no image output, make sure that the HDMI cable is connected to the port correctly from both ends, and if the micro HDMI is connected to HDMI input 0. If the issue continues, edit the `boot/config.txt` file and force the connection by enabling the *hdmi-force-hotplug* and selecting the HDMI mode by changing the *hdmi-drive* to 2. Then save the file and reboot the Pi. If the first solution did not work, enable *safe-mode* and save the file then reboot the device. If the issue continues, the reason may be insufficient power supply which is less than 2.5A.
2. If you receive **Boot not found** then either the operating system is not written correctly or the micro SD card is corrupted. And to know which file is exactly corrupted, you look into the green ACT/OK LED and how many times does it flash. If the LED flashed 3 times, then **start.elf** is not found or corrupted. If the LED flashed 4 times, then **start.elf** is not launched. If the LED flashed 7 times, then **kernel.img** is not found or corrupted. And if the LED flashed 8 times, then your SDRAM is not recognized or damaged. Which means either you need a newer **bootcode.bin**, **start.elf** **firmware**. The solution is to reinstall the operating system to the micro SD card.
3. If the screen is rainbow colored when it is the first boot, then wait for 5 minutes and it will go away. If the rainbow screen remained, then the error can be from different places. If the HDMI cable is connected to port to be used only when using two monitors. Also, make sure that you are using original SD cards such as Sandisk because the commercial SD cards can go through a lot of issues and errors. If the operating system is old, then update the operating system and try again. If none of the solution works, then the issue may be from insufficient power delivery to the Raspberry Pi which is from low quality type c wires. If the wire was fine then the final solution is to check the bootloader if it is not damaged by removing the SD card and reboot the raspberry pi and check if the green LED is flashing 4 times. If not, see the solution in <https://www.raspberrypi.org/forums/viewtopic.php?f=28&t=58151> [2]

2.3 Setting Up the Raspberry Pi

2.3.1 Enabling camera and SSH

Using raspi-config

1. Open Raspberry Pi Configuration Tool

```
1 $ sudo raspi-config
```

2. Select **Interfacing Options**
3. Select **Camera**
4. **¡Yes!**
5. **Finish** and exit.
6. Do the same thing for **SSH**.

Using Raspberry Pi Configurations in Preferences

1. Press on the Raspberry Pi image at the top.
2. Choose **Preferences**
3. Choose **Raspberry Pi Configurations**
4. Choose **Interfaces**
5. Enable the **Camera** and **SSH**

2.3.2 Expand SD card

1. After booting the Raspberry Pi successfully and setting up the device, start by increasing the SD card capacity to get more space for work

```
1 $ sudo raspi-config
```

2. Choose **Advanced Options** and **Expand Filesystem**, then reboot the Raspberry Pi.

3. **(OPTIONAL:** If you are having trouble with the space of the Pi, then uninstall and remove **libreoffice** and **wolfram engine** to free almost 1GB of memory.

```
1 $ sudo apt-get purge wolfram-engine
2 $ sudo apt-get purge libreoffice*
3 $ sudo apt-get clean
4 $ sudo apt-get autoremove
```

4. Then after the reboot, update the terminal's packages

```
1 $ sudo apt-get update
2 $ sudo apt-get upgrade
```

2.3.3 Install the OpenCV Dependencies

1. Now, after updating the system and packages, begin installing **cmake** and the other developer tools that will be essential for running opencv.

```
1 $ sudo apt-get install build-essential cmake pkg-config
```

2. Install the Image loaders(Packages that read image file types from the disk)

```
1 $ sudo apt-get install libjpeg-dev libtiff5-dev
   libjasper-dev libpng-dev
```

3. install the video readers (Packages that read video file types from the disk)

```
1 $ sudo apt-get install libavcodec-dev libavformat-dev
   libswscale-dev libv4l-dev
2 $ sudo apt-get install libxvidcore-dev libx264-dev
```

4. Install the **GTK** development library that is required by **highgui** in order to show image outputs on screen. And **atlas** for performing the complex matrix calculations needed for opencv on devices with limited resources such as the Raspberry Pi.


```

1 $ sudo apt-get install libfontconfig1-dev libcairo2-dev
2 $ sudo apt-get install libgdk-pixbuf2.0-dev libpango1.0-dev
3 $ sudo apt-get install libgtk2.0-dev libgtk-3-dev
4 $ sudo apt-get install libatlas-base-dev gfortran

```

5. Install the **qtgui** and **HDF5** datasets

```

1 $ sudo apt-get install libhdf5-dev libhdf5-serial-dev
  libhdf5-103
2 $ sudo apt-get install libqtgui4 libqtwebkit4 libqt4-test
  python3-pyqt5

```

6. install **python3** developer packages for getting the headers needed for compiling opencv since python 2.7 has ended

```

1 $ sudo apt-get install python3-dev

```

2.3.4 Installing Numpy and creating the virtual environment

After installing the packages needed for using and compiling opencv, you now can start the procedure of installing opencv. But before doing that, you need to create the virtual environment that you will use opencv with, and make sure that all the dependencies were installed correctly before proceeding.

1. start by updating or installing **pip** to the latest version.

```

1 $ wget https://bootstrap.pypa.io/get-pip.py
2 $ sudo python get-pip.py
3 $ sudo python3 get-pip.py
4 $ sudo rm -rf ~/.cache/pip

```

2. Install **virtualenv** and **virtualwrapper** for creating the virtual environments

```

1 $ sudo pip install virtualenv virtualenvwrapper

```

3. After installing the virtual environment tools, use **nano** to edit the **bashrc** file that runs upon booting the device.

```
1 $ nano ~/.bashrc
```

Then go to the end of the file and paste the following code that creates the variable of the virtual environment and virtual wrapper:

```
1 # virtualenv and virtualenvwrapper
2 export WORKON_HOME=$HOME/.virtualenvs
3 export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
4 source /usr/local/bin/virtualenvwrapper.sh
```

And save the file by pressing **ctrl+x** then **y** then **enter** .

4. Reload the source file by the following command to apply the changes.

```
1 $ source ~/.bashrc
```

5. Create the virtual environment. You can use any name but **cv** is preferred because it is not a complex name.

```
1 $ mkvirtualenv cv -p python3
```

6. This step is needed for the people who will use the camera module of Raspberry Pi.

```
1 $ pip install "picamera[array]"
```

2.3.5 Compiling OpenCV 4 from source

item In this step, you will use **wget** to get the source files of OpenCV from github.

```
1 $ cd ~
2 $ wget -O opencv.zip
   https://github.com/opencv/opencv/archive/4.1.1.zip
```

```

3  $ wget -O opencv_contrib.zip
    https://github.com/opencv/opencv_contrib/archive/4.1.1.zip
4  $ unzip opencv.zip
5  $ unzip opencv_contrib.zip
6  $ mv opencv-4.1.1 opencv
7  $ mv opencv_contrib-4.1.1 opencv_contrib

```

1. Increase the SWAP space in the `/etc/dphys-swapfile` file for running opencv on all the cores of Raspberry Pi for faster installation.

```

1  $ sudo nano /etc/dphys-swapfile

```

and change the size from 100MB to 2048MB to prevent hanging and lock ups

```

1  # set size to absolute value, leaving empty (default) then
    uses computed value
2  # you most likely don't want this, unless you have an
    special disk situation
3  # CONF_SWAPSIZE=100
4  CONF_SWAPSIZE=2048

```

and exit by pressing `ctrl` and `x` then `y` then `enter`.

2. Restart the `/etc/dphys-swapfile` file to apply changes

```

1  $ sudo /etc/init.d/dphys-swapfile stop
2  $ sudo /etc/init.d/dphys-swapfile start

```

3. Then compile opencv using the command:

```

1  $ workon cv

```

NOTE: If you received an error saying that "`cv`" does not exist or cannot compile the virtual environment "`cv`", then do the following every time you boot up the Raspberry Pi

- (a) go to `/.profile` file

```
1 $ sudo nano ~/.profile
```

(b) add the following code to the end of the `/.profile` file

```
1 # virtualenv and virtualenvwrapper
2 export WORKON_HOME=$HOME/.virtualenvs
3 export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
4 source /usr/local/bin/virtualenvwrapper.sh
```

(c) then restart the file by using **source**

```
1 $ source ~/.profile
```

4. If things are working perfectly, then you will see that the command line will look like this:

```
1 (cv) pi@raspberrypi:~$
```

then you install **numpy**

```
1 $ pip install numpy
```

and configure the build for opencv

```
1 $ cd ~/opencv
2 $ mkdir build
3 $ cd build
4 $ cmake -D CMAKE_BUILD_TYPE=RELEASE \
5     -D CMAKE_INSTALL_PREFIX=/usr/local \
6     -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
7     -D ENABLE_NEON=ON \
8     -D ENABLE_VFPV3=ON \
9     -D BUILD_TESTS=OFF \
10    -D INSTALL_PYTHON_EXAMPLES=OFF \
11    -D OPENCV_ENABLE_NONFREE=ON \
12    -D CMAKE_SHARED_LINKER_FLAGS=-latomic \
13    -D BUILD_EXAMPLES=OFF ..
```

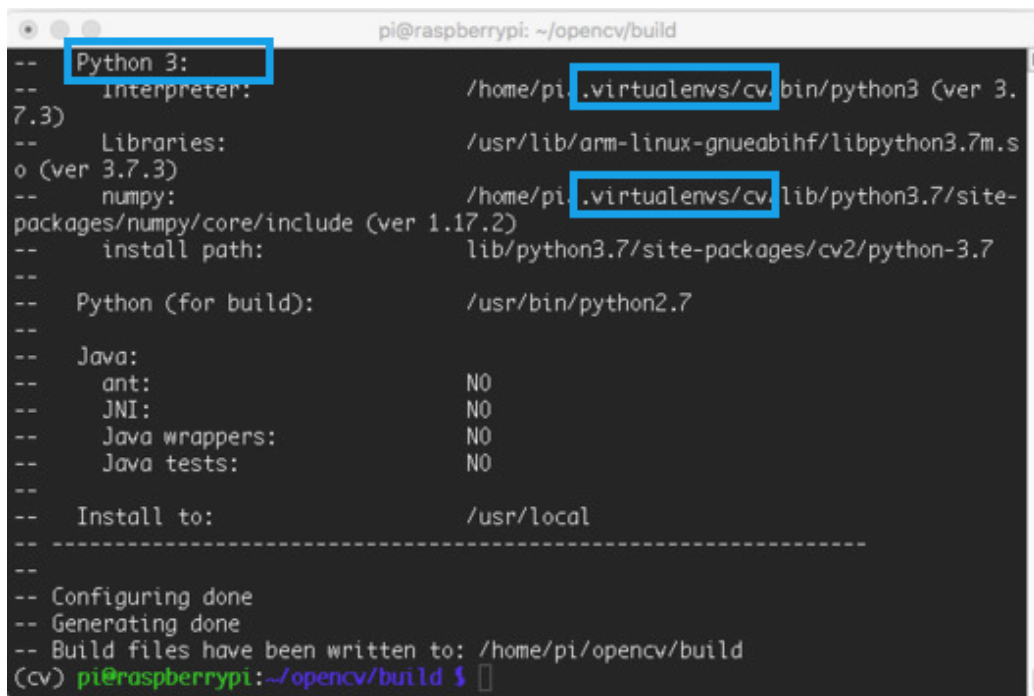
The flags of cmake in this process are four flags. **NEON** and **VFPV3** are set to **ON** to ensure that the compilation and optimization is at maximum level(Unless you are using a Raspberry Pi Zero W which is not compatible with NEON and VFPV3.

And the **NONFREE** is set to **ON** to provide the full version of opencv.

Finally, the **-latomic** flag is needed as in the source file of opencv.

NOTE: If the command *cmake* failed, make sure that you are in the directory of */opencv/build*

5. If the **cmake** command ran successfully, wait for the program to finish which can take up to 5 minutes. To know that **cmake** was finished properly, you must find that the output is in Python3. And the interpreter in addition to Numpy are in the directory of **.virtualenvs/cv** as can be seen in figure 2.3



```
pi@raspberrypi: ~/opencv/build
-- Python 3:
-- Interpreter:      /home/pi/.virtualenvs/cv/bin/python3 (ver 3.
7.3)
-- Libraries:       /usr/lib/arm-linux-gnueabi/libpython3.7m.s
o (ver 3.7.3)
-- numpy:           /home/pi/.virtualenvs/cv/lib/python3.7/site-
packages/numpy/core/include (ver 1.17.2)
-- install path:    lib/python3.7/site-packages/cv2/python-3.7
--
-- Python (for build): /usr/bin/python2.7
--
-- Java:
-- ant:             NO
-- JNI:             NO
-- Java wrappers:   NO
-- Java tests:      NO
--
-- Install to:      /usr/local
-----
-- Configuring done
-- Generating done
-- Build files have been written to: /home/pi/opencv/build
(cv) pi@raspberrypi:~/opencv/build $
```

Figure 2.3: Cmake Configuration Output.

And when the **Non-free algorithms** is set to **YES** as shown in figure 2.4

```
pi@raspberrypi: ~/opencv/build
-- OpenCV modules:
--   To be built:          aruco bgsegm bioinspired calib3d ccalib core
-- datasets dnn dnn_objdetect dpm face features2d flann freetype fuzzy gapi hdf hf
-- s highgui img_hash imgcodecs imgproc line_descriptor ml_objdetect optflow phase_
-- unwrapping photo plot python2 python3 quality reg rgbd saliency shape stereo sti
-- tching structured_light superres surface_matching text tracking ts video videoio
-- videostab xfeatures2d ximgproc xobjdetect xphoto
--   Disabled:            world
--   Disabled by dependency: -
--   Unavailable:          cnn_3dobj cudaarithm cudabgsegm cudacodec cu
-- dafeatures2d cudafilters cudaimgproc cudalegacy cudaobjdetect cudaoptflow cudast
-- ereo cudawarping cudev cvv java js matlab ovis sfm viz
--   Applications:        perf_tests apps
--   Documentation:        NO
-- Non-free algorithms:    YES
--
-- GUI:
--   GTK+:                 YES (ver 3.24.5)
--   GThread :             YES (ver 2.58.3)
--   GtkGlibExt:           NO
--   VTK support:          NO
--
-- Media I/O:
--   Zlib:                 /usr/lib/arm-linux-gnueabi/libz.so (ver 1.2.11)
```

Figure 2.4: Cmake Configuration Output of Non-free Algorithms.

6. Next, compile opencv on all four cores after preparing the source files and the Raspberry Pi for the compilation.

```
1 $ make -j4
```

The process will take at least an hour to four hours depending on which Raspberry Pi you have. The fastest time can be an hour using a Raspberry Pi 4.

7. When the compilation is successfully done after reaching 100%, install **make** which is the potimized version of OpenCV

```
1 $ sudo make install
2 $ sudo ldconfig
```

8. After the installation of opencv, go back to **/etc/dphys-swapfile** and reset the SWAP size to 100MB again or the SD card will be corrupted with time since increasing the SWAP size shortens the durability of the micro SD card.

```
1 $ sudo nano /etc/dphys-swapfile
```

Then restart the SWAP service

```
1 $ sudo /etc/init.d/dphys-swapfile stop
2 $ sudo /etc/init.d/dphys-swapfile start
```

9. Sys-link the opencv binding with the *cv* virtual environment which points the symbolic links with the files and directories in the disk.

```
1 $ cd /usr/local/lib/python3.7/site-packages/cv2/python-3.7
2 $ sudo mv cv2.cpython-37m-arm-linux-gnueabihf.so cv2.so
3 $ cd ~/.virtualenvs/cv/lib/python3.7/site-packages/
4 $ ln -s
    /usr/local/lib/python3.7/site-packages/cv2/python-3.7/cv2.so
    cv2.so
```

2.3.6 Test OpenCV Installation

To test if OpenCV was properly installed, run the following command:

```
1 $ cd ~
2 $ workon cv
3 $ python
4 >>> import cv2
5 >>> cv2.__version__
6 '4.1.1'
7 >>>
```

The result should be that the package *cv2* is imported without the error **No module named "cv2"** which occurs if you did not switch to the virtual environment of *opencv* before importing the package. Or if you faced some problems in the previous steps. The result on an actual Raspberry Pi can be seen in figure 2.5 [3]

```
pi@raspberrypi:~ $
pi@raspberrypi:~ $ source ~/.profile
-bash: ~/.profile: No such file or directory
pi@raspberrypi:~ $ source ~/.profile
pi@raspberrypi:~ $ workon cv
(cv) pi@raspberrypi:~ $ python
Python 3.7.3 (default, Jul 25 2020, 13:03:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'4.1.1'
>>>
```

Figure 2.5: Testing The Installation of OpenCV.

2.4 Create the Custom Haarcascade

2.4.1 Requirements

To create your own haarcascade, you will need a large number of data. At least 2000 negative images that can be any picture. The haarcascade training needs:

- Negative images

- Sample image
- OpenCV
- Python code **storeImages.py**
- **data**, **info**, **neg** directories

2.4.2 Instructions

The steps that are followed to create negative and positive images is to firstly use imageNet to get a database of URLs of any pictures. Use at least 3 imageNet URL links which can be at least 2000 or 3000 images.

1. Create the **data**, **info**, and **neg** directories using *mkdir* command.

```
1 $ mkdir neg
2 $ mkdir info
3 $ mkdir data
```

2. Use the python code **storeImages.py** to insert the URLs and run the code to download the images from the URL links.

NOTE: There will be so many errors such as "403 Forbidden" or "404 Not found". Because the image either became unlisted or private, or because the image was deleted. The program will continue adding pictures to the folders

NOTE: Make sure that you will save the images into the right file because the images will be overwritten if the code was ran with similar names

3. The same code will then create the background file **bg.txt** that contains the negative images and the **info.dat** that contains the images that will be used as positive training samples. **IF** you already have pictures of the drone without a background, then creating the list file will be better for the training.

NOTE: Before the next step, check the images because sometimes the image downloaded or used can be a corrupted image that will create an error for you later on when training the

haarcascade. So look for these images, and delete them before continuing.

4. After downloading the pictures and creating the background file, use **opencv-createsamples** to create positive images using the sample image. The command will place the sample image in the negative images in **bg.txt**, and store the coordinates of the sample image in the **info.lst** file. The new positive images will be stored in the **info** folder and the number of images that will be used for creating the positive images are 1950. Finally, the x,y,z angles are to decrease the angle of rotation and placement of the sample picture.

```
1  opencv_createsamples -img sample.jpg -bg bg.txt -info
   info/info.lst -pngoutput info
2  -maxxangle 0.5 -maxyangle 0.5 -maxzangle 0.5 -num 1950
```

5. After creating the positive samples, its time to create the vector file **positive.vec** that stores the positive images data using **opencv-createsamples**. The number of images used for training is 1950 as well or what number to your liking. Also, the width and height of the images must be at most 24 as recommended in the opencv documentation because if the image was bigger, the training will take more time and space in addition to the load on the RAMs which will be excessive load on the RAMs.

```
1  opencv_createsamples -info info/info.lst -num 1950
2  -w 20 -h 20 -vec positives.vec
```

```
1  opencv_createsamples -info info/info.lst -num 1950
2  -w 20 -h 20 -vec positives.vec
```

6. Next, its time to train the haarcascade using the **opencv-trainhaarcascade**, background file **bg.txt**, **positives.vec** vector file, and the haarcascade will be stored in the **data** directory. The training must be in the same size of the samples created. And the number of stages if increased, the accuracy will increase; but you will need more samples. And the number in **numPos** MUST be at most 1800 because the number increases

every stage. If the number was too high, the training will fail because the RAM is out of memory.

```
1  opencv_traincascade -data data -vec positives.vec -bg
    bg.txt -numPos 1800
2  -numNeg 900 -numStages 10 -w 20 -h 20
```

2.4.3 Troubleshooting

1. If you got an error during creating the positive samples in the **info** directory and the operation failed, then there is a corrupted image that led to the error creation.
2. If the error was in the process of crating the samples and the image was not found, check if there was a typo in the name of the image in **info.lst** file or if there was a space in the name of the image.
3. If you get the error of not being able to get new samples, the reason is the samples are not enough for the number of stages used. So the solution is either train the cascade on less stages or create more samples.
4. if the training is facing an error due to the size, then the width and height of the training data is more than the width and height of the samples. To fix it, use the same size used in **opencv-createsample**.
5. If you face the error of **std::out-of-range' what(): basic_string::substr: pos (which is 140) > this->size() (which is 11) Aborted (core dumped)**, then the error is in the image file as there is a corrupted image that was read that led to the abort of the operation.
6. If you face the error **Invalid background description file.** the reason is the file was created on windows and ran in linux. The solution is to install the **dos2unix** package and run the command to the file as shown below:

```
1  $ sudo apt-get install dos2unix
2  $ dos2unix bg.txt
```

If you have Geany text editor, you can open the text file and go to **Document, Set Line Endings, Convert and Set to LF(unix)**

2.5 Running the Python Codes

2.5.1 Requirements

- Raspberry Pi with OpenCV
- webcam / camera module
- Pan and Tilt Servo Motors
- Python Code **trackDrone.py**

2.5.2 Instructions

1. Make sure that the **cascade.xml** file is in the same directory of the or the path of the cascade is correct.
2. Correctly connect the servo motors to the right pins of the Raspberry Pi.
3. If the hardware connection is correct then activate the opencv virtual environment.
4. Install **imutils** package using the command:

```
1 $ pip install imutils
```

NOTE: Installing python packages and linux packages needs to be done in the virtual environment as well. So for example, if you try to use the *x* package outside the virtual environment, you will get the error "no module named x"

5. Move to the directory of the python code and run the code using the terminal [4]

```
1 $ cd capstone
2 $ python findDrone.py
```

2.5.3 Troubleshooting

1. If you received the error in the line of the `resize` or having the error of "Assertion failed", then the error is in the video output or a typo.
2. If you get an error in the line of the **`assert`**, then the error is in the connection of the servo motors since the function of the **`assert`** is to create a statement that if the angle is less than 0 or greater than 180, then the angle value is false.

Bibliography

- [1] P. Hut. My raspberry pi 4 will not boot/is faulty. [Online]. Available: <https://support.thepihut.com/hc/en-us/articles/360001887937-My-Raspberry-Pi-4-will-not-boot-is-faulty>
- [2] ——. Rainbow screen on the raspberry pi. [Online]. Available: <https://support.thepihut.com/hc/en-us/articles/360009035198-Rainbow-screen-on-the-Raspberry-Pi>
- [3] A. Rosebrock. (2019, September) Install opencv 4 on raspberry pi 4 and raspbian buster. [Online]. Available: <https://www.pyimagesearch.com/2019/09/16/install-opencv-4-on-raspberry-pi-4-and-raspbian-buster/>
- [4] H. Kinsley, “Creating your own haar cascade opencv python tutorial.” [Online]. Available: <https://pythonprogramming.net/haar-cascade-object-detection-python-opencv-tutorial/>