Department of Computer Science and Engineering

Graduation Project

A. Eye

Artificial Eye

By

Supervisor Head of the Department Dean

Declaration

We hereby declare that the work presented in this project is the result of our own independent work.

Names :

1-

2-

3-

4-

5-

6-

Date

Abstract

Abstract of about 500 words.

Acknowledgments

There are no formal rules here, but it is generally appropriate to acknowledge supervisors, funders, participants, and others in the academy who helped in the PhD process.
You can also acknowledge people from your personal life who supported your efforts.

Table of contents

List of figures

List of tables

Chapter 1: Introduction

Chapter 2: Feasibility Study

2.1 Introduction

2.2 Operational Feasibility

2.2.1 Applications Scenarios

We had a general idea of using a device based on artificial intelligence technology to help the public in one way or another. Therefore we searched for some possible scenarios to implement our idea through. The ideas we thought of were:

- Traffic control.
- Object counting in any possible application.
- Attendance system through face recognition.
- **Text reading.**
- **Assisting blind and visually impaired people.**

Through researching some more, we found a significant need for technology to help the visually impaired/blind individuals. So we have chosen to help this group by providing a standalone product to work as a voice assistant to help them navigate their environment and interact with it more easily without needing help and we provide an additional feature of text reading since it's a significant help to this group.

2.2.2 Problem Urgency

Vision Impairment

There are about 2.2 billion people who are diagnosed with near or distant vision impairment worldwide. One billion of them have severe vision impairment or blindness[1]. In Egypt there are 3 millions visually impaired individual and one million blind[2].

Based on some statistics, many people develop vision impairment over the age of 50. However, the loss of vision can affect people of all ages. Causes of vision loss vary, some examples are cataract, uncorrected refractive errors, age, corneal opacity, glaucoma, trachoma,.. etc[1].

Prosopagnosia (Face Blindness)

Prosopagnosia is the inability to recognize people's faces. There are 2 types of prosopagnosia: developmental prosopagnosia and acquired prosopagnosia[3].

Such a disease is thought to be rare but based on a researches and tests made by Ken Nakayama and Richard Russell[4][5], about 2 percent of the population are affected with that disease.

In the early days, most of prosopagnosia cases were thought to be due to a brain injury. Later on researchers have discovered that there are many people that have prosopagnosia without the cause of brain injury[3].

Dyslexia (Reading Disorder)

It is a known learning difficulty where the patient has problems with reading, writing, and sometimes spelling. People with dyslexia have difficulty recognizing different sounds that make up the words. It is not related to the person intelligence[6].

Statistics in 2017 suggest that there are approximately about 5% to 10% of the worldwide population suffering from dyslexia[7]. The cause of dyslexia is still unknown. But researchers found that it often shows to run in families[6].

2.2.3 Solutions

The solutions we came up with is to make a device that helps the user to:

- Detect objects and obstacles that s/he might encounter.
- Recognize faces of acquaintances, friends, and family members.
- Read text.

2.2.4 Survey

We have visited eye hospital in Menouf. We interviewed the hospital manager and several doctors and asked them several questions (see appendix A for the questionnaire). From the results we have concluded that:

- The project is very useful and will make a huge difference in the lifestyle of the end user.
- According to the specialists we interviewed, the expected price for a device to perform this function is expected to be very expensive from the commercial point of view. However, we are aiming to design and implement a low-cost device to serve our Egyptian community.
- If such a device is made in the market it would be recommended to the patients by doctors.
- They have no proposal for additional features.
- If the device is mass produced and it has spread in the market many charity foundations would gladly provide such a device for free for the incapable patients.
- Nearly all the specialists agreed to participate in a clinical trial of this device.

2.3 Technical Feasibility

2.3.1 Labor

The labor of the project consist of its three team members:

- One hardware engineer.
- Two programmers.

2.3.2 Location

According to our chosen application scenarios, the project is multi-purposed and doesn't depend on a specific location in most cases; to assist a blind person in everyday tasks, the project needs to function in the general environment and to read text for the client, the device can’t have a location dependency for such a task.

2.3.3 Technology

The project consists of 3 parts. One part is concerned with the software and two parts are concerned with hardware.

Software

Our software consists mainly of python libraries that are used in object detection, face recognition, OCR, and text-to-speech functions. Those libraries are downloaded on a RasPian OS that is installed on the device. Those libraries are:

- OpenCV
- numpy
- pytesseract
- gtts
- os

Main Apparatus

Consists of the main parts responsible for the input, processing, and output of the project. Which are consecutively:

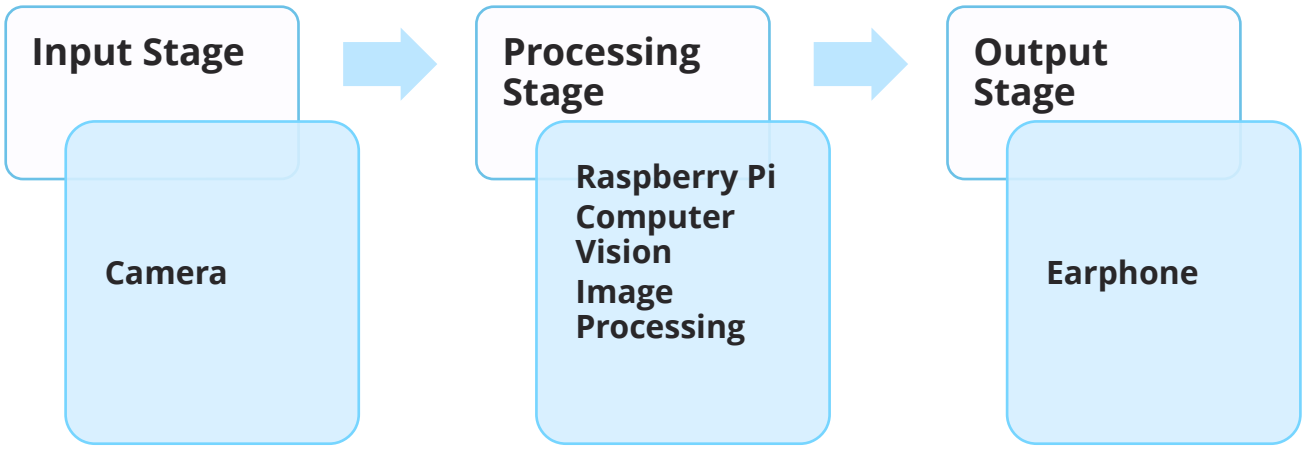
- Raspberry Pi Camera (for image input).
- Raspberry Pi (computer for processing and the main device).
- Earphone (for voice output).

Raspberry Pi Facilities

These components are connected to the raspberry pi to provide either needed or optional functionalities. They are categorized into two sections:

- Primary facilities:
  - Power bank (as the device power source).
  - Raspberry Pi case (for protection).
  - Heat sinks (for heat dissipation).
  - MicroSD card (for installing OS and any software component on it).
- Secondary facilities:
  - HDMI cable (for Raspberry Pi screen viewing while programming).
  - I/O facilities: keyboard, mouse,.. etc.

2.3.4 Block Diagram



1. At the input stage: the camera takes an image of the text/environment/object depending on the mode of operation that the user is using.
2. The raspberry pi fetches the image and starts to perform its operations, whether it's object detection, face recognition, or text recognition, using its corresponding python class and translates the text output to an audio file using a text-to-speech class to send to the output device (earphones) connected to it.
3. The sound file is then played on the earphones for the customer to listen to.

2.4 Market Research

2.4.1 Customer-base

Our customer-base consists of visually impaired people and blind people and their needs are:

- Moving freely without the need for human help.
- Recognizing surrounding objects.
- Avoiding accidents usually caused by the impairment.
- Reading text.
- Being able to detect and recognize known acquaintances and family members.

2.4.2 Competition

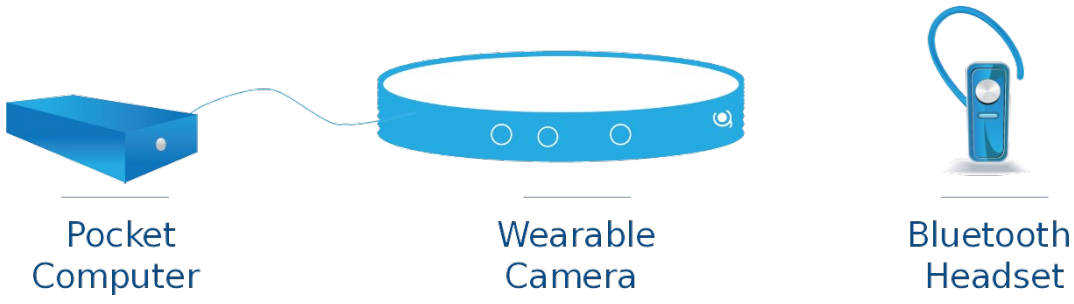
Low-Vision Aids – المعينات البصرية

They are a governmental agency that provides three classes of services to the visually impaired patients:

- Classical vision aids such as magnifying glasses, telescopes, eye contacts, and glasses.
- Non-vision aids such as reading lamps, painted contacts, display devices, and bigger-fonts books.
- Electronic devices that might help blind individuals.

Mubser – مُبْصِر[8]

1. A 3D camera on the belt takes images of the environment and sends it to the pocket computer for processing.
2. The pocket computer performs its functions to navigate the patients.
3. The patient is navigated using audio on a Bluetooth headset or vibrations on vibration motors on the belt.



Services:

- Calculating the number of people in a room.
- Detecting obstacles.
- Notification of obstacles existence using built-in vibration motors inside a belt.

Comparison:

	A. Eye	مُبْصِر
Processing unit	Raspberry Pi	Pocket computer
i/p	Raspberry Pi camera	3D camera
o/p	Voice	Vibrations and voice
no. of objects	80 objects	3 objects
Main technology	Python software	Embedded system

Arx[9]

This device consists of two main parts:

- ArxWear Headset:
  - Bone conduction speakers.
  - Microphone.
  - USB-C connection with phone.
- ArxApp:
  - Must be installed.
  - Detects and reads text.
  - Cloud text-to-speech.
  - Detects and remembers faces.
  - Detects emotions.
  - Detects objects.
  - Describe scenes.
  - Used with voice and/or touchscreen.

Comparison:

	A. Eye	ARx
Dependency	Standalone device	Depends on phone
Battery	Long battery	7 hours battery
Connectivity	Offline device	Requires internet
Earphones	External earphone	Bone conduction speakers

OrCam[10]

OrCam has two main products:

- OrCam MyEye. Features:
  - Read text.
  - Recognize faces.
  - Identify products.
  - Recognize barcodes.
  - Recognize money.
  - Identify colors.

- OrCam Read. Features:
  - Full page capture.
  - Laser guidance.
  - Read text.
  - Bright LED light.
  - Smart reading.
  - Point and click.

2.4.3 Project Success Rate

Success rate for wearable visual assistants have been in continuous increase for the last decade ranging from old technology, of using simple sensors and vibrators which is hardware focused, to using new and smart technologies like computer vision and artificial intelligence software embedded in their respective hardware components or installed on third-party systems like smartphones.

2.4.4 Success Measurement

- Battery: device needs no more than one full charge per day.
- High performance: close to zero system failure downtime.
- Response time: below 1 second for usual use and bellow 200ms for danger alerts.
- Cost: the final product does not exceed 5,000 L.E to be suitable for the Egyptian market for clients with above-average income.
- Average user rating of more than 70% after the mass production stage.

2.5 Economic Feasibility

2.5.1 Cost-Benefit Analysis

Costs	
Software Costs	
Open source python libraries	0EGP
Hardware Costs	
Raspberry Pi 4 8GB Model B with 1.5GHz 64-bit quad-core CPU (8GB RAM)	1880EGP (Full Kit)[11]
32GB Samsung EVO+ Micro SD Card (Class 10) Pre-loaded with NOOBS	
USB MicroSD Card Reader	
CanaKit Premium High-Gloss Raspberry Pi 4 Case with Integrated Fan Mount	
CanaKit Low Noise Bearing System Fan	
CanaKit 3.5A USB-C Raspberry Pi 4 Power Supply with Noise Filter	
Set of Heat Sinks	
Micro HDMI to HDMI Cable - 6 foot (Supports up to 4K 60p)	
CanaKit USB-C PiSwitch (On/Off Power Switch for Raspberry Pi 4)	
Raspberry Pi Camera Board – V2 (8MP)	800EGP[12]
Xiaomi Powerbank 10000mA	400EGP[13]
Earphones	External
Total Cost	**3080EGP**
Benefits	
Providing cheaper alternative to similar solutions	
Providing mobility to the user without affecting his/her movement with mechanical parts	

2.5.2 Project Lifecycle

We are working on our project using sprints in Agile principles but with every sprint taking one month to complete due to our busy schedule throughout the year. The timeline was as follows:

- August 2021: Project feasibility study
- September 2021:
- October:
- November

This should be a full-lifetime product that needs minimum maintenance since no software system updates/upgrades are required. Only hardware fixes might be needed and such problems should occur rarely. Done in 12 months.

Chapter 5: Object Detection

5.1 Introduction

In this section, we are going to talk about object detection. It is considered as a computer vision technique, used for identifying and localizing objects in an image or video. This allows us to identify, count, and track the precise locations of the objects while accurately labeling them [14].

Object detection approaches can be broken down into machine learning-based approaches, and deep learning-based approaches. The later is the most popular and common for making an object detection models [14].

There are many different deep learning-based algorithms used and the one we are going to use in our project is called You Only Look Once (YOLO) which is considered one of the fastest algorithms out there.

5.2 How it works?

The basic structure of deep learning-based object detection models typically has 2 parts [14]:

- Encoder: takes an image as input, then this image goes through a series of blocks and layers for statistical features extraction to locate and label the objects.
- Decoder: takes the output from the encoder and predicts the bounding boxes and labels for each object.

The simplest decoder is a pure regressor. It is used to predict the location and size of each bounding box by providing the X and Y coordinates of the object (center of the object) and width and height of the object. But the problem with pure regressor is that the required number of objects must be specified ahead of time in each image [14].

The solution for this problem is a region proposal network which is an extension of regressor approach. In this decoder, the model proposes regions of an image where it believes an object might reside. The pixels belonging to these regions are then fed into a classification subnetwork to determine a label (or reject the proposal). It then runs the pixels containing those regions through a classification network. The benefit of this method is a more accurate, flexible model that can propose arbitrary numbers of regions that may contain a bounding box. The added accuracy, though, comes at the cost of computational efficiency [14]. Figure 1 below shows a simple diagram of region proposal network.

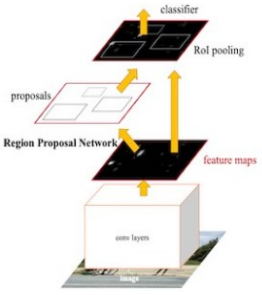


Fig.1 (simple diagram of region proposal network)

5.3 Deep learning-based Model Architectures

There are various architectures made for deep learning-based object detection and here are some of them:

R-CNN, Faster R-CNN, Mask R-CNN

These some popular models that belong to the R-CNN family (an abbreviation for Region Convolutional Neural Network). These architectures use the region proposal network approach discussed above. These models have become more accurate and more computationally efficient over the years, developed by researchers at Facebook [14].

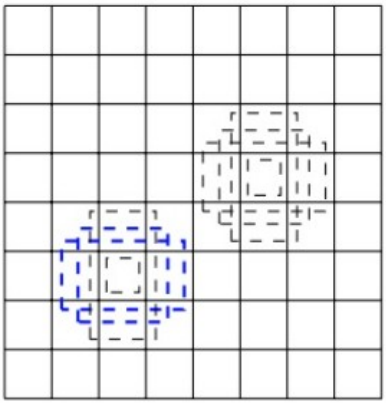
YOLO, MobileNet + SSD, SqueezeDet

These models belong to the single shot detector (SSD) family.

Single shot detectors (SSDs) seek a middle ground. Rather than using a subnetwork to propose regions, SSDs rely on a set of predetermined regions.

A grid of anchor points is laid over the input image, and at each anchor point, boxes of multiple shapes and sizes serve as regions, see fig.2 below. For each box at each anchor point, the model outputs a prediction of whether an object exists within the region and modifications to the box's location and size to make it fit the object more closely.

Because there are multiple boxes at each anchor point and anchor points may be close together, SSDs produce many potential detections that overlap. Post-processing must be applied to SSD outputs in order to prune away most of these predictions and pick the best one. The most



popular post-processing technique is known as non-maximum suppression. SSDs make great choices for models destined for mobile or embedded devices [14].

Fig.2

5.4 Adopted algorithm :YOLO

In this project the YOLO algorithm has been used. It uses CNN (Convolutional Neural Networks) to detect objects in real-time. It only requires a single forward propagation through the neural network to detect objects. This means that the predictions made in an image is done in a single algorithm run. That's why it's named "You Only Look Once" (YOLO). YOLO algorithm has various variants: YOLO, YOLOv2, YOLOv3, YOLOv4, …etc [15].

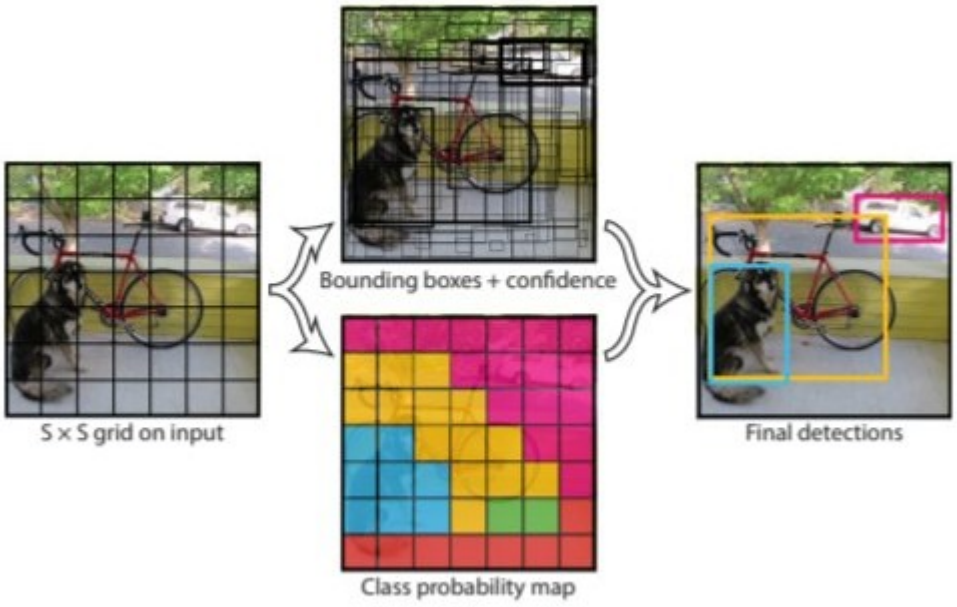
Why YOLO? [15]

1. Speed: this algorithm improves the speed of detection and can predict objects in real-time.
2. High accuracy: it's a predictive technique which gives an accurate result with minimal background errors.
3. Learning capabilities: it has excellent capabilities that enables it to learn representations of objects and apply them in object detection.

How YOLO works? [15]

The following figure demonstrates how the YOLO algorithm works.

Fig.3



YOLO works using the following three techniques:

- Residual blocks.
- Bounding box regression.

- Intersection Over Union (IOU).

Residual blocks:

The image is divided into various S x S grids. All cells in the grid have equal dimension and each of these cells is responsible for detecting the object that appear within them. See Fig.4 below.

Fig.4 residual blooks: dividing the image into S x S grid

Bounding box regression:

A bounding box is an outline that highlights an object in an image. Every bounding box has the following attributes: Width (bw), Height (bh), Class (c), center (bx, by), probability (pc). See Fig.5 below.

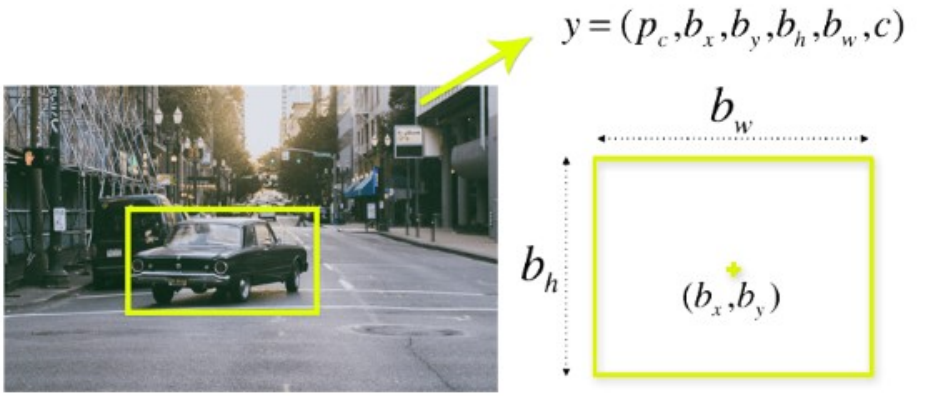
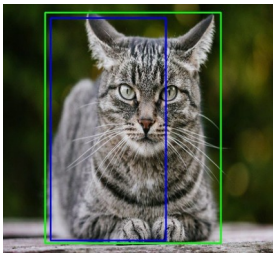


Fig.5

Intersection Over Union (IOU):

Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly.

Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU equals to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box. See the Fig.6 below.



In Fig.5, there are two bounding boxes, one in green and the other one in blue. The blue box is the predicted box while the green box is the real box. YOLO ensures that the two bounding boxes are equal.

5.5 Flowchart

Chapter 6 : Face Recognition[16]

6.1 Introduction

You have probably noticed how common face recognition is among the softwares of big tech companies such as Facebook, Snapchat, Twitter, or any social media software out there. It is also used in security systems to control users access. An example of that can be closer than you think; if you have a modern smart phone, you might be using face recognition technology to be allowed to open your phone.

Many other uses can be mentioned but as far as we are concerned in our project, we want to help visually impaired people to be able to detect the people they know as any other person would do. We do this by providing our A. Eye device with a face recognition model (see Appendix G for code) that keeps running and recognizing faces in real-time. This way is the most natural way to help our users feel close to other people that don't have the same circumstances.

The technology of face recognition has gone through several updates and many researches have been made to improve its accuracy and speed over the last decades. We didn't need to search much to find the best face recognition software to use in our A. Eye. It appears that the face_recognition library[30] made by Adam Geitgey[29] has all our needs. It has the speed needed for the real-time nature of the system, accuracy that is high enough to not get mistaken between faces, and simple syntax that helps getting the work done in the fastest way possible.

In this chapter, we'll look at how the face_recognition library works behind the scenes by utilizing OpenFace and dlib libraries written in Python. Face recognition is really a series of several related problems:

1. First, look at a picture and find all the faces in it.
2. Second, focus on each face and be able to understand that even if a face is turned in a weird direction or in bad lighting, it is still the same person.
3. Third, be able to pick out unique features of the face that you can use to tell it apart from other people; like how big the eyes are, how long the face is, etc.
4. Finally, compare the unique features of that face to all the people you already know to determine the person's name.

All of those problems can be solved by choosing one machine learning/deep learning algorithm, feeding in data, and getting the result. As a human, your brain is wired to do all of this automatically and instantly. In fact, humans are *too good* at recognizing faces and end up seeing faces in everyday objects.

We need to build a *pipeline* where we solve each step of face recognition separately and pass the result of the current step to the next step. In other words, we will chain together several machine learning algorithms as explained in the next sections.

6.2 Step 1: Face Detection

Face detection is a great feature that is commonly used for cameras. When the camera can automatically pick out faces, it can make sure that all the faces are in focus before it takes the picture. But we'll use it for a different purpose; finding the areas of the image we want to pass on to the next step in our pipeline.

Face detection went mainstream in the early 2000's when Paul Viola and Michael Jones invented a way to detect faces[17] that was fast enough to run on cheap cameras. However, much more reliable solutions exist now. We're going to use a method invented in 2005 called Histogram of Oriented Gradients[18] — or just ***HOG*** for short.

Histogram of Oriented Gradients (HOG)

To find faces in an image, we'll start by making our image black and white because we don't need color data to find faces. Then we'll look at every single pixel in our image one at a time. For every single pixel, we want to look at the pixels that directly surround it as shown in Figure 6-2.

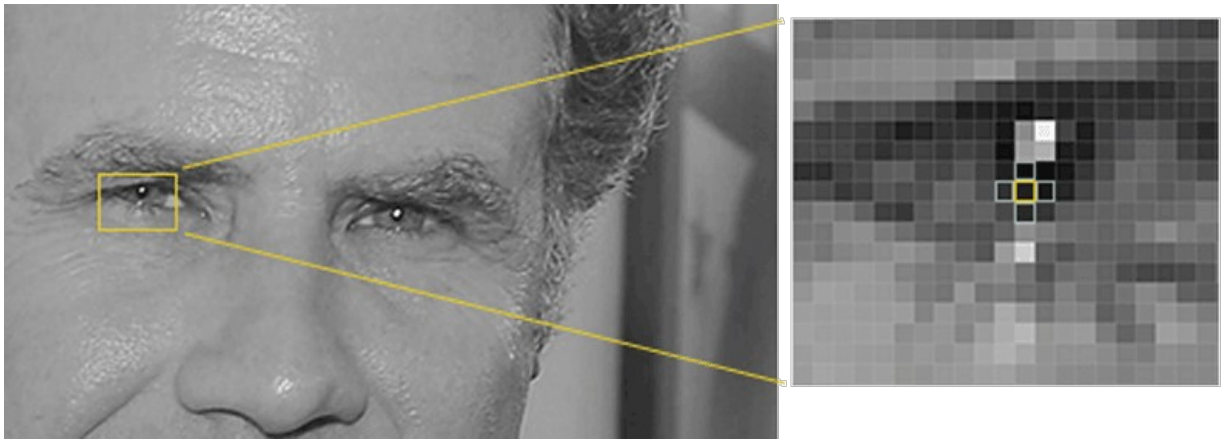


Figure 6-2: Checking surrounding pixels for every pixel

Then we need to figure out how dark the current pixel is compared to the pixels directly surrounding it. Afterwards, we want to draw an arrow showing in which direction the image is getting darker as shown in Figure 6-3.

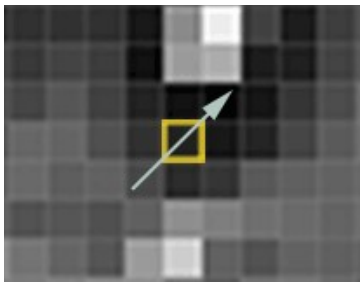


Figure 6-3: Image is getting darker towards the upper right

When repeating that process for every single pixel in the image, we end up with every pixel being replaced by an arrow. These arrows are called *gradients* and they show the flow from light to dark across the entire image. This is done on several levels but Figure 6-4 shows the final form of it.

This might seem like a random thing to do, but there's a really good reason for replacing the pixels with gradients. If we analyze pixels directly, really dark images and really light images of the same person will have totally different pixel values. But by only considering the *direction* at which the brightness changes, both really dark images and really bright images will end up with the same exact representation. That makes the problem a lot easier to solve.

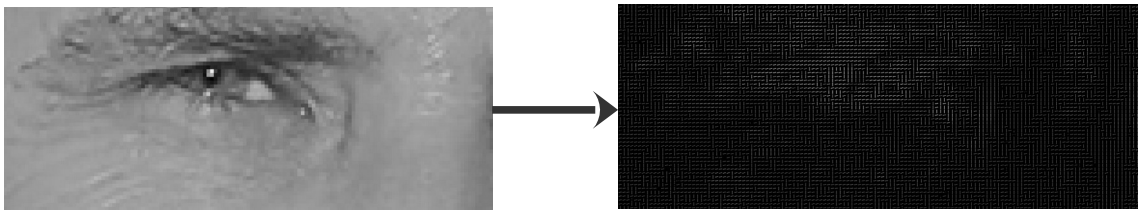


Figure 6-4: Pixels converted to gradients

But saving the gradient for every single pixel gives us way too much detail. We end focusing too much on small pixels while the real concern is the picture as a whole. It would be better if we could just see the basic flow of lightness/darkness at a higher level so we could see the basic pattern of the image.

To do this, we break up the image into small squares of 16x16 pixels each. In each square, we'll count up how many gradients point in each major direction (how many point up, point up-right, point right, etc…). Then we'll replace that square in the image with the arrow directions that were the strongest. The end result is we turn the original image into a very simple representation that captures the basic structure of a face in a simple way as shown in Figure 6-5.

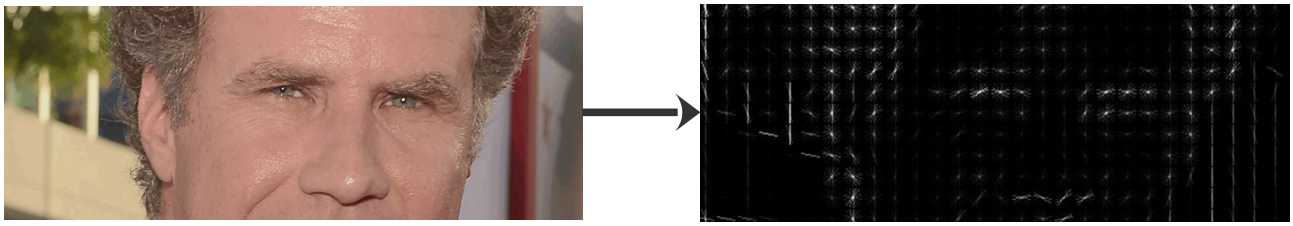


Figure 6-5: Original image turned into a HOG representation

To find faces in this HOG image, all we have to do is find the part of our image that looks the most similar to a known HOG pattern that was extracted from a bunch of other training faces as shown in Figure 6-6. Using this technique, we can now easily find faces in any image.

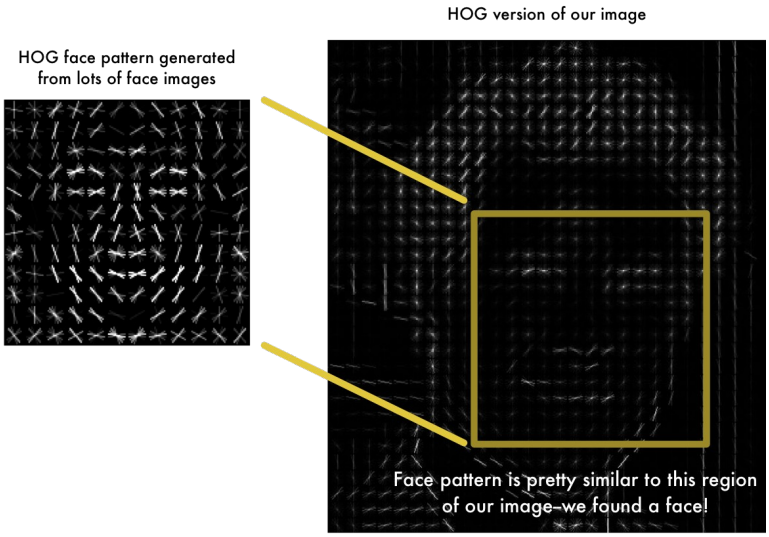


Figure 6-6: HOG patterns compared to each other

6.3 Step 2: Posing and Projecting Faces

After isolating the faces in our image, we now have to deal with another problem; faces turned to different directions look totally different to a computer. Humans can easily recognize if two images taken at different angles are of a person or not, but computers would see these pictures as two completely different people.

To account for this, we will try to warp each picture so that the eyes and lips are always in the sample place in the image. This will make it a lot easier for us to compare faces in the next steps. To do this, we are going to use an algorithm called ***face landmark estimation***. There are lots of ways to do this, but we are going to use an approach invented in 2014 by Vahid Kazemi and Josephine Sullivan[20].

Face Landmark Estimation

The basic idea is that we will come up with 68 specific points (called *landmarks*) that exist on every face; the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. All the 68 landmarks are represented in Figure 6-7. Then we will train a machine learning algorithm to be able to find these 68 specific points on any face. You might have noticed by now that using this technique makes it easy to computationally recognize faces represented in different angles as the same person without much processing.

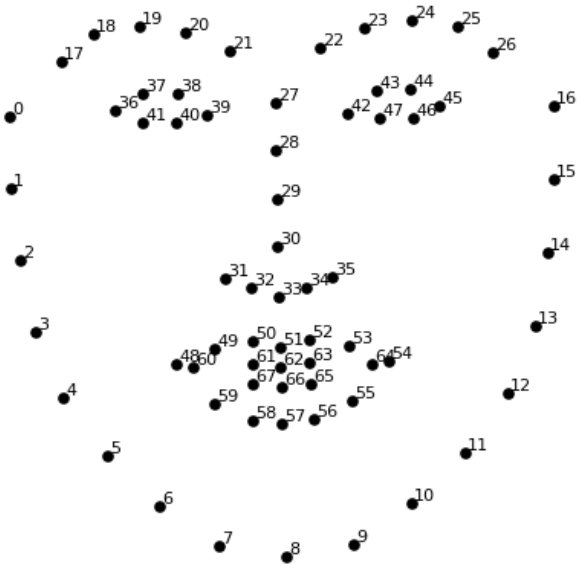


Figure 6-7: 68 face landmarks

Now that we know were the eyes and mouth are, we'll simply rotate, scale and shear[21] the image so that the eyes and mouth are centered as best as possible. We are only going to use basic image transformations like rotation and scale that preserve parallel lines (called affine transformations[22]). This is illustrated below in Figure 6-8. Now no matter how the face is turned, we are able to center the eyes and mouth in roughly the same position as in the image. This will make our next step a lot more accurate.

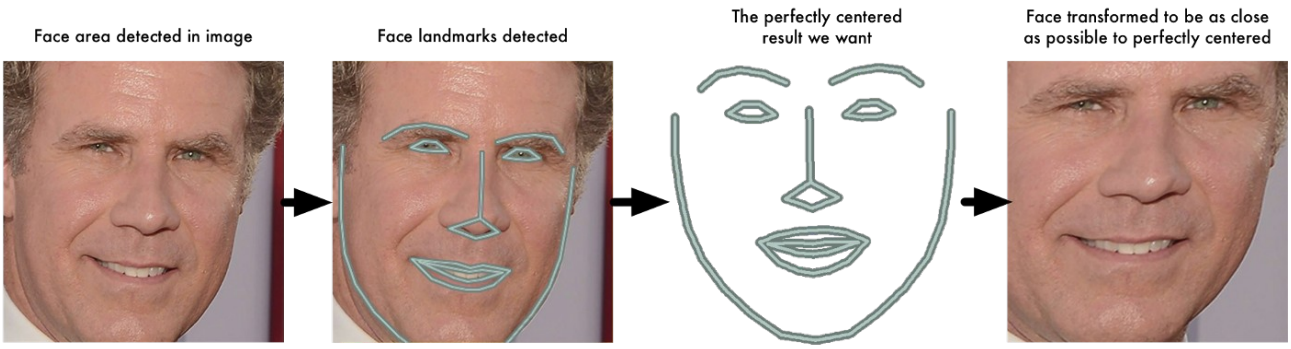


Figure 6-8: Face landmarks detection and face transformation

6.4 Step 3: Encoding Faces

At this step, we need to know how to tell faces apart. The simplest approach to face recognition is to directly compare the unknown face we found in Step 2 with all the pictures we have of people that have already been tagged. When we find a previously tagged face that looks very similar to our unknown face, it must be the same person.

This seems like a reasonably good idea at first glance, but there's actually a huge problem with that approach. A person walking around with the A. Eye and moving his/her head frequently can generate tens of photose per second that can contain multiple faces in each photo. The algorithm can't possibly loop through every previously tagged face to compare it to every newly uploaded picture. That would take way too long. It needs to be able to recognize faces in milliseconds, not minutes.

What we need is a way to extract a few basic measurements from each face. Then we could measure our unknown face the same way and find the known face with the closest measurements. For example, we might measure the size of each ear, the spacing between the eyes, the length of the nose, etc.

Now you might ask: which measurements should we collect from each face to build our known face database? It turns out that the measurements that seem obvious to us humans (like eye color) don't really make sense to a computer looking at individual pixels in an image. Researchers have discovered that the most accurate approach is to let the computer figure out the measurements to collect itself. Deep learning does a better job than humans at figuring out which parts of a face are important to measure.

The solution is to train a Deep Convolutional Neural Network (more on that can be found at reference point [19]). But instead of training the network to recognize pictures objects like we did last time, we are going to train it to generate 128 measurements for each face. The training process works by looking at 3 face images at a time:

1. Load a training face image of a known person.
2. Load another picture of the same known person.
3. Load a picture of a totally different person.

Then the algorithm looks at the measurements it is currently generating for each of those three images. It then tweaks the neural network slightly so that it makes sure the measurements it generates for #1 and #2 are slightly closer while making sure the measurements for #2 and #3 are slightly further apart. This is illustrated in Figure 6-9.

After repeating this step millions of times for millions of images of thousands of different people, the neural network learns to reliably generate 128 measurements for each person. Any ten different pictures of the same person should give roughly the same measurements.

Machine learning people call the 128 measurements of each face an embedding. The idea of reducing complicated raw data like a picture into a list of computer-generated numbers comes up a lot in machine learning (especially in language translation). The exact approach for faces we are using was invented in 2015 by researchers at Google[23] but many similar approaches exist.

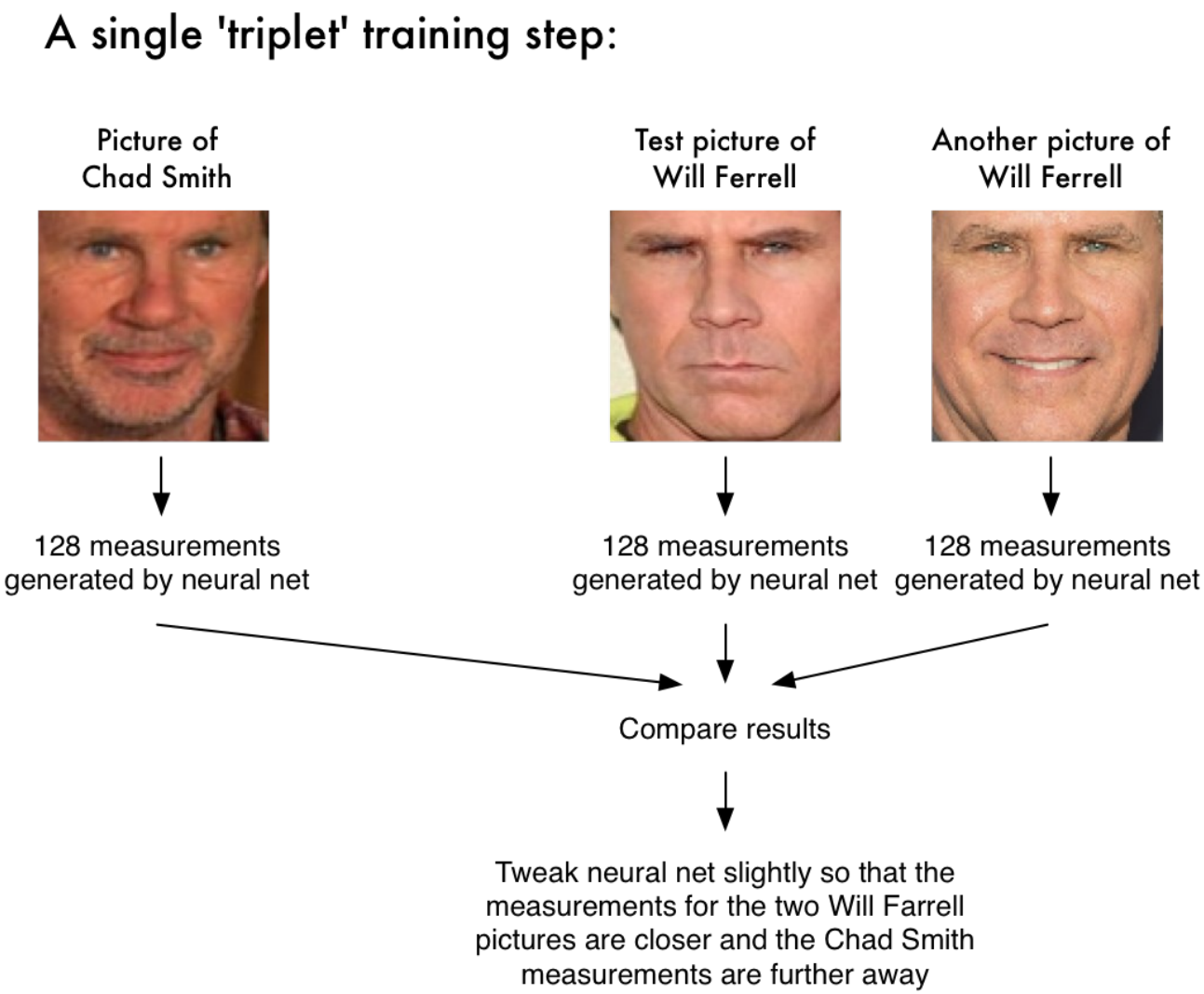


Figure 6-9: Triplet training step for CNN

This process of training a convolutional neural network to output face embeddings requires a lot of data and computer power. Even with an expensive NVidia Telsa video card[24], it takes about 24 hours[25] of continuous training to get good accuracy. But once the network has been trained, it can generate measurements for any face, even ones it has never seen before. So this step only needs to be done once. Fortunately, this has been already taken care of by people at OpenFace[26] already did this and they published several trained networks[27] which we can directly use thankfully. So all we need to do ourselves is run our face images through their pre-trained network to get the 128 measurements for each face. The measurements for our test image is shown in Figure 6-10.

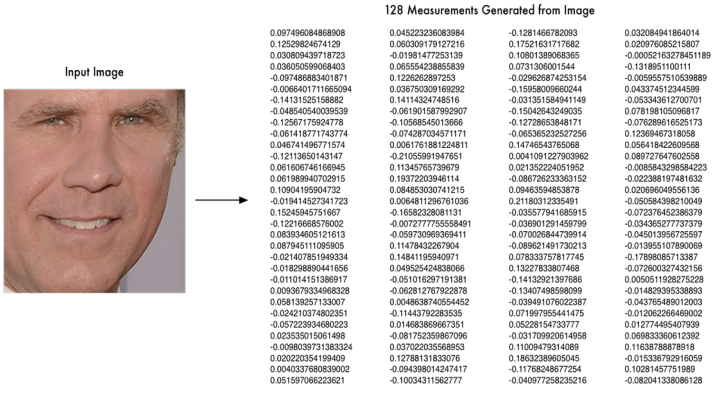


Figure 6-10: Example of the 128 measurements generated from the OpenFace network on a test image

So what parts of the face are these 128 numbers measuring exactly? It turns out that we have no idea. It doesn't really matter to us. All that we care for is that the network generates nearly the same numbers when looking at two different pictures of the same person.

6.5 Step 4: Finding the Person's Name from the Encoding

This last step is actually the easiest step in the whole process. All we have to do is find the person in our database of known people who has the closest measurements to our test image. You can do that by using any basic machine learning classification algorithm. No deep learning tricks are needed. We'll use a simple linear SVM classifier[28], but lots of classification algorithms could work.

All we need to do is train a classifier that can take in the measurements from a new test image and tells which known person is the closest match. Running this classifier takes milliseconds. The result of the classifier is the name of the person.

If you want to test this out, here's the steps to follow:

1. Train a classifier with the embeddings of about 20 pictures of each person you want your system to know.
2. Run the classifier on either testing pictures saved in a dataset or on every frame of a video containing the face of the person you want to recognize. It depends on your needs. In our project, we make the testing on a live video frames the are taken by the Raspberry Pi camera.

6.6 Summary

1. Encode a picture using the HOG algorithm to create a simplified version of the image. Using this simplified image, find the part of the image that most looks like a generic HOG encoding of a face.
2. Figure out the pose of the face by finding the main landmarks in the face. Once we find those landmarks, use them to warp the image so that the eyes and mouth are centered.
3. Pass the centered face image through a neural network that knows how to measure features of the face. Save those 128 measurements.
4. Looking at all the faces we've measured in the past, see which person has the closest measurements to our face's measurements. That's our match.

If you want to follow these instructions, you can read more about the face_recognition library written by Adam Geitgey[29]. It greatly simplifies the work needed to recognize faces and we used it to make our face recognition model in A. Eye (see code at Appendix G). However, you can still follow those steps using OpenFace and dlib by following the instructions written in the end of the article referenced at point [19].

References

[1] "Vision Impairment and blindness," World Health Organization. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/blindness-and-visual-impairment. [Accessed: 01-Aug-2021].

[2] "Eastern Mediterranean Region," World Health Organization. [Online]. Available: http://www.emro.who.int/egy/egypt-events/world-sight-day.html. [Accessed: 01-Aug-2021].

[3] "Face Blindness," NHS choices. [Online]. Available: https://www.nhs.uk/conditions/face-blindness/. [Accessed: 01-Aug-2021].

[4] S. Bradt, "Face-blindness disorder may not be so rare", 1-June-2006. [Online]. Available: news.harvard.edu/gazette/story/2006/06/face-blindness-disorder-may-not-be-so-rare/. [Accessed: 01-Aug-2021].

[5] K. Samuelson, "Prosopagnosia and face blindness: Why you may be face blind," Time, 14-Jul-2017. [Online]. Available: https://time.com/4838661/prosopagnosia-face-blindness/. [Accessed: 01-Aug-2021].

[6] "Dyslexia," NHS choices. [Online]. Available: https://www.nhs.uk/conditions/dyslexia/. [Accessed: 02-Aug-2021].

[7] "Sector specialisms," NCFE. [Online]. Available: https://www.cache.org.uk/news-media/dyslexia-the-facts. [Accessed: 02-Aug-2021].

[8] *بوابة الأهرام*. [Online]. Available: https://gate.ahram.org.eg/News/417438.aspx. [Accessed: 02-Aug-2021].

[9] *ARxVision*. [Online]. Available: Arx.vision. [Accessed: 02-Aug-2021].

[10] "StackPath", [Online]. Available: www.orcam.com/en/myeye2. [Accessed: 02-Aug-2021].

[11] "CanaKit Raspberry 8GB Starter Kit", [Online]. Available: www.amazon.com/CanaKit-Raspberry-8GB-Starter-Kit/dp/B08956GVXN/ref=sr_1_3. [Accessed: 03-Aug-2021].

[12] "raspberry pi camera board v2 8mp", [Online]. Available: makerselectronics.com/product/raspberry-pi-camera-board-v2-8mp. [Accessed: 03-Aug-2021].

[13] "Power Bank", [Online]. Available: egypt.souq.com/eg-ar/powerbank/s/. [Accessed: 03-Aug-2021].

[14] "Object Detection Guide", *Fritz AI*. [Online]. Available: www.fritz.ai/object-detection. [Accessed: 04-July-2022].

[15] "Introduction to YOLO Algorithm for Object Detection", *Section*. [Online]. Available: www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection. [Accessed: 04-July-2022].

[16] A. Geitgey, "Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning", *Behave Your Business*. 24-July-2016. [Online]. Available: medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78. [Accessed: 10-July-2022].

[17] "ViolaJones object detection framework", *Wikipedia*. 12-Dec.-2007. [Online]. Available: en.wikipedia.org/wiki/Viola-Jones_object_detection_framework. [Accessed: 10-July-2022].

[18] https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

[19] https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721

[20] http://www.csc.kth.se/~vahidk/papers/KazemiCVPR14.pdf

[21] "Shear mapping", *Wikipedia*. 9-July-2004. [Online]. Available: en.wikipedia.org/wiki/Shear_mapping. [Accessed: 10-July-2022].

[22] "Affine transformation", *Wikipedia*. 25-Feb.-2002. [Online]. Available: en.wikipedia.org/wiki/Affine_transformation. [Accessed: 10-July-2022].

[23] http://www.cv-foundation.org/openaccess/content_cvpr_2015/app/1A_089.pdf
[24] http://www.nvidia.com/object/tesla-supercomputing-solutions.html
[25] https://twitter.com/brandondamos/status/757959518433243136
[26] https://cmusatyalab.github.io/openface/
[27] https://github.com/cmusatyalab/openface/tree/master/models/openface
[28] https://en.wikipedia.org/wiki/Support_vector_machine
[29] https://github.com/ageitgey/
[30] https://github.com/ageitgey/face_recognition#face-recognition
[31]

Appendices

Appendix A: Questionnaire

1. In your point of view do you think the device is useful?
2. Do you know any company that produce a similar device?
3. What is the expected price for such a device?
4. Would you recommend the device if it is already produced and available in the market?
5. Do you want to add new features to the device?
6. Do you know any institution that would provide the device for free for the incapable patients?
7. In case the device is made by our team would you allow us to test it on your patients?

Appendix G: Face Recognition Model

put the face recognition code after it is finished