UMONS
Université de Mons

Faculté Polytechnique

Design of a gesture detection system at real time

Master Thesis

Submitted by Roger Villanueva Sert

In fulfillment of the requirements

For the Master's Degree in Electrical Engineering

POLYTECH MONS

Under the direction of:

Professor Stéphane Dupont

August 2017

« You have to have a big vision and take very small steps to get there. You have to be humble as you execute but visionary and gigantic in terms of your aspiration. In the Internet industry, it's not about grand innovation, it's about a lot of little innovations: every day, every week, every month, making something a little bit better »

(Calacanis, 2009)

POLYTECH MONS

Master's Degree in Electrical Engineering

Final Thesis Summary

AUTHOR	- Roger Villanueva Sert
WORK TITLE (English and French)	- Design of a gesture detection system at real time - Conception d'un système de detéction de gestes en temps réel

KEY WORDS (English and French)

Convolutional neural network, CNN, convnets, gesture detection, face detection, background subtraction, computer vision.
Réseau neuronal convolutif, CNN, convnets, détection de geste, détection de visage, soustraction de fond, vision par ordinateur.

SUMMARY (350 words, English and French)

The project I have realized consist in developing a gesture detection system to work at real time situations. In particular, it has the aim to detect a wink of an eye and activate a flag when that happens.

There are some actual projects and systems that already do that, but they are focused on voice detection. This project follows the same principles but it uses an input of video instead of a sound.

The creation of the pipeline was made in different parts. First of all, a convolutional neural network was created to detect the gesture in a

sequence of images and it had to be trained to do so.

Secondly, a convolutional neural network for face detection was used as background subtraction, in order to select the main part of the image.

Finally, different methods of optimization were taken into account, so as to make the processing operations work faster.

A code was implemented to prove the background susbstraction of the image in order to reduce the processing time. Using this code, results were obtained about the accuracy and the processing time using Python. However we only obtained results from the part of background subtraction because the part of detecting the gesture was finally proposed as future work according to the lack of time and resources.

All in all, the results obtained were about the simplification of the image doing the background subtraction using a face detection method. We obtained that the time to detect the zone of the face took an average of 0.65 second.

Knowing that the system needs to take an image with the camera, do the background subtraction and process a convolutional neural network several times to detect a gesture, we deduce that the time that lasts the face detection makes that imposible.

It is needed to improve it more to make it work at real time.

PROFESSOR **-** Stéphane Dupont

PRESENTATION DATE **-** 21st of August, 2017.

ACKNOWLEDGMENTS

This work would not have been possible without the collaboration and assistance of the people that surrounded me, and helped me to grow and reflect during my final thesis report about computer vision and machine learning.

First of all, I would like to thank the University of Mons for giving me the opportunity to invest and develop a work Master Degree related to the Electrical Engineering field.

I would also like to thank my final thesis professor and tutor for the project, Stéphane Dupont, to encouraging me to learn, for his help in the approaches and for giving me the opportunity to learn about this process, but also for showing the benefits of machine learning methods.

A special thanks to all the members of the University of Mons that have offered me the facilities and materials required to implement this project. I have to thank my friends for supporting me during the practical part of my work. Since the very beginning, they showed to be very interested on this domain, and opened to learn about it. This experience helped me to know about the strengths and weakness of this approach and to see which are the level and methods used on this field nowadays.

Finally, to the *Universitat Politècnica de Barcelona* for giving me the chance to do my master thesis abroad.

To all of you, thank you.

Design of a gesture detection system at real time

INDEX

Design of a gesture detection system at real time

1. Introduction

On the epoch we live in, people search for comfort and facilities to make their current life activities, even if they do them occasionally. This has only been possible thanks to the technology improvements that have been made.

One example of this is the application of voice recognition methods on electronic machines and devices. The most famous applications are SIRI (from Apple Inc.), Ok Google (from Alphabet Inc.) and Cortana (from Microsoft Corporation) that work as personal assistant in smartphones. The way how they work is formed with a program running in a loop and when it hears a word, it recognizes it and activates a flag, so another part of the code starts running.

The finality of this project is to develop a very similar software on how Ok Google works. There is a programming code running in a loop and when it hears the words Ok Google, it activates a flag that starts to recognize the words that you say after that two words.

This project is focused on the idea to activate a system without sound, focusing on the image. The main idea is that a camera will be recording images at real time and when it recognizes a human gesture, the system will get activated.

It can mainly be used in places where the sound is too loud, when the camera is far from the person that activates it or if the person is not able to speak.

The main challenges were that the system had to have a good detection rate with few false alarms and signal misses. Moreover, it need to have a low power consumption due the fact that it will be running all time in a low consumption CPU or GPU.

The report is structured in several sections. In objectives are defined the main goals of the project at the beginning and the ones we had last days. State of the art is divided in different parts where on similar projects we speak about the existing projects from where we based our work. The software used section is the main computer programs and libraries used to code. In CNN theory we make a brief introduction about convolutional neural networks. On data sets used we explain which ones we used and why.

In the section of optimization, the different techniques to make the code faster (optimization) are explained. On approaches we detail the different steps we follow to develop the project. On explanation of the code is shown which is the model we used on our system.

In the section of results, we show the output of the code and its times and accuracy.

Finally, in conclusions and future work we explain our personal opinion and future modifications to improve the code even better.

2. General method

2.1. Problem definition

The realization of this project has changed while the work was taking form. For this reason, the several objectives we had at the beginning and the ones we decided and accomplished at the end are going to be presented. The final objectives have been adapted from the initial ones, so a lot of sub objectives and methods have been maintained.

Initial objectives

The definition of this project has been taken from a list of proposed projects of the University of Mons (2017). The provided statement was:

"The second topic is focused on analysing and understanding the motion/gestures of an individual to detect specific gestures that are to be used to activate a system. This would be similar to the idea of vocal activation (like the "hey Siri" used to activate the Siri vocal dialog system on iphones) but here motion activation. Imagine it like a specific gesture that you do to start "talking" to you TV. It should be very robust and cheap in terms of computing as it would run all the time, waiting for a sign from the user".

To do that I decided to take as an input a sequence of time-continued frames and to use a facial gesture (wink an eye). The reason why the facial gesture was chosen was because there is a lot of methods on the field of face detection, so it was easier than doing it with hands gestures instead.

At first, it was necessary to define the methods to accomplish it. The general problems we found were:

- A way to make the system identify the motion/gesture of different people in different places, with different duration and with different characteristics (duration, rotation, perspective, size, etc). It was decided to use a convolutional neural network to do that.

- A way of reducing the computational time of processing the images to detect the gesture. There were several methods found and used to do that which are explained in the following sections.

However, there was the biggest problem that consisted in training a convolutional neural network in a face gesture. There was the need of a dataset of images to detect a specific gesture but we could not succeed on finding one, so it was not possible to train the system. For that reason, it was decided to avoid the gesture detection and focus on what it was obtained om that moment, that was the face detection.

Final objectives

Once it was clear that the part of face gesture detection could not be done, it was decided to focus on the actual work and to use it to solve a problem. At that moment we had a face detector, so it was decided to make a face tracker to be faster and cheaper in terms of computing.

2.2. Similar projects

In order to design our system, the main projects and methods used to take ideas about it are:

Viola Jones

Viola Jones face detection algorithm is one of the fastest tools to do face detection. The reason is that has a high accuracy and false positives rates compared to other algorithms. It was also one of the first to achieve real-time face detection at 15 frames per second so it is very known on this field (Patil, 2014; Smaridge, 2013).

The algorithm is based in four key features (Ramsrigouthamg, 2012; Wang, 2014).

- Integral image: it is the added area table that is necessary for fast calculations.
- Haar-like features: simple rectangular features generated randomly. They are used to form classifiers.
- Adaboost: algorithm that eliminates all the redundant features.
- Cascade filter: it discards all negative windows as early as possible to reduce the time on future operations. So at the end of the filter there will be two groups: the positive matches that are sent along to the next feature and the negative matches, that are rejected and exit the evaluations.

This algorithm was studied in order to do the part of background subtraction and how it used the cascade filters. As it is a general method there are a lot of existing projects adapting this method.

A convolution neural network cascade for face detection

This project is based on face detection with a convolutional neural network using cascade. As it does the same of one sub-objective we defined, it has been used on the project. Later on, it is explained on the report (Li, Lin, Shen, Brandt & Hua, 2015).

Max-Pooling convolutional neural networks for vision-based hand gesture recognition

This project developed by members of the university of Lugano (Switzerland) and the university Tenaga Nasional Putrajaya (Malaysia) explains a system of classifying hand gestures at real time. It interested us by the fact they use a convolutional neural network to obtain a class (there are 6 possible gestures). Another interesting things is how it works at real time and how it trains its neural network (Nagi, Ducatelle, Di Caro, Ciresan, Meier, giusti, Nagi F., Schmidhuber & Gambardella, 2011).

2.3. CNN theory

Convolutional neural networks, conv nets or CNN nets are a type of neural networks that consist on a set of layers where you take an image and pass it through a series of layers (convolutional, nonlinear, pooling and fully connected layers) to get an output. The output can be a single class or a probability of belonging to one that describes the image in the best way (Deshpande, 2016, Jefkine, 2016 & Kuwajima, 2014).

If you feed them with a bunch of faces, they will recognize lines and edges because they are a multi-layer neural network. At each layer it will recognize more complex features.

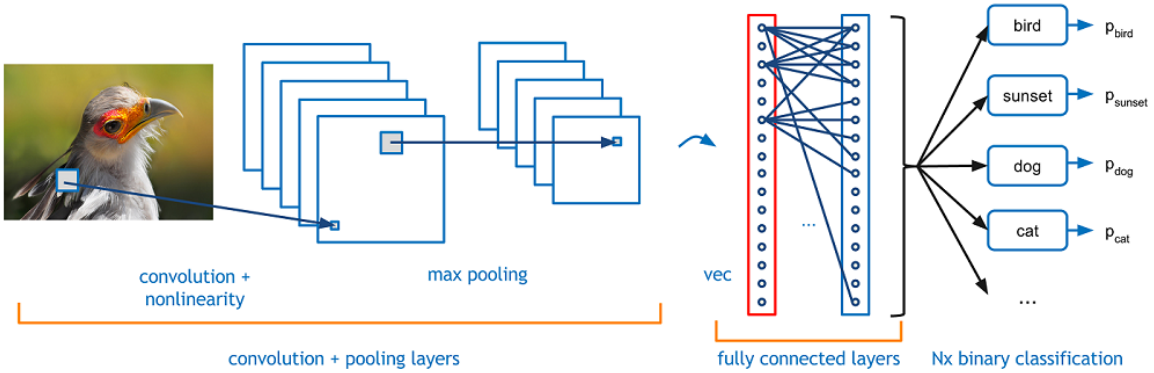

Figure 1: CNN algorithm [Source : Rohrer, B. (2016) How Convolutional Networks work]

The different types of layers that compose a CNN are (Deshpande, 2016, Jefkine, 2016 & Kuwajima, 2014):

- **Pooling**: this is how we shrink the image stack. We start with a window of a determined size (usually 2 by 2 or 3 by 3 pixels) and we move it in strides across each of the filtered images. For each window we take the maximum value (if it is used maximum pooling). It can happen that when we are doing the pooling, not all the pixels are represented in the last column. In order to solve this problem, several techniques can be used. For example, picking the maximum value or doing the average.

What we have once we have finished is a similar pattern but smaller. This has a lot of sense for images with lots of pixels, so the following operations will be faster to do. Another thing it does is that pooling does not care where the maximum value occurs, so that means that has a margin of detecting the big values, that is the same than saying that it will detect patterns that are a little moved, rotated, etc. Pooling is done to all the filtered images and in every case we obtain reduced images maintaining the patterns.

- **Normalization**: it is used to reduce the math operations applying non-linearity functions. There are several of them sigmoid, hyperbolic tangent, ReLu, step function, etc.

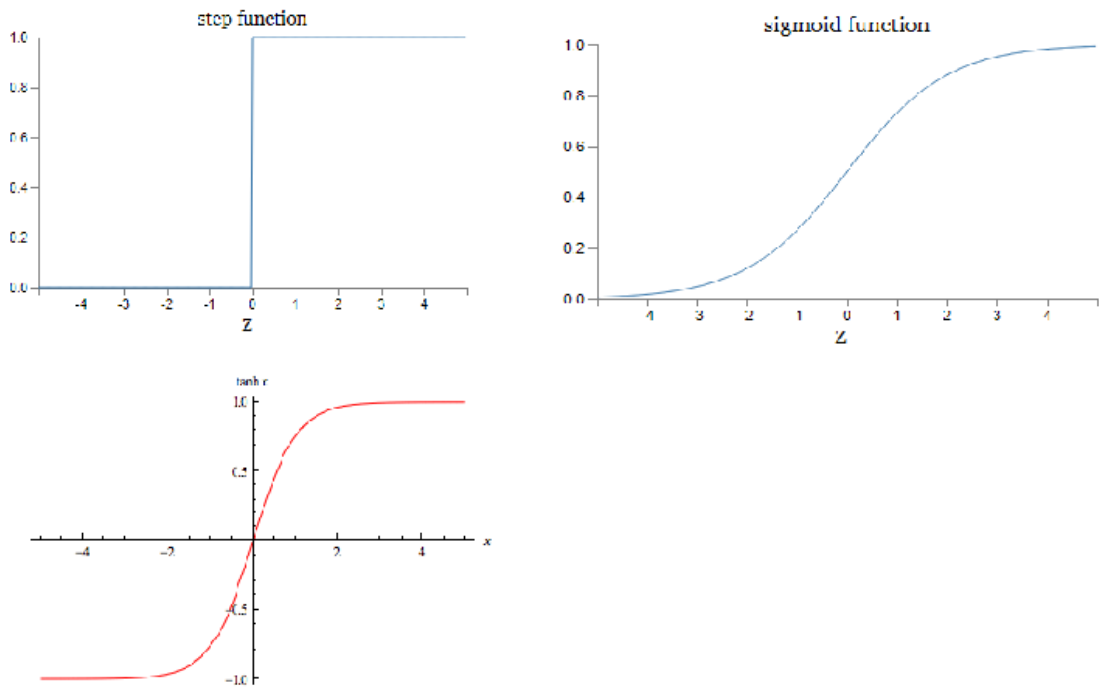

Figure 2: Non-linear functions [Source : Michael A. Nielsen (2015) Neural Networks and Deep Learning, Determination Press]

ReLu is used to change all the negative values to zero. This is done to all the images and it is obtained the same images with no negative values. The layer where it is done is called rectified linear unit layer (ReLU).

- **Fully connected layer:** it is the last step of a CNN. It consists that every value gets a vote. The outputs of the layers are resized as an array. Each value will have high values when it has the same patterns that the training image (depending on a weight). So at every case we will obtain different values that, depending if their values are closer to the ones of the training it will determine if the image belongs to a determined class.

All these layers are used stacked on to the other, so the output of one layer becomes the input of another. We can design the numbers of layers we want and their order, so in every convolutional layer the image will be filtered and all the neural network will reduce its dimension.

The function of each layer is to reduce the image size and obtain and maintain some characteristic features. So at the end of the first convolutional layer the features are going to be straight lines and curves, but as the number of convolutional layers increases, each one is going to detect more complex features.

On the figure 1 we can observe an example of a convent to detect the object of the picture.

For example, if we want to read an image and detect if there is a drawing of an X or and O there are several steps to make. Let us imagine that the input is a two dimensional image where the X and the O can be shifted, bigger, rotated, thicker or thinner. The easiest way is to compare pixel by pixel, but if the image is modified as I said before it does not work. A CNN uses small parts or features of the images and compare them to some general features that have relations between them. They multiply each image pixel by the corresponding feature pixel. They add them up and divide by the total number of pixels in the feature. This is called filtering. The process of doing the filtering to all the image is called convolution. The process of doing several convolutions with a bunch of filters is called a convolutional layer. Using a convolution in one image become a stack of

filtered images. We get as many images back as we have filters (Rohrer, 2016).

The most interesting thing about CNN is the way they learn to determine if an image belongs to a class or to another. This is done through backpropagation, that determines the features in the convolutional layers and the voting weights in the fully connected layers (the importance of each value on how affects the class classification). The first values of the convolutional layer and the voting weights are set randomly. We obtain an output after the fully connected layer, we compare it to an expected one and we get an error. Using the gradient descend and lots of iterations in a loop helps us to reduce the output error adjusting the parameters said that can be modified automatically (Mazur, 2015).

However, there are some parameters called hyperparameters that the designer needs to make. These are (Deshpande, 2016, Jefkine, 2016 & Kuwajima, 2014):

- The number of layers we want, the kind of them and in which order.
- In convolutional layers: the number of features that should be used and its size.
- In the pooling layers: the window size and the window stride.
- In the fully connected layers: the number of hidden and intermediate layers and the number of neurons.

Convolutional neural networks have many applications. The main one is to identify and recognize objects or situations from an image. For example, a convent can be trained to identify the kind of animal that appears. All the animals of a species are different and there are a lot of races. The CNN is trained with a bunch of images of each animal and it learns the main features of each animal, so later on it is possibly to classify the kind of being of the picture. It can be also done with type or vehicles, furniture, houses, landscapes, etc.

2.4. Algorithm

Once we had clear what were our objectives we thought about the design of the algorithm to accomplish it. Using different existing problems to take ideas to make our project, the resulting methods to solve it were:

Initial approaches

The proposed solutions for the initial objectives can be presented in three different parts:

- **Part 1**, where we take time-continuous frames and we obtain an output saying if the gesture happens or not.
- **Part 2**, which is a way to reduce the image to the zone of the face, so for next steps it is not necessary to compute all the image again. On the pipeline it goes before the Part 1.
- **Part 3**, which is used to make modifications to the code to reduce the number of computations.
- **Part 4** that consists on making the method work in real time.

At first, it was tried to develop and run the code in the NVidia Jetson TX1 (detailed on the hardware used section) connected to a laptop in order to run the code in the NVidia GPU. However, there were several problems with compatibility between the NVidia graphic card and the laptop. The main was that after installing the drivers to use an external GPU the laptop graphic card became unused to reproduce even the laptop screen, which made impossible to code. For that reason, it was decided to use only one laptop with a powerful graphic card and the chosen was the MSI laptop of the hardware section to make it work.

Part 1

The desired project should read a set of continuous-time images and analyse them if a determined gesture takes place. We decided to use as a

gesture the blinking of an eye of a short duration for the reason that there is a lot of actual projects working on facial recognition and detection, and blink one eye is an easy gesture that people do not do unconsciously. To do that, it was thought to use a convolutional neural network that has an input of real time video (continued frames) of a duration of x seconds. The CNN will identify if the gesture is done and then will exit an output saying if it happens or not.

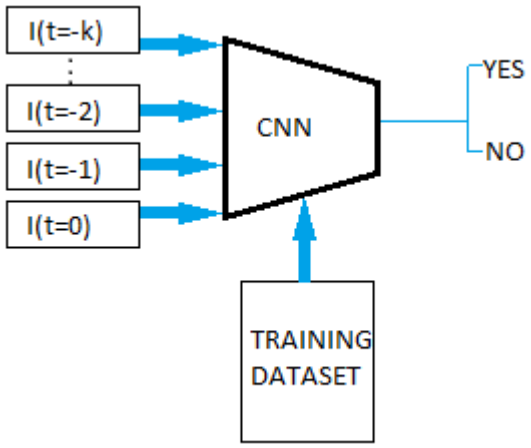

Figure 3: CNN training dataset. [Own Source]

The number of inputs of the CNN depends on how we want to last the blink of an eye. With 1-2 seconds it would be perfect. It is also necessary to know the duration of the processing of the code, because if it lasts too much it will be better to use several new images (instead only a new one) before running it again.

The input will be a sequence of time continued frames in real time, so it will be an array containing k images. At each time the CNN is run, the code will rotate the input array, making the images to move their position. By this way, the older images will disappear and it will save the last ones taken as the new ones.

To make the CNN work, it is necessary to train it. To do that (with the idea we want) it was necessary to search for a dataset of sequences of continued time images where the gesture takes place. This sequence should be of different people, in different angles, sizes, etc. However, it

was difficult to find something similar and we found nothing, so it was a problem to make this part of the code work.

Part 2

For the optimisation part we decided to reduce the non-interesting zones of the image (background subtraction). For that reason, we applied a convolutional neural network for face detection. Its input consists on an image and the output we want is, in case there is a face, its region of interest (ROI), so for the next methods (in case there is a face) it will only be necessary to process the part of the image where the face is.

This convolutional neural network was taken from an existing project of face detection - A Convolutional Neural Network Cascade for Face Detection (Li et al, 2015) - but several changes were needed in order to make the code compatible with the desired project. A lot of tests were made changing the number of layers but the better results were obtained with the configuration provided, so it was maintained. Moreover, several changes of modifying general parameters like the number of epochs, the stride and how the layers are connected were made. So the perfect idea is when you run this code it reads an actual frame, it looks after a face and provides an output saying if there is a face or not. In case there is a face it exits the region of interest where the face is.

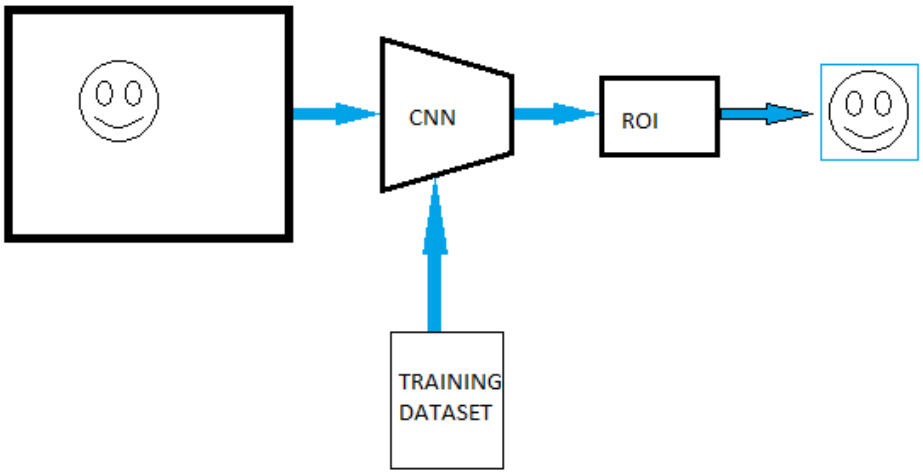

Figure 4: Region of interest. [Own source]

The size of the image does not matter, the CNN reads it and resize it into a standard size and resolution.

The code used needed to be trained in order to be able to detect the face, so we used three different datasets that are going to be explained later on.

This part is connected to the Part 1 in the way that the inputs of Part 1 are the ROI of this part. This will allow the project to run faster.

Part 3

We looked for several forms of reducing the computations. One of them is the background subtraction (Part 2) although there are more methods.

Another way was to reduce the resolution. As there is a lot of computations, if the number of bits were adapted to the amount of data they contain it would allow to make faster computation. So the type of all the used variables need to be related with the information it has to contain. In fact, a resolution of 8 bits for a convolutional neural network is good (it is used on actual projects of it).

Part 4

To make the project work on real time, it was necessary to control the processing time of the different parts of the code.

At the beginning of the program there is a camera that reads images that are used as inputs of the pipeline. Then, the program obtains the ROI of the face. Finally, it processes the CNN to detect if there is a face in the image or not.

Let us suppose that the system lasts X seconds to run the background subtraction of an image (part 2, where obtains the ROI of the face) and Y seconds to compute the CNN to see if the gesture is done (part 1). In this case, the time between the images taken from the camera should be a minimum of X + Y seconds. If during the background subtraction no face is detected, it is not going to be necessary to execute the gesture recognition (part 1). This one will be only ran if a face is detected previously.

Moreover, the code does not need to be running all time. We are interested in taking frames that are continued enough so as to contain a gesture. During the rest of the time, the processor is not running the code.

Final approaches

The objectives and the methods to achieve them were not possible to be programmed. Its main problem was that no dataset was found to train a convolutional neural network to detect a face gesture and there was not time during the project to create one. For that reason, as the objectives had been adapted, the methods became adapted too.

The project was modified to read an image, detect if there is any face and obtain the region of interest. The parts of the guideline of the previous section are only the part 2. The rest of them (part 1, part 3 and part 4)

were made to be used for the gesture detection, but as it was not possible they were left in the theoretical part.

It was also looked for a way to be able to train a CNN for facial recognition without using a data set made for that. It was planned to use an existing dataset to train a system for face features or emotion recognition (UCF-101, youtube-8M, etc) and using the trained system on detecting face features to adapt it to detect facial gestures. However, it was very complex to do and we left it in the planning.

2.5. Hardware used

To make this project work it was necessary a processing unit where to run the developed code. The ones used were:

- MSI GT62VR 7RE Dominator Pro (MSI, 2017). This is the laptop used to code and run the code. It contains a GPU NVidia GeForce GTX 1070 (memory frequency of 8Gps, video memory of 8Gb GDDR5, 256-bit interface) and a processor Intel Core i7-7700HQ with 2.8GHz.
- Jetson TX1 (Nvidia, 2017). Used on the first attempts to code. It contains a GPU NVidia Maxwell (256 cores) and a CPU 4 CPU-cores, 64-bit ARM 4x A57 2MB L2.

2.6. Software used

In order to make this project it was necessary to use several computer programs to run the code and the examples. These are:

- **CUDA:** it is a compiler and a set of tools that allow to run algorithms in the GPU of nVidia, that is translated in more speed. It was initially installed on the computers. The version used is the 8 (Corporation, 2017).
- **CuDNN:** it is a library that belongs to nVidia CUDA for deep neural networks that provides standard routines for convolutional, pooling, normalization and activation layers. The version used is the 5.
- **OpenCV:** it is a library made for computer vision. It contains a lot of functions to deal with images. The version used is 3.2.
- **Tensorflow:** it is a library used for designing and processing of neural networks. The version used is 1.2.1 (Tensorflow, 2017).
- **Git:** it is a software used to save programming code, share it, work in parallel with other people. It also allows to recover versions of what have you been saving.
- **Pycharm:** is a programming environment mainly used by Python language. It allows to code in an ergonomic way and to run and debug it (JetBrains, 2017).

The code used to program the project was python 2.7 (it is the most used for machine learning) and the operative system was Ubuntu.

Moreover, there were several libraries from python that needed to be installed to solve particular problems.

2.7. Datasets used

The code used to do this project, as it was a convolutional neural network, it was necessary to be trained to make it work. Many images to train the system to detect faces were needed. The datasets used for that are:

- Annotated facial landmarks in the wild (AFLW):

  It consists on a set of face images obtained from the net, with different appearances (gender, age, ethnicity, position, etc.) and in different environments and backgrounds. There are around 25000 images with annotations of 21 landmarks per image. Their format is jpg. It has been developed by the Institute of Computer Graphics and Vision. We use it as the positive dataset (Koestinger, Wohlhart, M.Roth & Bischof, 2011).

- Common objects in context (COCO):

  This dataset consists on several images of common objects to train and to test. It also provides several files with annotations of the information of the images. There are around 80.000 images. We use the training dataset as the negative one (COCO, 2016).

- Face detection data set and benchmark (FDDB):

  It consists on a set of images of face regions to solve problems of unconstrained faces detection. It contains annotations for 5171 faces in a set of 2845 images taken from the Faces in the Wild dataset. The annotations say how many faces are there and its coordinates. The data set belongs to the University of Massachusetts. We use it as the test dataset to make trials and to see if our code works correctly (Jain & Learned-Miller, 2010).

2.8. Efficiency methods

The desired system, in order to be used in a real-time situation, need to be faster and cheaper in terms of computational time. For this reason, it was necessary to look for several techniques to reduce the number of operations of the code. The most interesting techniques found were:

- Cascade filters
- Background subtraction
- Image reduction
- Signal detection theory

Cascade filters

The way they work is very simple, it consists in a condition that sets two outputs (Chen, Parada, & Heigold, 2017; Patil, 2014; Ramsrigouthamg, 2012):

- Positive matches, that are sent along to the next feature.
- Negative matches, that are rejected and exit in the evaluation.

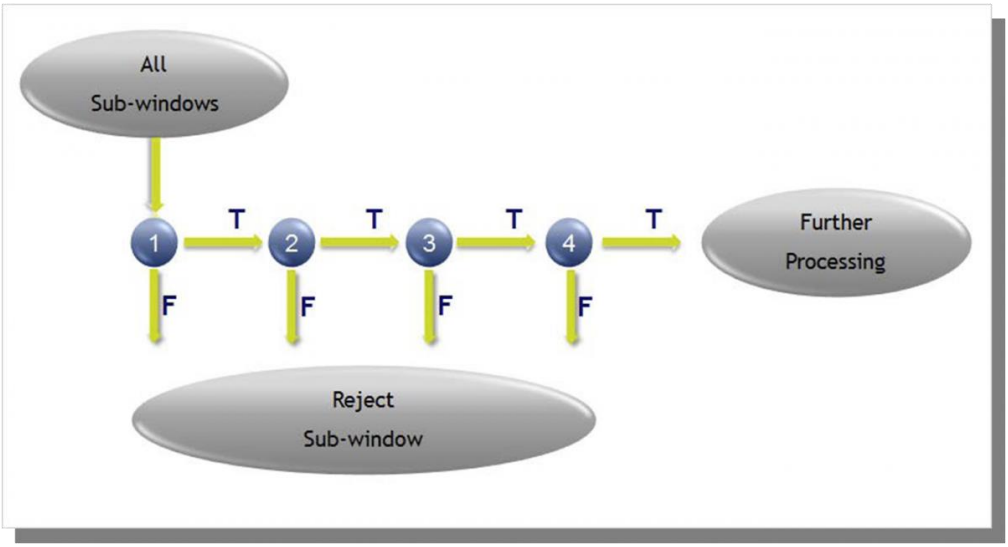

Figure 5: Graphique representation of Cascade Filters [Source : Alliance, E. V. (2017). CEVA. Obtenido de Design Guidelines for Embedded Real-Time Face Detection Aplications: https://www.embedded-vision.com/platinum-members/ceva/embedded-vision-training/documents/pages/design-guidelines-face-detect]

This is useful because as it discards features and regions of the image if they are not interesting, it avoids them to be processed on the following layers, so the number of computational operations and time is reduced.

Background subtraction

This is a method that consists on removing zones of the image that are not interesting for the system. This allows that in next steps of the pipeline, less number of pixels are taken into account to further processes.

There is not a perfect method, so investigators are searching for methods to improve the resolution of this problem. Most of the actual methods are used when the background is static because they suffer small or no changes respect a reference image. By this way, in order to solve this problem, it is only necessary to compare the obtained image with the reference one and to compare the zones that do not change (or are very similar) to discard them. This can be done applying a parameter to define a margin between the similarity of the two images. Other ways can be using the mean filter and the Gaussian filter. Basically it has to fulfil the next equation (frame_i corresponds to an image taken at a time i and image_i-1 is an image that was taken an instant before):

$$|frame\_i - frame\_i\text{-}1| > Threshold$$

On the other hand, dynamic background is an actual problem. As the code is supposed to work on real time, images from one frame to another will not have significant differences on the background. So the background will be very similar with a margin of displacement and brightness.

Image reduction

To analyse an image to find an object, face, etc. it is necessary to look for characteristic patterns. However, as the bigger the number of pixels is, the bigger the number of operations need to be done, that is translated in

processing time. For that reason, the solution is to reduce the size of the input images along with its resolution and convert them to grayscale. The process to detect face features does not depend on colours. There is a minimum resolution the images need to have that does not affect the features that are contained.

Signal detection theory

Detection theory or signal detection theory is a mean to discard the non-interesting information (also called noise). It is based on the probability of a target of belonging a class. In every case there is a factor called sensitivity index that the user determines to make the target belong to a class or to another. There is also a measure called bias that says if an answer is more probable than another. The class that correspond to the target is always the one with higher probability (Heeger, 2003-2007).

Each response of the detection can be:

	Respond "Absent"	**Respond "present"**
Signal present	Miss	Hit
Signal absent	Correct rejection	False alarm

In this case, there are four possibilities. The perfect results are the ones in Correct rejection and Hit. Miss and False alarm interest us to be as reduced as possible. However, it depends on the sensibility we decide, so we could decide to make more false alarms than missed or the opposite. Referring to this project, the best option is to be more prone to miss the signal than to make false alarms (so in case it is not very sure if there is a face it will discard it).

2.9. Explanation of the code

In order to provide a system with efficiency and low power consumption we used several methods explained in the previous section (efficiency methods) used on a CNN to do background subtraction. To do so we based our code on an existing project [A Convolutional Neural Network Cascade for Face Detection (Li et al, 2015)] that is introduced and explained in the part 2 of the section 2.4. Algorithm. Next it is explained the model of the used neural network.

The project used has 6 convolutional neural networks concatenated. Three of them are used for face identification and the other three to calibrate the bounding boxes (Li et al, 2015).

The general pipeline is (Li et al, 2015):

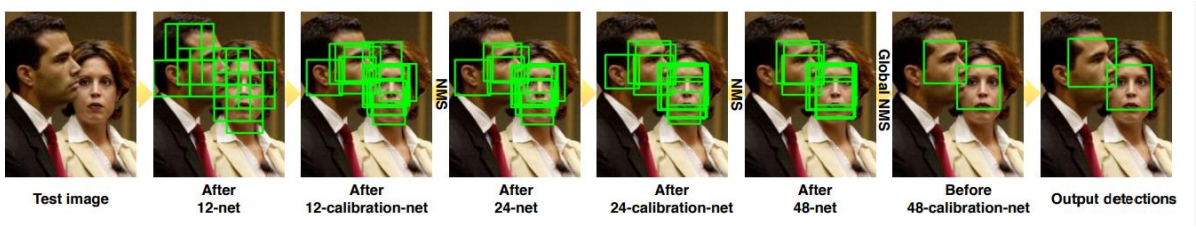

Figure 6: Test pipeline of the detector - detection windows.[Source : Li, H. Lin, Z. Shen, X. Brandt, J. & Hua, G. (2015). A Convolutional Neural Network Cascade for Face Detection. Open Class Version - Computer Vision Fundation, 5327]

The sequence of files that needed to be executed in order are: 12-net, 12-calibration-net, 24-net, 24-calibration-net, 48-net and 48-calibration-net.

The file 12-net scans a test image that can have different scales and rejects the 90% of the detection window. The result is the input of the 12-calibration-net, that are images of 12x12 and adjust its size. After that it takes place the Non-maximum suppression to eliminate the overlapped detection windows (to reduce the number of them).

The remaining windows are resized into a 24x24 format that serves as the input of 24-net. This will allow to reject around the 90% of it. As it was with the 12-net, the results of the 24-net are calibrated by the 24-calibration-net where it is applied the non-maximum suppression another time to reduce the number of detection windows.

Finally, the results are passed through the 48-net that accepts images of the size 48x48 to check the detection windows. It is also applied the non-maximum suppression. The 48-calibration-net is the last thing to be applied and it calibrates the exit bounding boxes with the detection of the face.

The design for the different CNN is:

12-net

The pipeline of this part consists on:

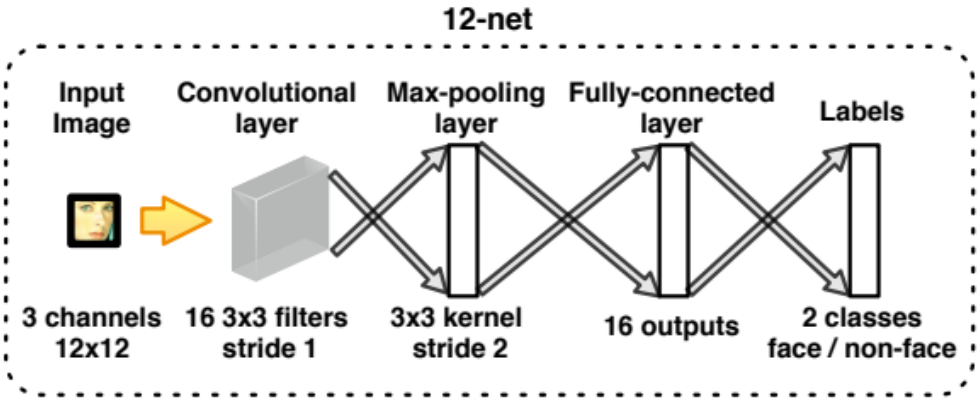

Figure 7: 12-net CNN test pipeline [Source: Li, H. Lin, Z. Shen, X. Brandt, J. & Hua, G. (2015). A Convolutional Neural Network Cascade for Face Detection. Open Class Version - Computer Vision Fundation, 5328]

This convolutional neural network is the first on the pipeline. It takes as input the images to test, it scans them and it converts them into 12x12 detection windows, that are each point of the confidence map which are ([(W-12)/4]+1)X([H-12)/4]+1) where W and H are the width and height of the input image.

12-calibration-net

The configuration of this part of the code is:

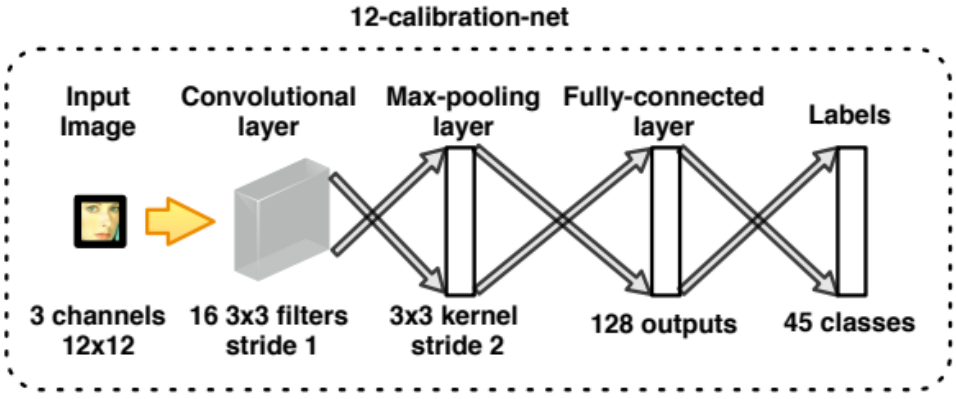

Figure 8: 12-calibration-net CNN test pipeline [Source: Li, H. Lin, Z. Shen, X. Brandt, J. & Hua, G. (2015). A Convolutional Neural Network Cascade for Face Detection. Open Class Version - Computer Vision Fundation, 5328]

This part is used for bounding box calibration. As the image is cropped and resized, the resulting outputs are a bunch of bounding boxes. However, there are several of them in the same image, so we have to use the best of them. Each bounding box has a rate of confidence, so the first solution is to apply a threshold that discards the ones that have low confident patterns. This threshold is the average of the values of the different bounding boxes, so as closer the results are, the confidence on that point will be higher.

24-net

The layers of this CNN are:

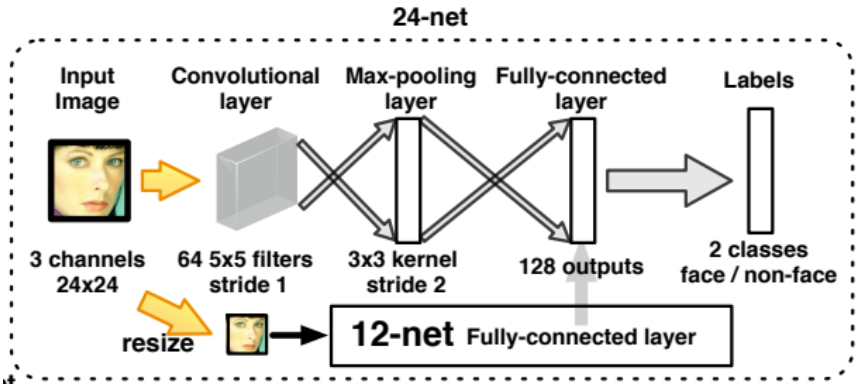

Figure 9: 24-net CNN test pipeline [Source: Li, H. Lin, Z. Shen, X. Brandt, J. & Hua, G. (2015). A Convolutional Neural Network Cascade for Face Detection. Open Class Version - Computer Vision Fundation, 5328]

This CNN is an intermediate one used for binary classification. Its main function is to reduce the number of detection windows. The images of the output of 12-calibration-net are resized to 24x24. Moreover, it is used also the 12-net in parallel due to the fact that it provides better results on helping to detect the small faces.

24-calibration net

The pipeline of this part is:

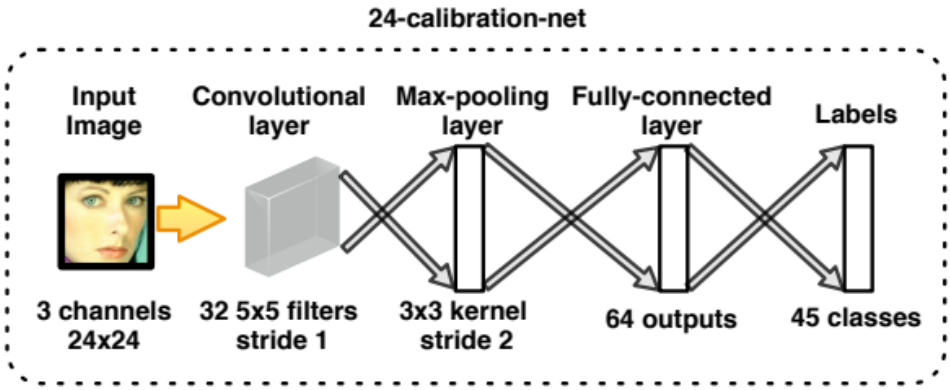

Figure 10: 24-calibration-net CNN test pipeline [Source: Li, H. Lin, Z. Shen, X. Brandt, J. & Hua, G. (2015). A Convolutional Neural Network Cascade for Face Detection. Open Class Version - Computer Vision Fundation, 5328]

This way of calibrating is the same than the one in the 12-calibration-net with the exception of the inputs, that are with a size of 24x24.

48-net

The configuration of this part is:

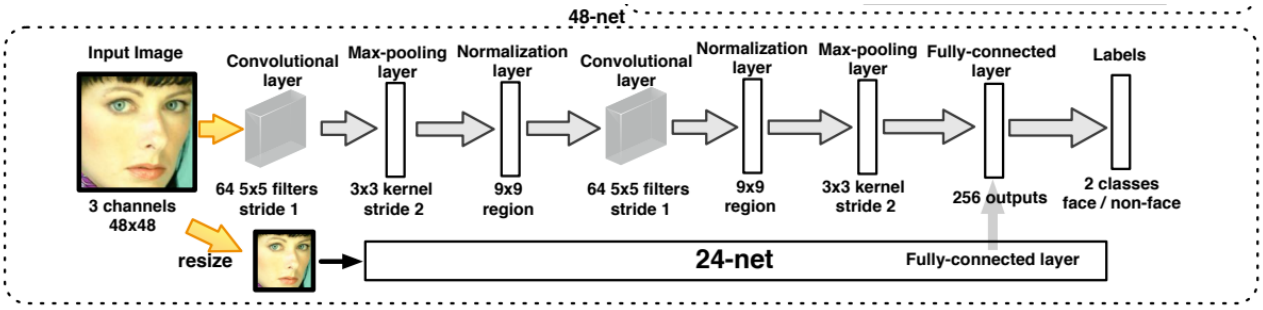

Figure 11: 48-net CNN test pipeline [Source: Li, H. Lin, Z. Shen, X. Brandt, J. & Hua, G. (2015). A Convolutional Neural Network Cascade for Face Detection. Open Class Version - Computer Vision Fundation, 5328]

This is the last classification on the pipeline. It is the most powerful CNN and the slowest due the fact that is more complicated than the others. Like the 24-net, we use the previous CNN (in this case the 24-net) to help to detect small faces.

48-calibration-net

The layers of this CNN are:

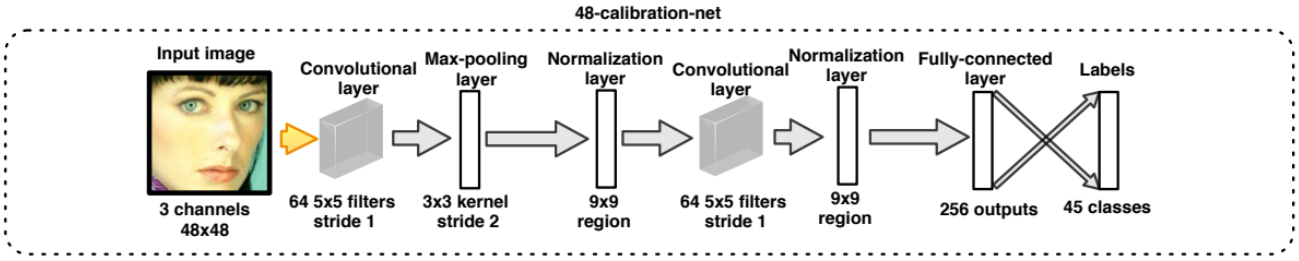

Figure 12: 48-calibration-net CNN test pipeline [Source: Li, H. Lin, Z. Shen, X. Brandt, J. & Hua, G. (2015). A Convolutional Neural Network Cascade for Face Detection. Open Class Version - Computer Vision Fundation, 5328]

The way of the calibration is the same than the 12-calibration-net and 24-calibration-net except for the fact that the inputs are with a size 48x48 and it uses only one pooling layer.

Finally, there is the non-maximum suppression implementation used in the different nets. It works selecting the detection window with a high confidence score and eliminates the rest. In the 12-net and 24-net the method of NMS is not discriminative enough to detect false positives. After

the 12-calibration-net and the 24-calibration-net, the false positives have a higher confidence than the true positives. Also, after these two nets we apply NMS separately for detection windows that have the same scale.

Finally, 48-net is applied globally to all the detection windows with different scales to obtain best accuracy for detection windows and avoid redundant windows in the 48-calibration-net.

3. Results

Graphical results

On this project, to test how good the code worked on the part of face detection, there were a lot of images (around 3000) tested from the Face Detection Data Set and Benchmark (Jain & Learned-Miller, 2010) and images taken by the user. Processing these images with the code using the GeForce GTX 1070 (details on hardware used section) some of the results are:

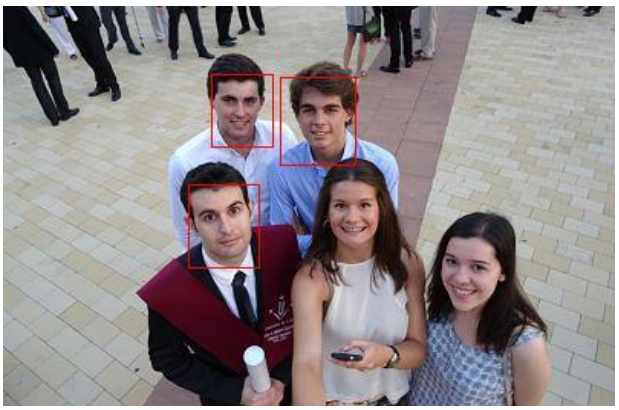

Figure 13: Face detection example 1 [Own Source]

Figure 14: Face detection example 2 [Own Source]

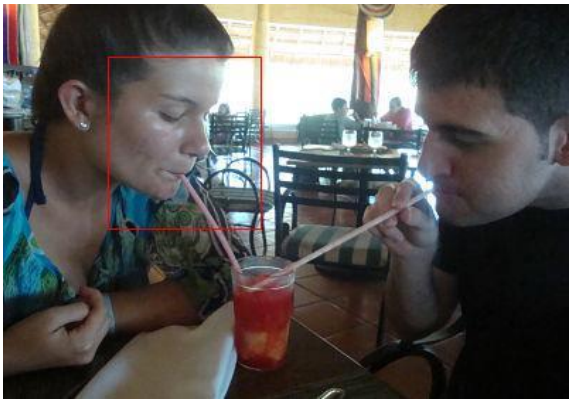

Figure 15: Face detection example 3 [Own Source]

Figure 16: Face detection example 4 [Own Source]

Figure 17: Face detection example 5 [Own Source]

Figure 18: Face detection example 6 [Own Source]

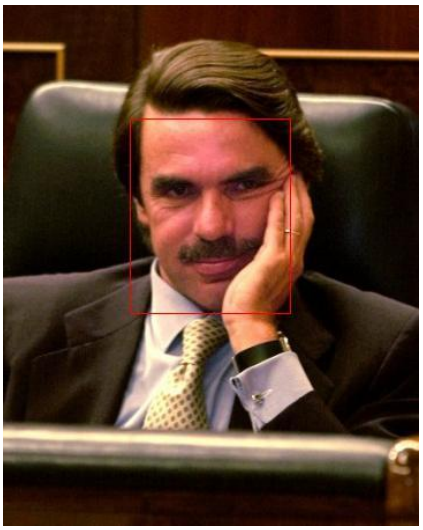
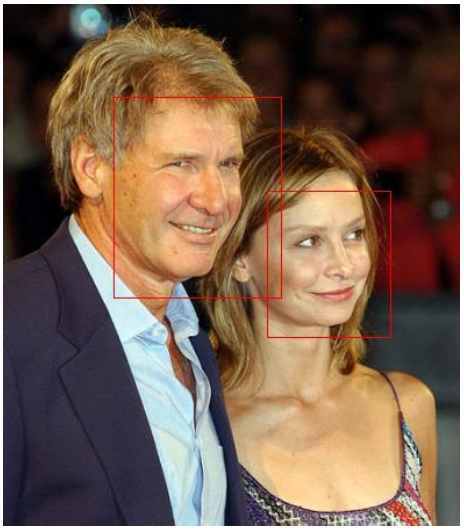

Figure 19,20: Face detection example 6 & 7 [Source: Jain, V. & Learned-Miller, E. (2010). *FDDB: A Benchmark for Face Detection in Unconstrained Settings* . Technical Report UM-CS-2010-009, Dept. of Computer Science, University of Massachusetts, Amherst. Retrived from http://vis-www.cs.umass.edu/fddb/]

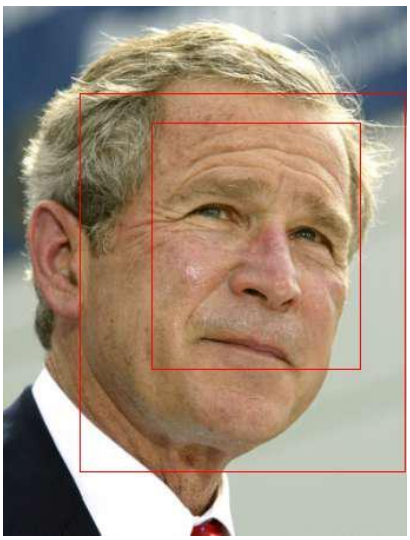


Figure 21,22: Face detection example 8 & 9 [Source: Jain, V. & Learned-Miller, E. (2010). FDDB: A Benchmark for Face Detection in Unconstrained Settings . Technical Report UM-CS-2010-009, Dept. of Computer Science, University of Massachusetts, Amherst. Retrived from http://vis-www.cs.umass.edu/fddb/]

Figure 23: Face detection example 10 [Source: Jain, V. & Learned-Miller, E. (2010). FDDB: A Benchmark for Face Detection in Unconstrained Settings . Technical Report UM-CS-2010-009, Dept. of Computer Science, University of Massachusetts, Amherst. Retrived from http://vis-www.cs.umass.edu/fddb/]

As we can see, the code worked on detecting faces. However, there were exceptions. Even though the majority of faces were detected, there were still some that were not. The results from a set of 400 images were:

- 50% with perfect results. All the faces were detected with the bounding box surrounding them in a good way.
- 15% have good results but not perfect. Being several faces some were detected and others or the bounding box surrounded a part of the face and not all of it.
- 35% did not detect any face.

Also it was tried in a set of nearly 3000 images and the average of face detection was 77%.

From these results we can say that the hit rate is 60% (detects faces when are present), the false alarm (detects faces when are not present) is very low (around 5%) and it misses the image when it is present 35% of the times. We can say that the system is focused on avoiding false alarms, so it will rather miss a face than detect one when it is not present.

Looking at the results we obtained we can say that the results depend on how the image was on the faces. A lot of women with long hair were missed because they did not show they ears, so as the system did not found that feature it discarded them. The main reason of that is because when the code was trained it learned that a face has ears (there were few images with the ears hidden). This also happened with people with hats, because the dataset contained very few images and in the test were not recognized as faces. We can deduce that face detection depends on some characteristic features that works on most of the cases, but not always.

Another fact to take into account is that it does not matter the age or the colour of the skin, it is detected in a similar proportion than the others.

Finally, the main way the faces were detected is when they are showing the main features (eyes, mouth, ears, form of the face, etc.) with the same distance to the camera (no face rotated). On this case, almost all the images worked. When these features start to vary, the system did not work so well and the bounding box did not surround the face perfectly, or even no face was detected.

Mathematical results

On the pipeline there were three parts that should be taken into consideration to see the viability of the project in terms of computational time. These are:

- The time to take an image with a camera. This depends on the camera, so it is not treated on this project. We suppose a digital basic camera for this project because few resolution is needed.
- The time to detect the bounding box. We took into account two factors that we are interested in: the computational time and the success rate. For the first one we have to say that after running the code several times with different images, the average time we obtain running the algorithm using the hardware described in the hardware used section was 0.65 seconds. The complementary actions of reading images and writing them to the disk provided a processing time of 0.8 seconds.
- The time to do the gesture recognition. This was not done and was proposed as future work.

The perfect processing time would be 20 photograms per second for all the pipeline, so each processing of the code would be around 0.05 seconds. When the gesture takes place the computational speed of the pipeline could decrease to 0.1 seconds. By this way, it will have to process around 10-20 images for the sequence of the gesture taking into account that a duration between 1 and 2 seconds of real time video is proposed.

Comparing the desired time with the one obtained in the face detection part, we see that our actual system is not viable because it cannot be implemented on real time on a processor similar than the one used, the time to process the code is too slow. It will need more improvements to achieve a computational speed, being able to run the program on real time.

4. Conclusions and future work

4.1. Conclusions

The purpose of this project was to develop the idea of creating a gesture detection system working at real time. This idea was thought to improve technology methods to work in situations where it is not possible to do it with sound (too noisy places, far distance between the user and the system, mute people, etc.). In order to develop this method, the general structure of the pipeline has been designed and some methods to improve its accuracy and timing have been studied.

The creation of this gesture recognition method requires a lot of optimization to be able to work at real time. Moreover, it would require many improvements and tests to have enough level to be implemented in commercial devices.

The first task was to make a market study to search for similar apps or projects about that. We found that although it was a new idea, there were very close projects, like detecting the position of the hand and fingers to make a translator of sign language at real time. However, these projects used specific methods of optimization and we made a search in order to find out better ones that we could use.

The realization of this project made me realize that the creation of real time systems is very difficult due the fact that we need to optimize the code to reduce the time in all the computations. This requires lots of tests and improvements.

Finally, I must say that although I have not been able to complete the final part of the project, I have been capable to plan and think deeply on the idea. By this way, I have developed planning skills that I consider that are indispensable for Engineers. In addition, I have acquired problem-

solving abilities, that have allowed me to find the best way to solve the problems found.

By working through these changes, I have grown into something new, better, stronger.

4.2. Future work

As the final results were only half of the initial objectives, there are several measures that have been taken to obtain the desired objectives.

The main thing to focus on is to find or make a database composed of several frames that are time continued and with a face gesture present. It is not necessary to be the blinking of an eye, this was chosen because it was a good an easy way to activate a system, but it depends on the dataset found.

Once we have a dataset of a face gesture, the next step is to code the convolutional neural network that has an input of several time continued frames at real time and exits the output if the gesture takes place. This will require to train the CNN with the dataset said before.

Another thing to modify is to code how the images of the region of interest (the faces) become the input of the second CNN. As it is said on previous sections, once the image background takes place, the cropped images are saved into an array where are ordered by time. The way to do so it is also a future modification on the desired pipeline.

In the case a face is detected on a frame, in the next one the position will be closer to that one, so it is not going to be necessary to scan all the image again to do background subtraction. It is only necessary to take into account the position of where the face was and the zones that are close to it. During milliseconds it is not possible to change the position of the face radically. This will be a complementary method of background subtraction.

In addition, we can adapt the type and number of bits of all the variables so it will reduce the processing time. A general resolution of 8 bits could be good for a neural network.

There are other ways of background subtraction and optimization of operations that can be used. Some more research could show us the methods and how to apply them.

Applying all these methods should reduce the processing time enough to be applied on real time to be useful.

Finally, another good idea is to add an option to the code, so the system only will recognize a particular gesture of a particular person. This can be used as a security system.

5. References

Alliance, E. V. (2017). *CEVA*. Obtenido de Design Guidelines for Embedded Real-Time Face Detection Aplications: https://www.embedded-vision.com/platinum-members/ceva/embedded-vision-training/documents/pages/design-guidelines-face-detect

Calacanis, J. (2009). *Angel - How to invest in Technology Startups - Timeless Advice from an Angel Investor.* USA: Harper Business.

Chen, G., Parada, C & Heigold, G. (2017). *Small-footprint keyword spotting using deep neural networks.* Retrieved from https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42537.pdf

COCO. (2016). *Common Objects in Context*. Retrived from http://mscoco.org/

Corporation, N. (2017). *Nvidia Accelerated Computing*. Retrived from CUDA Toolkit Download: https://developer.nvidia.com/cuda-downloads

Deshpande, A. (2016) *A Beginner's Guide to Understanding Convolutional Neural Networks.* Retrieved from https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/

Heeger, D. (2003-2007) *Signal Detection Theory.* Department of Psychology: New York University. Retrieved from http://www.cns.nyu.edu/~david/handouts/sdt/sdt.html

Jain, V. & Learned-Miller, E. (2010). *FDDB: A Benchmark for Face Detection in Unconstrained Settings .* Technical Report UM-CS-2010-009, Dept. of Computer Science, University of Massachusetts, Amherst. Retrived from http://vis-www.cs.umass.edu/fddb/

Jefkine (2016) *Backpropagation in Convolutional Neural Networks.* Retrieved from http://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

JetBrains. (2017). *PyCharm*. Retrived from Python IDE for Professional Developers https://www.jetbrains.com/pycharm/

Koestinger, M., Wohlhart, M.Roth, P. and Bischof, H. (2011). *Annotated Facial Landmarks in the Wild: A Large-scale, Real-World Database for Facial Landmark Localization.* Retrived from https://www.tugraz.at/institute/icg/research/team-bischof/lrs/downloads/aflw

Kuwajima, H. (2014) *Memo: Backpropagation in Convolutional Neural Network.* Retrieved from http://es.slideshare.net/kuwajima/cnnbp

Li, H. Lin, Z. Shen, X. Brandt, J. & Hua, G. (2015). A Convolutional Neural Network Cascade for Face Detection. *Open Class Version - Computer Vision Fundation*, 5325-5334.

Mazur, M. (2015). *A Step by Step Backpropagation Example.* Retrieved from https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

MSI (2017) GT62VR 7RE Dominator Pro. Retrieved from https://www.msi.com/Laptop/GT62VR-7RE-Dominator-Pro.html#hero-overview

Nagi, J., Ducatelle, F., Di Caro G., Ciresan D., Meier, U., Giusti. A, Nagi, F., Schmidhuber, J & Gambardella, L.M. (2011). Marx-Pooling Concolutional Neural Networks for Vision-based Hand Gesture Recognition. *IEEE- International Conference on Signal and Image Processing Applications.* pp.342-347

Nvidia Corporation (2017) *Modules and Developer Kits*. Retrieved from http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html

Patil, R. [RahulPatil]. (2014, May 19). Viola Jones Face Detection Explained [Video File] Retrieved from https://www.youtube.com/watch?v=_QZLbR67fUU

Ramsrigouthamg [Ramsrigouthamg]. (2012, September 16). Viola Jones face detection and tracking explained [Video File]. Retrieved from https://www.youtube.com/watch?v=WfdYYNamHZ8

Rohrer, B. [BrandonRohrer]. (2016, August 18). How Convolutional Networks Work [Video file]. Retrieved from https://www.youtube.com/watch?v=FmpDIaiMIeA&amp=&t=1193s

Smaridge, Z. [ZackSmaridge]. (2013, December 10). Viola-Jones Rapid Object Detection Project [Video File]. Retrieved from https://www.youtube.com/watch?v=Wwn81tVIR10

TensorFlow. (2017). *Installing TensorFlow*. Retrived from https://www.tensorflow.org/install/

Wang, Y. (2014) An Analysis of the Viola-Jones Face Detection Algorithm. IPOL (Image Processing On Line), *5Vol*, pp. 128-148. http://dx.doi.org/10.5201/ipol.2014.104

IMAGES INDEX