POLITECNICO DI MILANO

Department of Electronics, Information and Bioengineering

Master of Science in Computer Science and Engineering

TWO APPROACHES FOR FACE RECOGNITION WITH IOT TECHNOLOGIES

Supervisor: Assistant prof. Alessandro Redondi

Master Graduation Thesis By:

Uroš Popić

Matr. 841334

Academic year 2017/2018

ABSTRACT

Facial recognition system, has been seen back in 1960. A system that classifies photos of faces by hand using a RAND tablet, a device that people could use to input horizontal and vertical coordinates on a grid using a stylus that emits electromagnetic pulses. The system could manually record the coordinate locations of various facial features including the nose, hairline, eyes and mouth. (brief-history-of-face-recognition-software, 2018)

Since then, face recognition has evolved using different kinds of algorithms to identify a person as accurately as possible. After series of failures, verifying a person from a digital image or a video frame from a video source is today used in security systems, for law enforcement agencies, and state governments. Spreading to social media, through unlocking mobile devices, face recognition plays a big part in security and other similar technologies today. (Facial recognition system, 2018)

The Internet of things allows objects to be sensed or controlled remotely across existing network infrastructures. It converges of multiple technologies such as wireless communication, real-time analytics, machine learning, commodity sensors, and embedded systems. Iot's global market value is grossing quickly, it is estimated that it will reach $7.1 trillion by the year 2020. This gives perfect reason to use such systems for face recognition. It is easy distributable, easy-accessible, compatible and easy to deploy. (Internet_of_things, 2018)

In this thesis, we discuss about two facial recognition approaches using the Raspberry Pi 3 model (Iot device). We will include different technologies with same samples for matching results. We will study the physics of the approaches, we will compare these results to see which is better suited for what, and what are their pros and cons.

ACKNOWLEDGEMENT

Since the beginning of the project, it has been a new, exciting and challenging experience which will help me throughout my career.

I would like to express my sincere gratitude to my supervisors **Assistant professor Alessandro Redondi**, Department of Electronics, Information and Bioengineering, Politecnico Di Milano for guiding me and assisting me throughout the process. Any doubts or additional information were answered within the smallest period of time. Concise answers with information leading me in the right direction. The work was done right instead of quick which I am also thankful for. This allowed me to think different and understand the Machine Learning link.

I would also like to thank my family and my friends for being supportive and inspiring me to go further, do not give up and push harder in achieving my goals.

CONTENTS

LIST OF FIGURES:

1 INTRODUCTION

1.1 Context

The speed of technology advancement is far beyond what it was back in the 1982', when the first Internet-connected appliance has been developed. It was a modified Coke machine at Carnegie Mellon University which reported its inventory and when new loaded drinks were cold. Back then, Iot was described as packets of data to a large set of nodes, so as to integrate and automate everything from home appliances to entire factories. This field started showing signs of momentum in 1991 and became popular through the Auto-ID Center at MIT and related market-analysis publications. Implementing the Internet of things by equipping all objects in the world with machine-readable identifiers or tiny devices granted motion-picture publishers much more control over end-user private devices by remotely enforcing copyright restrictions and digital rights management. One major advancement was extending the "things" from the data generated from devices to objects in the physical space. This enables geographically dispersed users to cooperatively accomplish tasks and solve problems by using the network to actively promote the flow of material, energy, techniques, information, knowledge, and services in this environment. (Internet_of_things, 2018)

So, the idea is to make a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction. (internet-of-Things-IoT, 2018)

1.2 Problem statement

Although research for face recognition has been going on for a long time and a lot of algorithms and methods have been proposed, with great progresses achieved recently, face recognition technology is still developing and many papers on new face verification and recognition algorithms are being published almost daily. The reason is because problems still persist.

Problems such as big illumination variations. The direction of the light source may cause the image to be too bright or too dark, which disturbs the face recognition algorithm of getting the information of the face accurately. Today, this is still considered as a bottleneck as indicated from the tests of FERET (Facial Recognition Technology), FRVT (Face Recognition Vendor Tests) and FRGC (Face Recognition Grand Challenge) since illumination has a significant impact on the performance of face recognition.

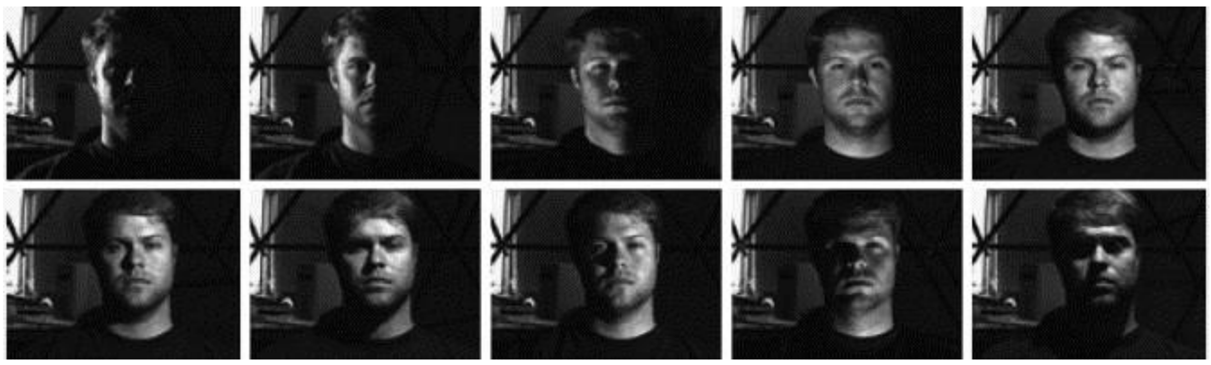


Figure 1-1 Same person in the same pose, but with different light exposure

Also posture and expression cause lower recognition rate.

Pose variation changes the whole shape and surface for the algorithm, which makes a drastic system performance drop. This means that more images (frames) are required for the same person to be recognized successfully.

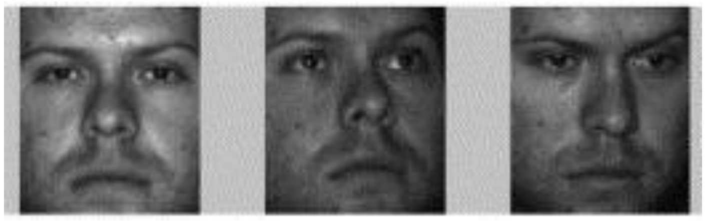


Figure 1-2 Same person in same illumination, but posing differently

Other problems relate to obstructed faces and application scalability issues which still have not been addressed to as much as the previously mentioned.

1.3 Proposed solutions

Images of the same person appear more different due to the change in lighting. If the change affected by this is larger than the difference between individuals, the system will not be able to recognize the input image. Researchers have proposed various methods for handling illumination problems. One suggestion is reducing the variation by discarding the most important eigenface. It has been shown that removing few of the first eigenfaces resulted in more precise accuracy. However, it caused system performance fall for input images taken under frontal illumination. Also, different images and distance measures were evaluated which lead to the conclusion that none of the methods were successful by themselves.

Furthermore, an illumination subspace for a person has been constructed for a fixed view point. This means that the recognition result could be illumination–invariant. Even though this method goes forward in solving the problem, it also needs many images per person to construct the basis images of illumination subspace.

One major method suggests using Principal Component Analysis (PCA) to solve parametric shape-from-shading (SFS) problem. It involves reconstructing a 3D face surface from a single image using computer vision techniques and computing the frontal view image under frontal illumination.

In the world, cameras are placed at different angles on different positions, therefore faces that appear in the scene are recognized from very different angles. Mathematically speaking, faces appear in different poses. To solve this problem, solutions have been divided in to three approaches:

1. multiple images per person are required in both training stage and recognition stage
2. multiple images per person are used in training stage but only one database image per person is available in recognition stage
3. single image-based methods

In the first one, the method is based on an illumination cone to deal with the illumination variation. Due to rotation, it needs to completely resolve the GBR (generalized-bas-relief) ambiguity when reconstructing 3D surface.

Second, also known as the Hybrid approach method includes three reprehensive methods: linear class-based method, graph matching based method and the view-based eigenface method. These methods have shown significant improvement for a rotating face. Nevertheless, these drawbacks are present: they require too many images per person, they only deal with the pose problem, separately from the illumination problem.

2 TECHNOLOGY

2.1 Raspberry pi 3

Also known as Rpi 3 is a small single-board computer developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and in developing countries. Several generations of Raspberry Pis have been released. All models feature a Broadcom system on a chip (SoC) with an integrated ARMcompatible central processing unit (CPU) and on-chip graphics processing unit (GPU).

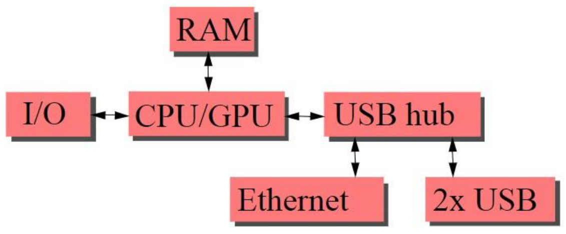


Figure 2-1 Raspberry Pi 3 hardware block diagram

The raspberry pi 3 model used for the study is equipped with:

- **SoC:** Broadcom BCM2837
- **CPU:** 4× ARM Cortex-A53, 1.2GHz
- **GPU:** Broadcom VideoCore IV
- **RAM:** 1GB LPDDR2 (900 MHz)
- **Networking:** 10/100 Ethernet, 2.4GHz 802.11n wireless
- **Bluetooth:** Bluetooth 4.1 Classic, Bluetooth Low Energy
- **Storage:** microSD
- **GPIO:** 40-pin header, populated

- **Ports:** HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

The most compatible system and specifically designed for this device is Raspbian.

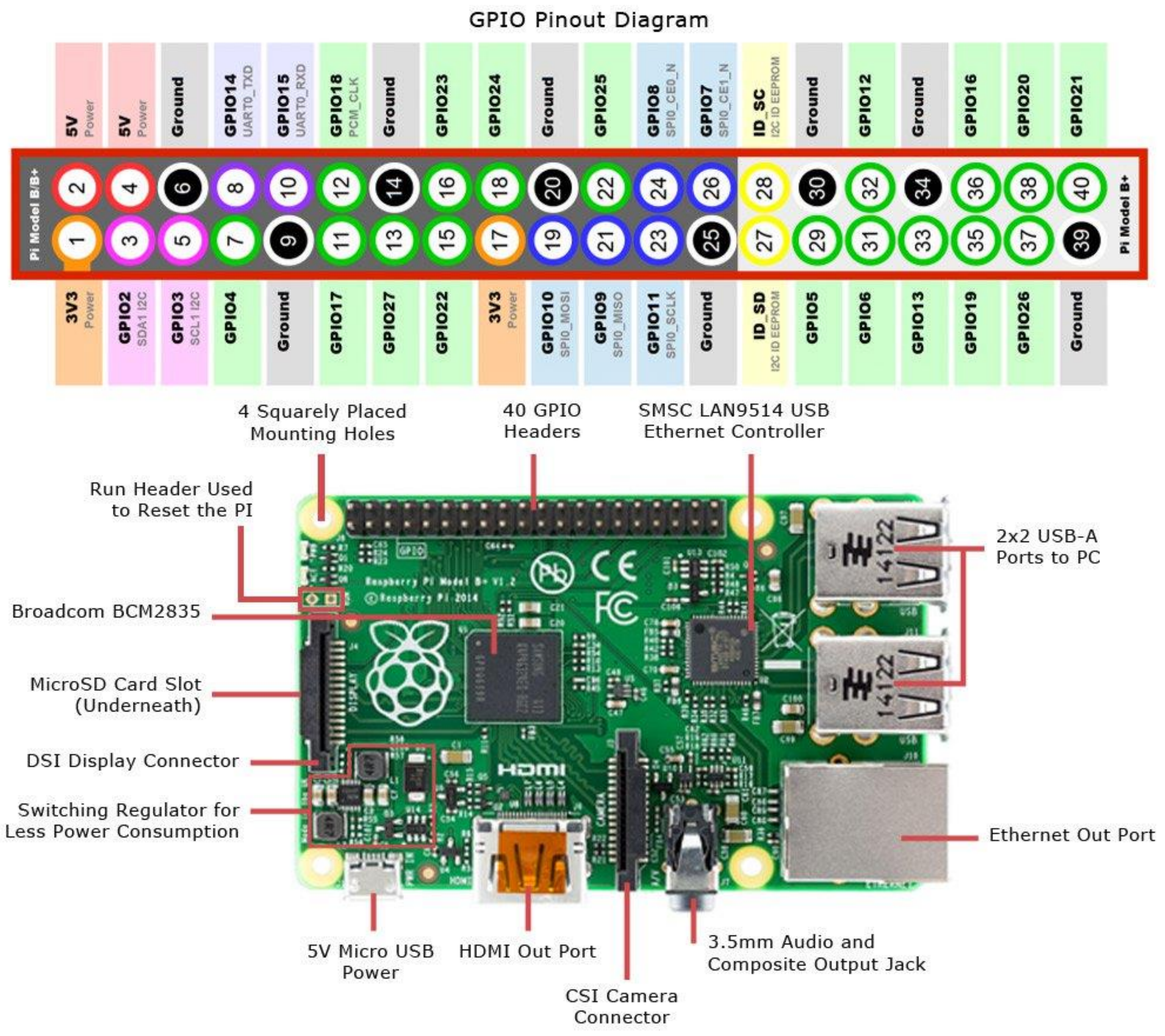


Figure 2-2 pi 3 board with pins described

2.1.1 Raspbian

Is an open-source Debian-based computer operating system for Raspberry Pi.

Figure 2-3 Raspbian logo

The operating system is still under active development.

Raspbian is highly optimized for the Raspberry Pi line's low-performance

ARM CPUs. (Raspbian, 2018)

During this research, the latest version was used: Debian 9, codename Stretch, which will be supported for the next 5 years. Numerous updated software packages were included, which enabled the latest versions of technology.

2.1.2 Raspbian setup

The official Raspberry website offers free download of the latest operating system. Both Desktop and Lite versions available.

Steps:

1. Download the image from https://www.raspberrypi.org/downloads/raspbian/
2. Download the Etcher flashing tool https://etcher.io
3. Flash the downloaded image onto a Micro SD card which will be used as storage for the Raspberry pi.

2.2 OpenCV (Open Source Computer Vision)

is a library of programming functions mainly aimed at real-time computer vision.

Used for computers to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.

Using methods for acquiring, processing, analyzing and understanding digital images, and extraction of high-dimensional data from the real world in order to produce numerical or symbolic information. (Computer_vision, 2018)

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE.

OpenCV comes with a trainer as well as detector. It facilitates face detection using Haar Feature-based Cascade Classifiers.

2.2.1 Haar feature-based cascade classifiers

OpenCV does object detection using Haar feature-based cascade classifiers proposed by Paul Viola and Michael Jones. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

The algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under white rectangle from sum of pixels under black rectangle.

Feature types:

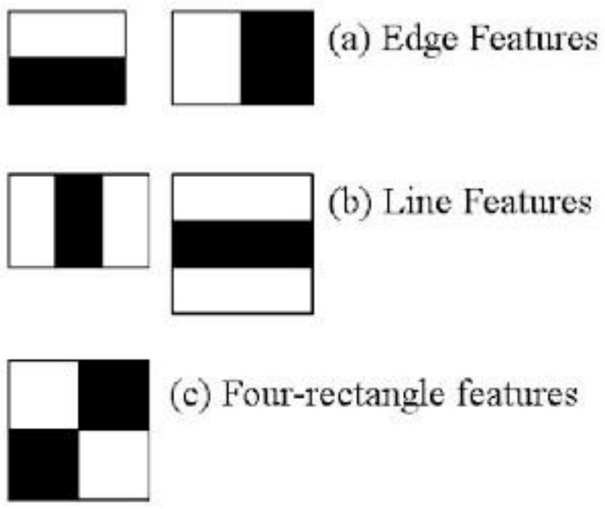


Figure 2-4 Haar features

(tutorial_py_face_detection.html, 2018)

2.2.2 OpenCV setup

Depending on the type of image flashed, OpenCV can be installed via Remote access via SSH or VNC or physically access our Raspberry Pi 3 and using the terminal to execute the commands.

Firstly, we should expand the filesystem to include all available space on the micro-SD card, by opening the terminal and running *sudo raspi-config* and then select the "Advanced Options" menu item. Select Expand filesystem and click finish.

1. Free up memory (optional):

- $ *sudo apt-get purge wolfram-engine*
- $ *sudo apt-get purge libreoffice**
- $ *sudo apt-get clean*
- $ *sudo apt-get autoremove*

2. Install dependencies:

- $ *sudo apt-get update && sudo apt-get upgrade*

  (update and upgrade any existing packages)

- *$ sudo apt-get install build-essential cmake pkg-config*

  (install some developer tools, including CMake)

- $ *sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev*

  (install image I/O packages that allow us to load various image file formats from disk)

- $ *sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev*

- $ *sudo apt-get install libxvidcore-dev libx264-dev*

  (install video I/O packages)

- $ *sudo apt-get install libgtk2.0-dev libgtk-3-dev*

  (install the GTK development library for sub-module named highgui)

- $ *sudo apt-get install libatlas-base-dev gfortran*

  (install matrix operations dependencies)

- $ *sudo apt-get install python2.7-dev python3-dev*

  (install both the Python 2.7 and Python 3 header files)

3. Download the OpenCV source code**:**

- $ *cd ~*

  (navigate to desired download directory)

- $ *wget -O opencv.zip https://github.com/Itseez/opencv/archive/3.3.0.zip*

- $ *unzip opencv.zip*

  (download OpenCV and unzip)

- $ *wget -O opencv_contrib.zip https://github.com/Itseez/opencv_contrib/archive/3.3.0.zip*

- $ *unzip opencv_contrib.zip*

  (download OpenCV contrib repository and unzip it)

4. Python 2.7 or Python 3:

- $ *wget https://bootstrap.pypa.io/get-pip.py*
- $ *sudo python get-pip.py*
- $ *sudo python3 get-pip.py*

  (download and install Python package manager)

- $ *sudo pip install virtualenv virtualenvwrapper*
- $ *sudo rm -rf ~/.cache/pip*

  (using virtual environment is good practice and optional)

- # virtualenv and virtualenvwrapper

  export WORKON_HOME=$HOME/.virtualenvs

  source /usr/local/bin/virtualenvwrapper.sh

  (update ~/.profile to include these two lines)

- $ *mkvirtualenv cv -p python2*

  (create virtual environment for python 2)

- $ *mkvirtualenv cv -p python3*

  (create virtual environment for python 3)

5. Install NumPy on the Raspberry Pi**:**

- $ *pip install numpy*

  (install numphy for numerical processing)

6. Compile and Install OpenCV**:**

- $ *workon* cv

(use cv virtual environment)

- $ *cd ~/opencv-3.3.0/*
- $ *mkdir build*
- $ *cd build*
- $ *cmake -D CMAKE_BUILD_TYPE=RELEASE *

  *-D CMAKE_INSTALL_PREFIX=/usr/local *

  *-D INSTALL_PYTHON_EXAMPLES=ON *

  *-D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib-3.3.0/modules *

  -D BUILD_EXAMPLES=ON ..

  (setup build using cmake)
- $ *make -j4*

  (compile OpenCV)
- $ *sudo make install*
- $ *sudo ldconfig*

  (install OpenCV 3)

7. Finish installing OpenCV on our Pi:

- $ *cd /usr/local/lib/python3.5/site-packages/*
- $ *sudo mv cv2.cpython-35m-arm-linux-gnueabihf.so cv2.so*

  (rename cv2.cpython-35m-arm-linux-gnueabihf.so)

2.3 Dlib (library)

is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high-performance computing environments. (dlib, 2018)

2.3.1 Dlib setup

1. For installing Dlib, we need to install Boost, Boost.Python, CMake and X11:

- $ *sudo apt-get update*
- $ *sudo apt-get install build-essential cmake*
- $ *sudo apt-get install libgtk-3-dev*
- $ *sudo apt-get install libboost-all-dev*

2. Access the cv virtual environment

- $ *workon cv*

3. Create a virtual environment for Python 2 or Python 3(optional)

- $ *mkvirtualenv py2_dlib*
- $ *mkvirtualenv py3_dlib -p python3*

4. Install NumPy, SciPy stack and scikit-image

- $ *pip install numpy*
- $ *pip install scipy*
- $ *pip install scikit-image*

5. Install dlib

- $ *pip install dlib*

2.4 Amazon Web Services

is a subsidiary of Amazon.com that provides on-demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis. The technology allows subscribers to have at their disposal a full-fledged virtual cluster of computers, available all the time, through the Internet. Aws's version of virtual computers have most of the attributes of a real computer including hardware (CPU(s) & GPU(s) for processing, local/RAM memory, hard-disk/ssd storage); a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, crm, etc. (Amazon_Web_Services, 2018)

2.4.1 Amazon S3 (Simple Storage Service)

is a web service offered by Amazon Web Services (AWS). Amazon S3 provides storage through web services interfaces (REST, SOAP, and BitTorrent).

2.4.2 Amazon IAM (Identity and Access Management)

Aws Identity and Access Management (IAM) enables secure control access to Aws services and resources the users of your resources. Using IAM, we can create and manage Aws users and groups, and use permissions to allow and deny their access to Aws resources.

2.4.3 Amazon Rekognition

Amazon Rekognition makes it easy to add image and video analysis to your applications. You just provide an image or video to the Rekognition API, and the service can identify objects, people, text, scenes, and activities. It can detect any inappropriate content as well. Amazon Rekognition also provides highly accurate facial analysis and facial recognition. You can detect, analyze, and compare faces for a wide variety of use cases, including user verification, cataloging, people counting, and public safety. Amazon Rekognition is based on the same proven, highly scalable, deep learning technology developed by Amazon's computer vision scientists to analyze billions of images and videos daily—and requires no machine learning expertise to use.
(rekognition, 2018)

2.4.4 Amazon CLI (Command Line Interface)

The Aws Command Line Interface (CLI) is a unified tool to manage your Aws services.

3 THE TWO APPROACHES

3.1 Recognition done locally

The first approach was done using the OpenCV library. OpenCV is an image processing library which offers face recognition using trained haar cascades. Haar cascades are used for detecting objects and faces. Here the system is provided with a couple of positive and negative images, and the feature selection is done along with the classifier training using Adaboost and Integral images.

First, the cascades detect the face and then the OpenCV processes that image for recognition. In order to recognize a person, sample images, referred to as a training set had to be stored locally as target images. The app is built on a Raspbian operating system using the Python scripting language.

In this case, our source is a simple Logitech usb camera used for a live video feed. The idea is, to draw a rectangle around the face, once it has been detected and recognize any faces that appear in the scene of the camera lens. Even though OpenCV and haar cascades look like a logical way to detect and then recognize faces, the tests have shown that this combination seems to detect faces that are not there. For no reason, rectangles were appearing simultaneously in the scene, when no person appeared in the video. This shows that algorithm has failed reaching the desired standard of recognition, and the problem hasn’t been addressed, even though 3.3 version of OpenCV was used, which was the latest version during this research.

After further analysis, the best solution was using dlib's state-of-the-art face recognition built with deep learning. A very powerful tool which has attained 99.38% accuracy in the Labeled Faces in the Wild benchmark. A training set, which was comprised of 31 images, was used to compare to the 31 images of the same people, but with different facial expressions. The script uses dlib's Python bindings to extract facial landmarks.

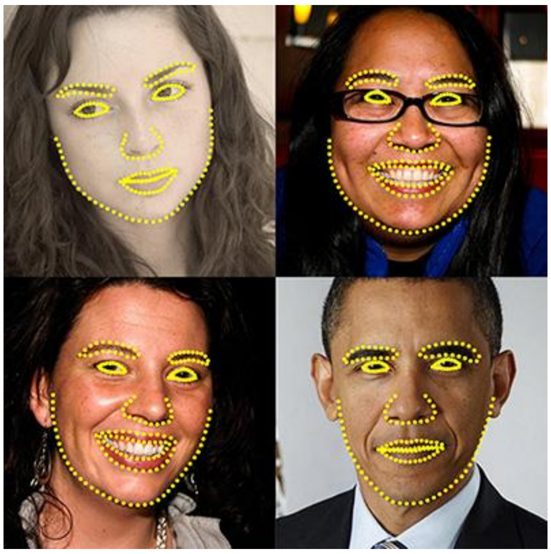


Figure 3-1 Dlib facial landmarks

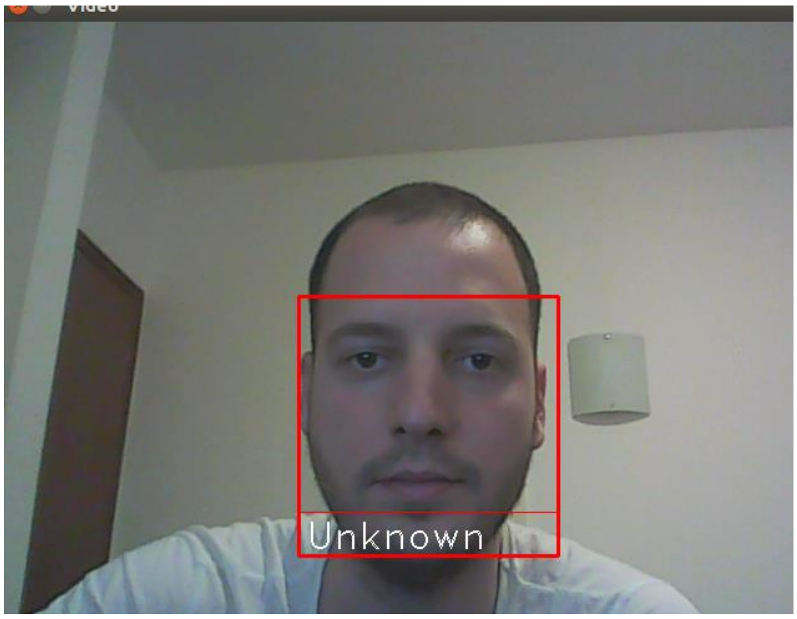


Figure 3-2 Face detection without a training set example

The test proven itself to be a success, showing high accuracy with high responsiveness. Instant rectangle with the person's name underneath it. The face in the scene is compared directly with the image stored in our local database.

Before running the script, we have to install the face_recognition module from pypi using the following command:

$ *pip install face_recognition*

Depending on which python we want to use, we can install either with pip2 or pip3.

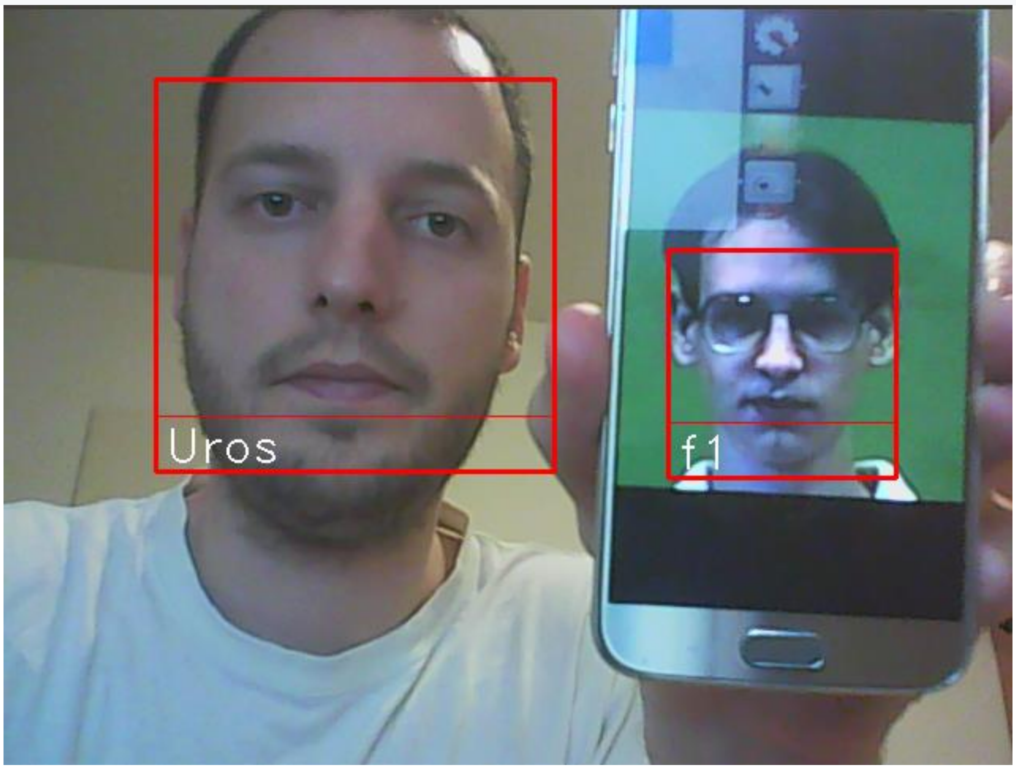


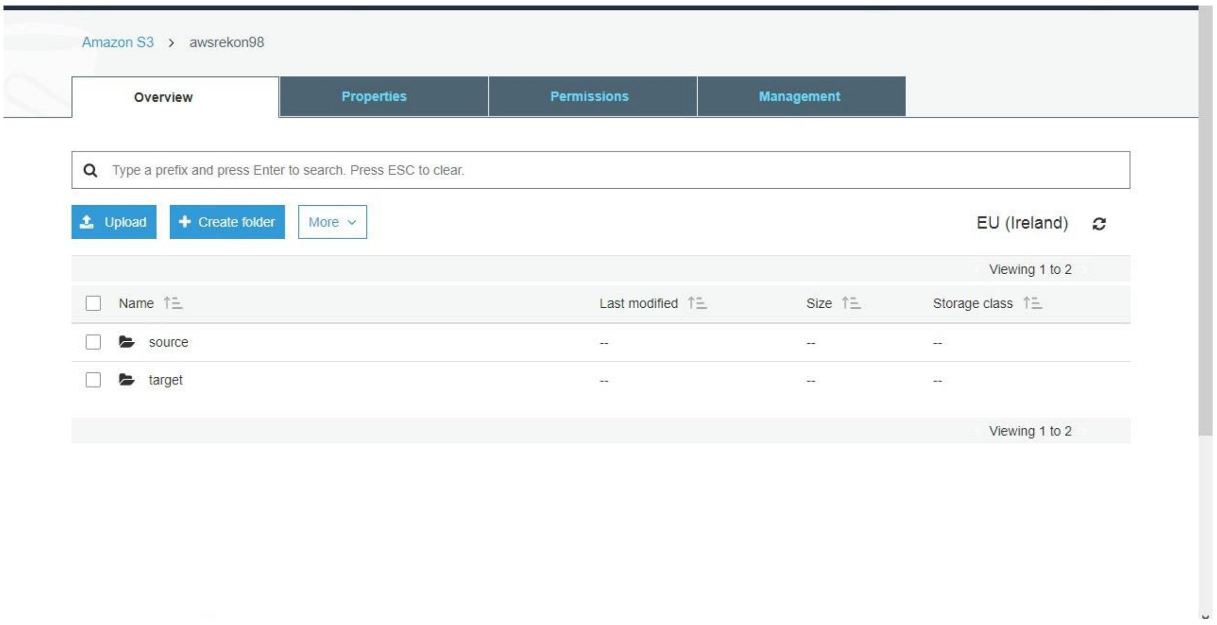
Figure 3-3 Multiple face detection at the same time

3.2 Recognition done using an online server

The idea of the second approach was to use Amazons new face recognition libraries to compare faces. Instead of using a web camera, both the source images and the target images are uploaded on to an online database, in this case, Amazons S3 service. Here, buckets are made, acting as a database which the Rekognition API uses while processing. In each bucket, we have to set its region.

The Amazons Rekognition library also uses deep learning technology developed by Amazon's computer vision scientists.

The API was manipulated also using the Raspbian operating system. Once the training set and the test set were uploaded, read/write permissions had to be given. Amazon Rekognition's support is limited to JPG and PNG formats.



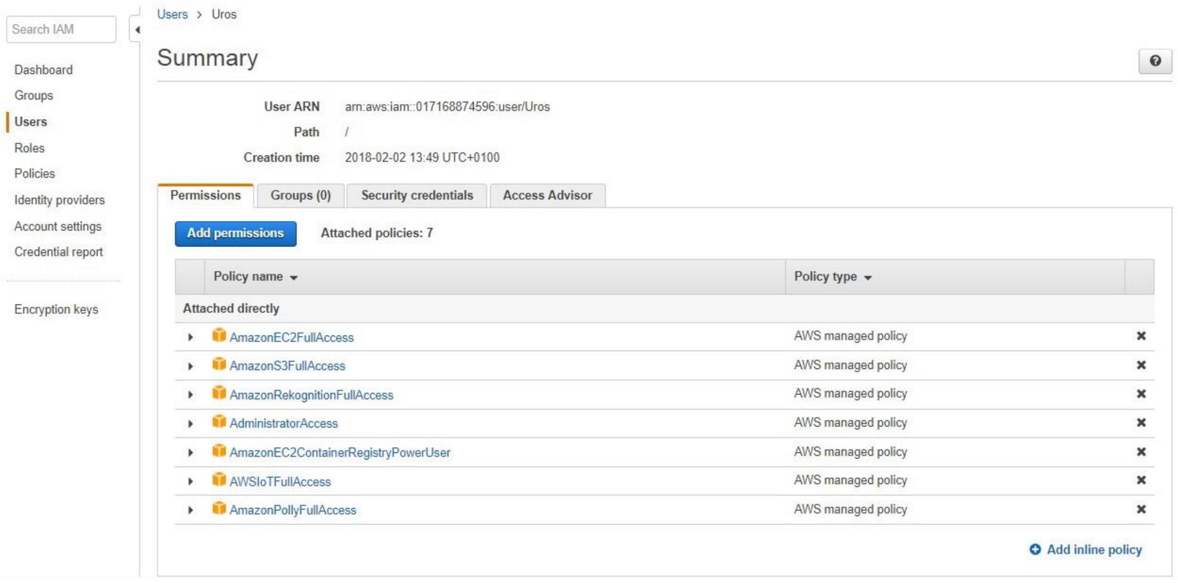
Viewing 1 to 2

Name	Last modified	Size	Storage class
source	--	--	--
target	--	--	--

Viewing 1 to 2

Figure 3-4 Aws bucket

The next step was to set up the configuration for the Aws Command Line Interface, by creating a named profile for the Administrator user(us) and entering the credentials assigned upon creation. When we create a profile in the Aws Identity and Access Management, we receive an Aws access key id and an Aws secret access key. Before that, we need to grant the user certain permissions to access Aws resources.



Summary

User ARN arn:aws:iam::017168874596:user/Uros

Path /

Creation time 2018-02-02 13:49 UTC+0100

Attached policies: 7

Policy name	Policy type
Attached directly	
AmazonEC2FullAccess	AWS managed policy
AmazonS3FullAccess	AWS managed policy
AmazonRekognitionFullAccess	AWS managed policy
AdministratorAccess	AWS managed policy
AmazonEC2ContainerRegistryPowerUser	AWS managed policy
AWSIoTFullAccess	AWS managed policy
AmazonPollyFullAccess	AWS managed policy

Figure 3-5 Aws user permissions

The last configuration point is the Region.

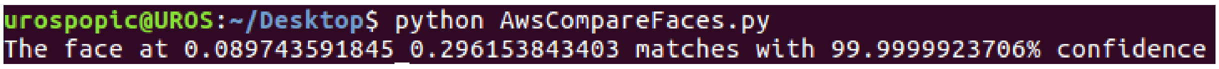
The list can be found in the Amazon Web Services General Reference. This means, that the region of the bucket needs to be the same as the region of our CLI configuration.

The process can be done by directly writing into the command line, or just by running the Python script. The first process is a base64-encoded representation of the input image, serialized as part of the API request. And the second is just the reference to the image hosted on S3. The second option would be better, because it is more efficient and easier to use and it allows API consumers to avoid network inefficiency and reuse uploaded files.

All in all, the accuracy results will remain the same from both methods.

The output we get is an array of face matches, source face information, source and target image orientation, and an array of unmatched faces. For each matching face in the target image, the response provides a similarity score (how similar the face is to the source face) and face metadata such as the bounding box of the matching face and an array of facial landmarks. In order to run the python script, we need to install boto3.

Boto is the Amazon Web Services (AWS) SDK for Python, which allows Python developers to write software that makes use of Amazon services like S3 and EC2. Boto provides an easy to use, object-oriented API as well as low-level direct service access.



```
urospopic@UROS:~/Desktop$ python AwsCompareFaces.py
The face at 0.089743591845 0.296153843403 matches with 99.9999923706% confidence
```

Figure 3-6 Aws face_compare output example

4 PERFORMANCE COMPARISON

After setting up both types of recognition, tests were performed for each, in a different way, to understand the algorithms and retrieve the accuracy results. Both tests had similar results, but not the same procedure. Each approach had its own limitations which dictated the cross-referencing faze. Tests were performed on images that were downloaded from an internet database of face samples.

4.1 Recognition done locally

In the first recognition test, I have taken 5 pictures as reference faces, and 20 comparison images to cross-reference. Since the testing could not fully be computerized, the results are 99,9% accurate. For each of the 20 images taken for cross-reference, they were presented in front of the camera in a span of 15 second. The 5 images were the same people from the reference set, but with a different facial expression, and from a different angle. The rest were different people with the same lighting, same green background and similar hair and beard styles. Some had the same accessories, in this case, eyeglasses. Mismatches were witnessed only on the images that were the exact same person. These mismatches happened at most 6 times. Mismatch in the sense that when we move a cross-referencing image in the video back and forth or change the angle or cover a small portion of the face, the name would change only for a split second. More accurately, 0.03 seconds. Which means that in the case that a mismatch happened 6 times, resulted in a 0.18 second mismatch within 15 seconds, which is highly reliable. When the image is held stable, the accuracy is very close to 100%.

The best way to depict the accuracy is by populating a table just to see closely how the values differ very little from one another.

Table 4-1 Accuracy table of the local recognition

Accuracy	A	B	C	D	E
A1	98,9959%	0,001%	0,001%	0,001%	0,001%
B1	0,001%	98,9885%	0,001%	0,001%	0,001%
C1	0,001%	0,001%	98,9898%	0,001%	0,001%
D1	0,001%	0,001%	0,001%	98,9939%	0,001%
E1	0,001%	0,001%	0,001%	0,001%	98,9939%
F1	0,001%	0,001%	0,001%	0,001%	0,001%
G1	0,001%	0,001%	0,001%	0,001%	0,001%
H1	0,001%	0,001%	0,001%	0,001%	0,001%
I1	0,001%	0,001%	0,001%	0,001%	0,001%
J1	0,001%	0,001%	0,001%	0,001%	0,001%
K1	0,001%	0,001%	0,001%	0,001%	0,001%
L1	0,001%	0,001%	0,001%	0,001%	0,001%
M1	0,001%	0,001%	0,001%	0,001%	0,001%
N1	0,001%	0,001%	0,001%	0,001%	0,001%
O1	0,001%	0,001%	0,001%	0,001%	0,001%
P1	0,001%	0,001%	0,001%	0,001%	0,001%
Q1	0,001%	0,001%	0,001%	0,001%	0,001%
R1	0,001%	0,001%	0,001%	0,001%	0,001%

S1	0,001%	0,001%	0,001%	0,001%	0,001%
T1	0,001%	0,001%	0,001%	0,001%	0,001%

This test was performed in 3 different time spans. Firstly, with a duration of 10 seconds, then adding 5 seconds to it, and finally adding 10 seconds. Even though different timing was used to measure the accuracy, in the end, the results were the same. This lead to the conclusion that slight mismatches will always happen in the same way and at a fairly similar time, regardless of the time factor.

In this table (4-1), as mentioned, almost absolute precision from 98,9885% counted as the lowest till 98,9959% counted as the highest score was when the source image person was the same person in the target image. Names A, B, C, D, E are assigned to the source images. While, on the other hand, the indexed letters on the right, are assigned to the target images for cross-referencing. The index points out that in this picture, the angle of the face (position) is different from the source picture. The 0,001% value is because the same thing of a wrong name appearance in a duration of 0,03 seconds happened 2 or 3 times. This means that it also might have happened for the correct match.

4.2 Recognition done using an online server

For the second type of recognition strategy, in the aws_compare_faces.py file, I have set three parameters to be parsed to the aws server. The aws cli enables us to use a simple three argument passing philosophy. Before the easy step, we have to set all parameters previously. These parameters include the s3 bucket zone parameter, the buckets permissions, the uploaded images permissions, the IAM user permissions, the access id key, the secret access key, the default region type of cli and the cli output format. After stating the users account, we access the bucket by declaring the buckets name. The other two parameters are the source and the target image, where we declare the path of the files and their full names. As a response, we get a json object that is comprised of the faces position and the match percentage.

In this approach I have uploaded just a single target image, and it is made of 20 people. The algorithm checks every single face for a match, while the source image is a single person. This test was run 5 times. Each time a different face was searched within the same 20 people, which means that in this code I just change the source file parameter every time I change an image, since in this test, there are only 5 different people to process. The resulting table is as follows:

Source	Position X	Position Y	Confidence
A	0,4412698477	0,0329999998212	99,9981918335%
B	0,15714286678	0,0599999986589	99,9999160767%
C	0,321428567171	0, 0450000017881	99,9959793091%
D	0,160317465663	0,243000000715	99,9999008179%
E	0,593650817871	0,0480000004172	99,9997406006%

Briefly described, even if we add more reference faces, the execution time will still be the same. Only if the target image number of people increases, the execution time increases, since the algorithm has its own process time for every new face it finds. Having that said, the duration is pretty short, for a fair number of faces, taking around 4 seconds. This increases linearly, rather than exponentially.

Just to be sure, I have added a timer to the program to precisely measure the duration. The timer ends as soon as the response is received. The timings for the first try of persons A, B, C, D and E were:

- 4.6242s (A)
- 4.7594s (B)
- 5.7679s (C)

- 5.0732s (D)
- 5.3005s (E)

In the second go it was:

- 4.8825s (A)
- 4.9048s (B)
- 5.1928s (C)
- 4.848s (D)
- 4.8931s (E)

Each time we run the algorithm, the response time is different, but similar.

5 CONCLUSIONS

In todays day and age, security plays one of the most important role when we talk about human recognition. Used in law enforcement, social media tagging, online payment, all sorts of authorizations, fraud detection and many others. However, security systems are constantly threatened by scammers of all kinds, which means that we need to constantly go forward in building better and more precise algorithms to enhance the efficiency of detection in any kind of environment exposed to. Even though face detection is present, it is still not sufficient when it comes to safety. We don't just want to detect the culprit, we want to send this data instantly to the authorities and instantly identify this person. For having such a security system, can be very expensive for personal use. In the future, the Raspberry pi shows promises as cheap full prof security system.

Raspberry pi is physically light, easy to transport cause of size convenience, has fast computational power for its size, easy to setup, very cheap to have, very easy to maintain and highly versatile cause of its I/O compatibility. Among many of its advantages, the Rpi's GPIO pins, introduce a whole new world of sensors. This means that the device can be used in bioengineering, chemical engineering, mechanical engineering, environmental engineering and so on. One drawback of being a silent device is that it has no cooling system, meaning it does not respond well to temperature. It CPU will overheat and may cause damage to the board and other components attached to it. During this research, the Rpi performed well for all of the tasks, which means that generally it will do so, if not abused intentionally by high temperatures.

After thoroughly testing the raspberry pi with both face recognition approaches, I came with significant conclusions.

5.1 Recognition done locally

Recognition done locally is a highly responsive method to use, as it delivers precision with minimum waiting time. It is very accurate and the results are instantly shown. The high accuracy of around 98% makes it a reliable choice. This approach using dlib technology does not need an internet connection. It does not connect to any online server, introducing zero wait time and local storage data processing. All computation is done at run time and the system fully works without internet.

Updating means introducing only a few lines of code and adding a picture of the person to recognize. Even though the easiness of manageability makes it convenient for personal use, the maintenance still has to be done by the user of the device.

Also, the local storage of the Rpi can be limited, which can be a problem if it is used for a company that has too many employees that have to be recognized in order to gain entrance authority and facial detection can fail if the target is exposed to a very dark environment or if the target is too far away from the camera.

Overall, this approach has proved itself more than worthy of the task.

5.2 Recognition done using an online server

The difference in this approach, is that we solely rely on the internet connection. The Aws recognition is highly accurate in detection and works well with distant faces. A picture of many faces is completely processed. Even with a slightly darker environment, it will apply the recognition algorithm. Recognition from distance and with low light is a very important factor and very hard to achieve.

In this type of approach, all computation is done server side, and the user will not be concerned with maintaining his own code. Adding a face to the server is straightforward and convenient. However, if there is no internet connection, or if something happened to the server, there is not much a user can do at that point. The whole detection is very slow. If we want instant detection, we have to detect the face, take a snapshot, send image to the server, then processed the image and then the response has to be sent back to the user which is inefficient for this kind of use.

Both of these approaches have their advantages and disadvantages. The Aws Rekognition is slightly more accurate, but the Local recognition using dlib is immensely faster. The first approach is more reliable for the full process, but the second approach is more reliable for the detection.

A lot can be said regarding the two, but in the end, both have helped us further understand how image processing for face recognition works and how we can introduce and distribute such complexity with minimum cost and effort.

6 FUTURE WORK

Face recognition is widely used today and is more available for personal use. Nevertheless, it is just a small percentage.

For example, in school facilities, everyone is allowed to enter, even though there are campaigns that stress out the threats of low security level. Certain educational facilities have keycard access, which is better, but still offers an inefficient level of security.

A proposed idea would be to have a face recognition system at certain entrance points of the facility which should be present worldwide. A raspberry pi that can trigger a simple mechanism built in the door to unlock upon recognition for the person that has access permission.

This would ensure that not anyone can enter. Also, save time on having to use a keycard every time. We only want certain people to use the school library or the school playground for example. This would be the school personnel, the children attending and the parents. By using this system, we can also keep track of unregistered unauthorized people trying to access without permission.

Also, in most companies today, the personnel use a keycard to open the gate in order to park their car inside. They use the keycard to register the time they have arrived and the time they have left from work, keycard use for entering using a back door and so on. All movement is recorded, which still does not mean that an unauthorized person cannot steal a keycard and abuse the system and take advantage.

By introducing a surveillance system that unlocks or locks a door when a person is recognized, we can keep track of the time of the people that come and go, we can manage the permission of who can and who cannot enter a certain room, we have saved time on having to use our keycard every time and most importantly, we have drastically increased the security and safety of the facility. Such a system like this exists but is not common in everyday use.

7 REFERENCES

7.1 Retrieved from

- Amazon_Web_Services. (2018). Retrieved from wiki:
- https://en.wikipedia.org/wiki/Amazon_Web_Services

- brief-history-of-face-recognition-software. (2018). Retrieved from https://www.facefirst.com/blog/brief-history-of-face-recognition-software/

- Computer_vision. (2018). Retrieved from wiki: https://en.wikipedia.org/wiki/Computer_vision

- dlib. (2018). Retrieved from dlib: http://dlib.net

- Facial recognition system. (2018). Retrieved from Wikipedia:

- https://en.wikipedia.org/wiki/Facial_recognition_system

- Internet_of_things. (2018). Retrieved from wiki: https://en.wikipedia.org/wiki/Internet_of_things

- internet-of-Things-IoT. (2018). Retrieved from internetofthingsagenda.techtarget: http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

- Raspbian. (2018). Retrieved from wiki: https://en.wikipedia.org/wiki/Raspbian

- rekognition. (2018). Retrieved from amazon:
- https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html

- tutorial_py_face_detection.html. (2018). Retrieved from docs.opencv.org: https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html

7.2 Read from

- RS Raji (1994) - "Smart networks for control".

- Friedemann, Floerkemeier (2010) - "From the Internet of Computers to the Internet of Things".

- Weiser (1991) - "The Computer for the 21st Century" .

- Pontin (2005) - "ETC: Bill Joy's Six Webs".
- Gaddam (2000) - "M-commerce key to ubiquitous internet".
- Magrassi, Panarella, Deighton, Johnson (2001) - "Computers to Acquire Control of the Physical World".
- Zhuge (2005) - "The Future Interconnection Environment".
- http://ieeexplore.ieee.org/document/5366328/?reload=true
- https://realpython.com/blog/python/python-virtual-environments-a-primer/
- https://www.pyimagesearch.com/2017/09/04/raspbian-stretch-install-opencv-3-python-on-your-raspberry-pi/
- https://www.pyimagesearch.com/2017/05/01/install-dlib-raspberry-pi/