Emotion detection in real-time on an Android Smartphone

A Master's Thesis
submitted to the faculty of
ETSETB - UPC
by
Jordi Saumell y Ortoneda

In partial fulfillment of the
requirements for the degree of
Master in Telecommunications Engineering

Advisors: Dr. Jafar Saniie
Montse Pardàs

Barcelona, January 2018

Title of the thesis: Emotion detection in real-time on an Android smartphone

Author: Jordi Saumell y Ortoneda

Advisor: Montse Pardàs, Jafar Saniie

Co-advisors: Thomas Gonnot
Guojun Yang

ABSTRACT

During the last decade the proliferation and capabilities of smartphones have exploded, reaching a situation in which, nowadays, almost everybody carries one of these devices with enormous computational power, cameras and several sensors at any time. The omnipresence, the capabilities and the tools available to facilitate developing applications are the perfect combination for proposing solutions to numerous problems, some existing and some created. In this research, an approach of the widely researched topic of automatically detecting human emotions is brought to the Android platform by means of an application. A system is developed using some existing tools in order to detect emotions with few computations so that it may run in real time on smartphones with less computational power. Given the limited time available for developing this project, the scope of this work is to set the foundation so that the application can be improved in the future by other researchers; therefore, some limitations are set, and some stages are not fully optimized. Finally, future improvements are proposed to facilitate the continuation of this project.

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisors Dr. Jafar Saniie, Montse Pardàs, Thomas Gonnot and Guojun Yang for their guidance and for aiding me in the decision making. Also, to Guojun Yang for his help in the first steps with TensorFlow as well as with the difficulties dealing with some C++ libraries.

I would also like to thank all the volunteers, who accepted being recorded for the database.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

One of the singularities of human beings that have contributed enormously to the development and growth of mankind is the ability to communicate precisely with rich and powerful spoken (and later in history, also written) languages. Having said that, a significant percentage of what is communicated does not circulate through those languages, but through nonverbal cues. These cues can be in the form of gestures performed, for example, with the hands; or also facial expressions that convey information about what is inside but not necessarily spoken.

Given how relevant facial expressions have been to human interactions, it is not surprising that they have been researched for centuries. In [17] it is described how studies on facial expressions were already performed in the Aristotelian era —4th century BC. It can also be read how the foundations of current research were written in the 17th century. Then, in the 19th century Charles Darwin developed one of the important works on facial expression analysis [17] that is more helpful for the science of automatic recognition performed by machines. His work classifies different expressions in groups by similarities, as well as the facial deformations associated to each of these groups.

Then, in the 1970s the psychologist Paul Ekman and his associates developed essential work ([25], [26], [27], [31], [33], and more) that has been referenced and used by the majority of posterior researchers on the topic of expression recognition.

With the development of computers and the rapid advancement of the capabilities of these machines, the research on facial expressions shifted from psychologists to computer vision scientists, and a myriad of papers and work have been developed to automatically recognize expressions, especially since the 1990s.

Researchers trying to build automatic systems have dived into the topic from different approaches. Some have built systems that evaluate pictures and others have built systems that process videos, even in real time. In some cases the information provided by time is taken into account, with models that have memory, and in others the previous samples are ignored. Researchers have also worked with occlusions of portions of the face, mostly the mouth or the eyes; some to discover what organs or regions convey more information and others to develop an algorithm capable of functioning without all the information available. Different illumination conditions are tested, as well as camera angles —frontal face, profile, etc—, and much more.

Most of these approaches mentioned are based on a similar structure, where a first stage detects —and potentially tracks— the faces on the image, by means of multiple systems that involve an entire and very active branch of computer vision. Then the most relevant information of the face image extracted is processed and generally condensed so that it is easy to work with than the entirety of the pixels; but this is done in a huge amount of different methods. Finally, a classifier decides the emotion that considers most likely, and again, several different classifiers have been tested and trained in multiple ways.

Also, a significant amount of databases have been built, and many of these have been made available to other researchers. Actually, several investigators have also referred to the difference between posed emotions and spontaneous or real ones, towards which the trend is veering. Most databases cover the six basic emotions —anger, disgust, fear, joy, sadness and surprise—, or even fewer; but other databases are more complete. However, no data collection has been assembled in a

complete and versatile enough manner so that it became a standard; therefore, no benchmarks are available.

Accompanying all the research that has been undertaken, there are several applications for automatic recognition of emotions. One of the most straightforward applications is for interaction between human beings and machines. First, with the growth of robotics and the fact that an important part of human interactions are done through nonverbal communications, a necessary attribute of robots is the understanding of human emotions. And maybe with less priority but also important, if humanoid robots are build producing facial expressions similar to human beings will probably be desired, too.

There are many more interactions with machines that could benefit from recognizing emotions. An example would be security systems in cars to prevent users from driving when in a dangerous state, such as angry, nervous, or excessively tired.

Following the same topic of security, surveillance systems from governments (without discussing the ethics of this usage) will probably benefit from this approach, and surveillance systems in casinos have been implementing it for a long time.

Also, a different kind of surveillance that may have even more benefits from recognizing emotions is the one performed by content providers. Websites track all the movements users do in order to obtain useful information, as for example, what interests users and what does not. TVs, tablets, smartphones and more devices could use the frontal camera to know how the user feels with each interaction, hopefully only with user consent.

Another important branch of applications is related to health. Psychologists and psychiatrists can benefit from these systems too, as emotion recognition is crucial for their jobs. For example, some patients have emotion perception deficits, such as autism or schizophrenia.

Other possible applications could be for aiding the visually impaired, education —educational games, remote lessons, etc— and more.

1.1. Objectives

After a brief introduction to the topic and its applications, the objectives of this project will be described.

The main goal of this project was to build an Android application capable of detecting emotions on real-time. Also, given the possibility of this project being continued by someone else, and the lack of expertise on the topic of facial expression recognition, reviewing the state of the art in an as broad as possible way became a secondary goal.

As it was discovered that emotion detection libraries were closed-source and did not offer options of tuning, the goal of the project was made more specific: building a system that could detect emotions on an Android device —so, with limited computations—, open to being further improved in the future. This system needs to function with the images obtained by the camera on real-time.

With the complexity of the task, some restrictions were established to facilitate the task. Only frontal faces need to be analyzed, and transitions between emotions may be ignored. Also, posed expressions are used.

The system is to be tested with subjects from the laboratory, and may be included in the training.

1.2. Development

With the intent of this work expressed, a short summary of what has been done will be introduced before commencing the extended explanation.

The state of the art has been reviewed and part of it summarized in this document. The structure of the system has been designed, in a parallel manner to what is done in the state of the art. A method for detecting faces is chosen, with the knowledge that it is not necessarily the best option as the state of the art changes constantly. Then a method to select the most important features is chosen based with the intention of reducing all the information that needs to be processed to a few numbers, to reduce the computations performed by the last step: the classifier. The information reduction is performed by an algorithm found on the 'dlib' [12] library that finds the position of 68 face landmarks fast enough for real time. These positions are used by the classifier to predict the emotion in the present image or frame.

To train the classifier the Radboud [8] database is used. When trying it in video it is discovered that the performance in moving video decreases significantly respect to the static images of the database. Then, an android app to build a video database that is labeled automatically is created, and a small database is constructed. Several tests are run and another approach is compared.

Finally, several improvements are proposed for possible future researchers that continue this work.

The state of the art and the theoretical background of the tools used can be found in the following section. After that, the methods used will be described in more detail. The next section will show and discuss the results. In sections 6 and 7 the conclusions and future development can be found, respectively.

Finally, there are other portions that can be found of the table of contents: time plan, budget and glossary.

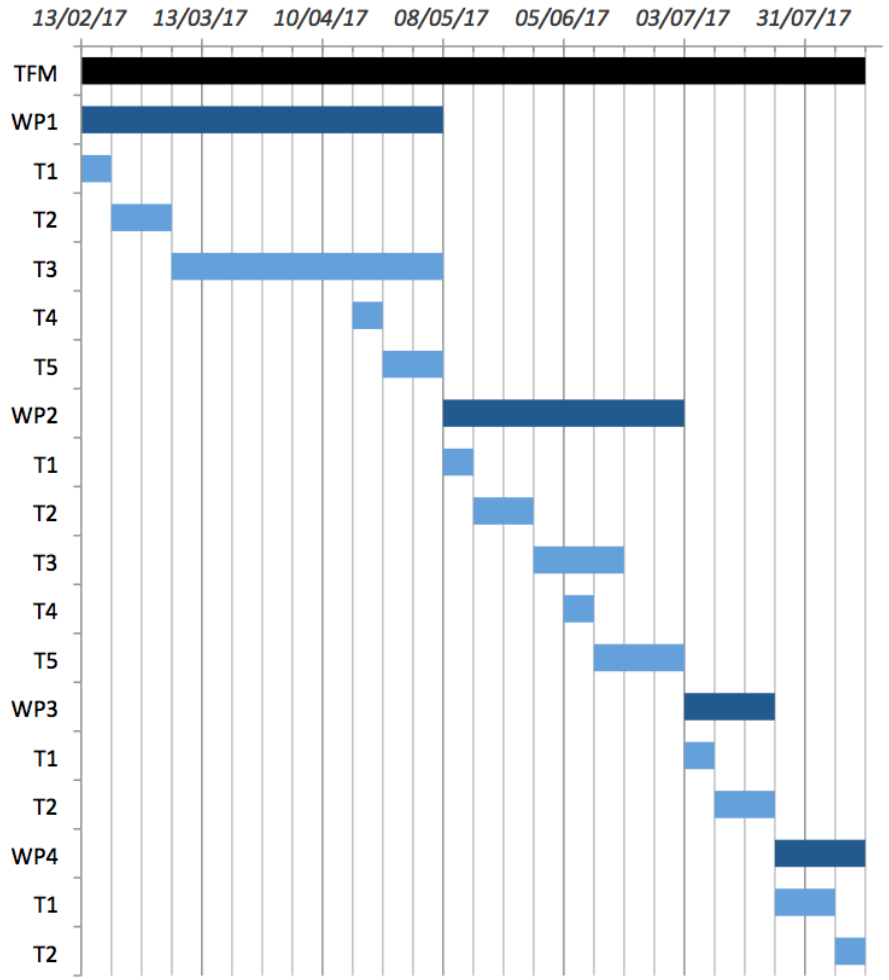
1.3. Time plan

In the following page the Gantt diagram is shown, with the tasks and Work Packages. It is important to note that the first 12 weeks (work package 1) consist of a workload of 10-15 hours per week, as this project was being pursued simultaneously with subjects. The other 14 weeks are full-time on the project.

During the first part of WP1 there was no plan because the project scope was not well defined, as it was exploratory. Once the possibilities had been discussed and state of the art had been reviewed, the following work was determined and planned.

WP2 and WP3 went approximately according to plan, only slightly slower. Some possible tasks were left undone, but these tasks were already set as optional depending on the available time.

Task	Start	End	Duration
TFM	13/02/17	14/08/17	26
WP1: Research	**13/02/17**	**08/05/17**	**12**
T1: Checking libraries	13/02/17	20/02/17	1
T2: Affectiva app	20/02/17	06/03/17	2
T3: Reviewing state of art	06/03/17	08/05/17	9
T4: System design	17/04/17	24/04/17	1
T5: OpenCV and face detection	24/04/17	08/05/17	2
WP2: System on desktop	**08/05/17**	**03/07/17**	**8**
T1: dlib desktip and face det.	08/05/17	15/05/17	1
T2: Feature extraction	15/05/17	29/05/17	2
T3: Choosing DB, preparing data	29/05/17	12/06/17	2
T4: Lab subjects DB	05/06/17	12/06/17	1
T5: Training	12/06/17	03/07/17	3
WP3: System on Android	**03/07/17**	**24/07/17**	**3**
T1: Setting up dlib on Android	03/07/17	10/07/17	1
T2: Implementing the algorithm	10/07/17	24/07/17	2
WP4: Report and presentation	**24/07/17**	**14/08/17**	**3**
T1: Writing the Final Report	24/07/17	07/08/17	2
T2: Preparing the presentation	07/08/17	14/08/17	1



2. BACKGROUND AND STATE OF THE ART

In this section a review of the literature is presented. The technology used in the methodology of this project will also be explained.

2.1. Emotions and the face

Emotions could be described as strong feelings that are related to one's circumstances or mood, and have different characteristics associated to strong mental activity and some pleasure or displeasure. These feelings are important and thoroughly researched because they have a critical influence on human behavior and decision making. Therefore, recognizing emotions can help partially understand how fellow human beings may react to different events. For example, it may help someone decide how to conduct himself, or what to say or avoid saying; in other words, it is essential for social relationships.

A common way of showing emotions; and thus, of perceiving them on other individuals, is through alterations in facial expressions. For this method to be reliable it is necessary that the relation between emotions and changes in facial expressions are common among different subjects, including different cultures.

Darwin already suggested that they were in fact universal, but there was no consensus. For that reason, Ekman and Friesen studied subjects from different cultures and concluded that certain emotions are universally recognized [25], and later on provided strong evidence [26] to refute the critics. Since then, the universality across cultures of emotion recognition through facial expressions has become an agreed fact. However, in [26] it is also acknowledged that the expression of emotions does depend on social context. An example offered is the fact that Japanese subjects in the presence of an authority are more reluctant to show their emotions through facial expressions.

In [25] there are six basic emotions researched: happiness, sadness, anger, surprise, disgust and fear; however, other investigators have identified an enormous variety of other emotions that can be displayed through the face. An example of these investigations is [29], where 136 emotional states are discerned and classified in a tree structure where some more basic emotions are on top (love, joy, surprise, anger, sadness, fear) and these primary emotions are divided into secondary emotions, and also into ternary emotions.

2.1.1. Parametrization

With this enormous range of facial expressions that can be exhibited, associating them to emotions becomes a difficult task. In order to facilitate this, some researchers have parametrized the possible changes in facial expressions, and most of the research on the topic now references two of the approaches: the Facial Action Coding System (FACS) [27] and the Facial Animation Parameters (FAPs) [28].

Before the appearance of the FACS, researchers relied on human observers to determine emotions. This approach was not reliable because the observer could be influenced by context [27] and, for example, give more importance to the voice. For that reason Ekman and Friesen developed a system that allowed a more scientific approach, and has since become a de-facto standard.

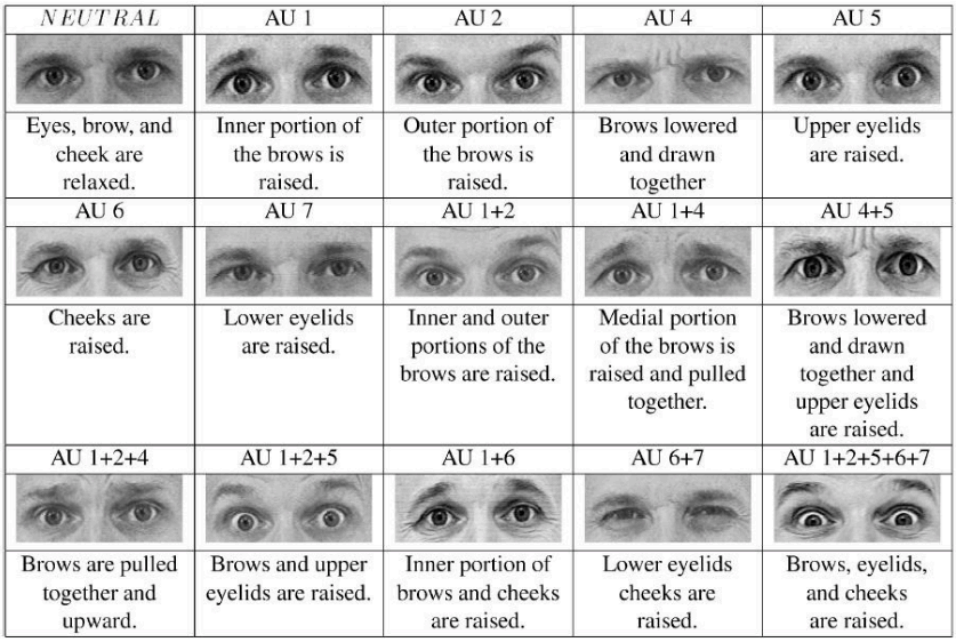


Fig 1. Examples of AUs in FACS. Figure extracted from [30].

It consists on identifying the muscle groups that cause changes in facial behaviors: Action Units or AUs. Then an expression is associated to one or more AUs, and since there is a relationship between emotions and facial expressions, the list of different combinations of AUs that are connected to each emotion can be established. Hence, a system that can recognize AUs will be able to determine the emotion of the subject through a conversion table. In [31] there is a comparison of FACS with some other systems that are extensions or modifications of FACS.

An example of some AUs in the eyes regions can be seen in figure 1 where some AUs can be seen with an image and a text description.

The FAPs were introduced by the MPEG group to be used as a standard for describing facial animations, and they are related to the FACS. As it can be seen in the MPEG-4 standard [28] there is a neutral face model with all muscles relaxed and then 84 feature points (FPs) are used to understand the deformations respect the neutral face, which are produced by muscles.

Through FAPs, head motion and facial actions are represented and a value determines the intensity or amount of deformation. A set of normalizations are defined to represent all face models. Although these parameters were built for animations, they have been used by emotion recognition researchers, by applying some of these definitions on detected faces, such as in [32].

2.1.2. More relevant information

Although most research aims at distinguishing emotions based on the facial expressions in a certain image, there is another characteristic that some have taken into account. Facial expressions are not constant through time, and instead offer three different stages: onset, apex and offset. The first stage corresponds to when the expression appears, the apex is when the expression is at its peak —and it is what most researchers take into account— and the offset refers to when it fades away.

This allows —through a more complex system— to use more information that can help improve detection as well as understand transitions between emotions. There are complexities, such as the fact that there are a lot of possible transitions that need to be taken into account, or the fact that a system that requires the three stages will not work if it cannot be guaranteed that the information from all stages can be recorded.

There is another categorization of expressions that is extremely important for research that is different from the previous ones. Emotions can be 'posed' or 'spontaneous'. The former are artificial, and appear when the subject is asked to compose the expression; and the latter is the one that occurs when the subject performs it genuinely and without being conscious about it. Most researchers working on automatic recognition have used posed expressions because these are easier to recognize, and especially, much easier to produce a database.

Although posed expressions are more convenient, psychologists have shown that these are different from the spontaneous ones in both appearance and temporal characteristics; for example, in [33]. As the expressions that occur in real-life are spontaneous, it is more useful to build systems to recognize those, and for that reason researchers are moving towards spontaneous expressions recently. Furthermore, most posed databases use excessively exaggerated posed expressions, that make them even less realistic. For this reason, this topic is especially relevant for databases.

Finally, the six basic expressions that have been mentioned before are the ones that have been researched the most. In [17] results from several papers are compared and the mistakes between different expressions are summarized. Anger and disgust are commonly confused; fear is confused with happiness and anger; and sadness is confused with anger too. On the other side, surprise and happiness are the easiest to recognize.

2.2. Databases

In the previous section some information about the relationship between emotions and facial expressions has been presented. This information will be relevant to one of the critical aspects of the process, which is the database.

Databases are used to train the algorithms that are developed to solve the problem of emotion detection, and also to test the results and extract conclusions. There is a large number of databases with annotated facial expressions or emotions that are typically used. Ideally, there would be a standard database that was used by all researchers and allowed proper benchmarking, but this is not the case. The difficulty is that such database would have to satisfy the needs of all researchers.

The first challenge, as it has been mentioned above, is the distinction between posed and spontaneous databases. Given that spontaneous and posed expressions are different, with posed expressions being more useful, and the research community shifting towards spontaneous databases, a standard should be based on this kind of expressions. This is a challenging process that generally is approached by developing a video kiosk where subjects watch emotion inducing videos. These subjects are recorded by a hidden camera and once the recording is done they are asked for permission to use the images for research. If granted, they also typically respond to questions of what they felt during the video. If this process was not challenging enough already —it has to be done for a large number of subjects and videos—, some expressions were found to be difficult to induce, such as fear and sadness. Furthermore, expressions performed by the subjects were sometimes misleading as some subjects looked sad when they were actually happy [36].

Other requirements of a standard and future proof database include recording in multiple lighting conditions, having head rotations, different camera angles and occlusions of portions of the face. Some researchers also encountered problems because they worked with the three stages of the emotions —onset, apex and offset— and most databases did not include the entire sequence.

Also, the database needs to be properly labeled. Labels of emotions are necessary, but some researchers also require AUs, or other metadata that can be useful for different purposes —e.g. lighting, presence of occlusions, ethnicity, etc.

The databases described next are some of the existing ones, which are not as complete or large. They are separated in two groups, depending on whether they contain pictures or videos.

2.2.1. Picture databases

There are several picture databases annotated with emotions or expressions, with a wide variety of characteristics. It follows a summary of some of the most relevant databases and their characteristics. The information from databases without a direct reference has been extracted from [17] and [43].

JAFFEE [44] is a database that only contains images of Japanese subjects with 7 facial expressions, all of which are poses using multiple AU's —i.e. involving several facial muscles in the expression, instead of, for example, faking happiness by moving only the lips in a smile.

The FERG [45] is extremely large (55,767 images), but it contains computer-generated images instead of pictures.

Then, the AR [46] Face database contains 4000 color images of 126 subjects and pictures have different illumination conditions, as well as occlusions —scarfs and sunglasses.

A large database is the CMU pose, illumination, expression (PIE) [6] that has 41,368 images of 68 subjects, with four different facial expressions. 43 different illumination conditions are included as well as 13 camera angles. Although the database is quite large, the facial expressions contained are only 'neutral', 'smiling', 'blinking' and 'talking'.

Another database that has been used by several publications is the Cohn-Kanade, also known as CMU-Pittsburg [7]. 100 subjects were selected in the age range between 18 and 30 years old, where 65% were male. Regarding ethnicity, 15% were african-american and 3% asians and latinos. The database contains 500 image sequences of posed expressions, and instead of emotions it has AUs (from FACS [31]) annotated. 23 different facial displays were tested, including different combinations of action units, beginning from a neutral face and moving towards the target pose.

Finally, the Radboud faces database [8] is built with 67 models including caucasian males, females, boys and girls; and Moroccan dutch males. 8 emotional expressions are displayed: angry, contemptuous, disgusted, fearful, happy, neutral, sad, and surprised. Also, 5 camera angles are used and 3 gazing directions (of the eyes) are recorded for each emotion and subject.

2.2.2. Video databases

Three of the video databases will be presented. The first one is the MMI [9], comprising over 2000 videos of about 50 subjects (48% female) that display facial expressions upon request under natural lighting conditions; and some subjects displaying spontaneous expressions. Ages from 19 to 62 are covered, with European, Asian and South American ethnicities. Expressions consist of a single

or multiple AUs, and are preceded and followed by a neutral face. These AUs and the corresponding emotions are annotated, including onset, apex and offset.

Then, the DISFA [10] database consists of non-posed facial expressions with AUs annotated by two FACS [31] experts, including the intensity (0-5) of these AUs. 27 adults (12 females) and different ethnicities. As the expressions are not posed in this instance, the expressions are induced by showing an emotive four-minutes-long video to the subjects.

Finally, the Multimedia Understanding Group (MUG) database [11] contains 86 subjects —51 men— who perform facial expressions while sitting on a chair with a blue screen background. Good lighting is used, to avoid shadows. All subjects are caucasian and between 20 and 35 years of age and a total of 1462 sequences are made available.

The first part of the database is formed by basic posed expressions, consisting of anger, disgust, fear, happiness, sadness and surprise; whereas the second part contains laboratory induced emotions. The posed expressions are done following the FACS [31] manual, but before and after a neutral face is shown. The categories of these sequences are labeled, as well as 80 landmark facial points of 401 images.

The set that contains induced emotions has more expressions than the basic ones, but it has not been labeled yet.

2.3. Face detection

Face detection is the process through which a computer algorithm analyzes digital images and detects the regions that contain human faces, generally including the size and position of each face. It is one of the most researched computer vision problems, with a multitude of papers and work done.

2.3.1. Viola Jones

One of the most frequently used techniques is the Viola-Jones, described in Robust Real-time Object detection [18]. When it was presented, in 2001, it was one of the first real-time algorithms with a high accuracy and reliability. For these reasons it has been implemented in several tools, such as OpenCV. This makes it accessible to everyone and is part of the reason for its popularity.

This framework offered three key contributions: a new way of representing images called “integral image”, which speeds the computation of certain features; a learning algorithm based on AdaBoost to select the most critical features; and the “cascade” to combine classifiers for a much faster image processing.

Some reasons for using features is that the relevant information can be stored using less space (bits, in a computer) while allowing much faster computations. In this framework, the features used have some similarities with the Haar Basis functions, but they are more complex. These features consist of “dark and light” rectangles (figure 2 on the left) where the sum of the pixels is computed for each color. Then the value of one color is subtracted from the value of the other color. Although other filters are more versatile —e.g. steerable filters—, the computational speed improvement of using rectangles makes these a better option.

The integral image at a location (x, y) contains the sum of the pixels above and to the left of that position, which is equivalent to a mathematical double integration. This integral image can be computed with only one pass, by using some previously computed values, which makes it fast to calculate. Then, computing the integral of a rectangular area is immediate, by using the four vertices; furthermore, the difference between adjacent rectangles only needs six vertices or 'pixels' from the integral image.

The problem with these features is that there are far more features than pixels. What the authors realize is that using only a small number of these features can yield a robust classifier. Then, they use a variation of the AdaBoost algorithm to select the features and train the classifier.

In a short explanation, AdaBoost selects basic classifiers with poor performance and combines them to obtain a system with good performance. Many learning problems are solved and weight values are applied based on the results, this process is followed several times until a reliable classifier is obtained using few of the initial basic classifiers.

The third contribution, the cascade, achieves much lower computation time by focusing on a simple system that discards most of the area of the image. The idea is that a smaller, simpler and faster classifier is run in all the sub-windows and discards the majority. These simple classifiers must not discard almost any sub-window that contains a face, but on the other hand it has to reject as many non-face regions as possible. An example would be the backgrounds, which are notably different from a face and may cover an important proportion of the image. Then, in later stages of the cascade other (more complex) classifiers are used to reject as many false positives as possible, thus achieving a low error rate.

The AdaBoost algorithm briefly explained before is used to obtain these staged classifiers. Starting with a two-feature classifier adjusted to minimize false negatives, according to the validation data used 100% of the faces can be detected with a false positive rate of 40% in the first stage. The cascade can be seen in figure 2 on the right, where the first stage is the most simple —least time consuming— and rejects a large amount of sub-windows; then the positive results proceed to the following stages, which are more complex and more precise so that the the last stage is only performed on very few sub-windows.

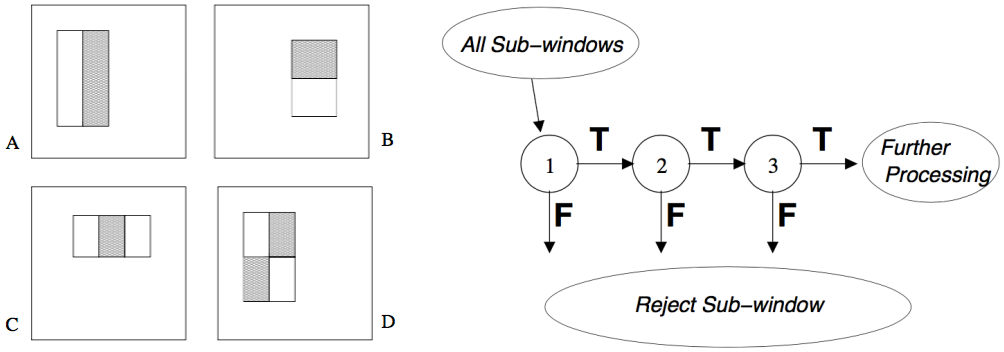


Fig 2. Viola-Jones rectangles on the left, cascade on the right. Extracted from [18].

This process is efficient under the assumption that the majority of the image portions are negative —non faces—, which is almost always true. But besides being efficient, it can also lead to

extremely low false positive rates. As an example, a cascade of 10 classifiers with poor rates, 30%, will yield $0.3^{10} \approx 6 \times 10^{-6}$.

Their final detector consists of a classifier with two features (rejects 60% of non-faces and detects almost 100% of faces), another classifier with five features and rejection of 80%, three layers with 20 features each, two 50-feature classifiers, five 100-feature classifiers and finally twenty 200-feature classifiers; where these sizes were found by trial and error. This classifier was significantly faster than the state of the art, while achieving a similar accuracy.

2.3.2. HOG

A few years later Histograms of Oriented Gradients for Human Detection [19] was built with a simpler structure and better accuracy than the state of the art at the time.

This method evaluates well-normalized local histograms of image gradient orientations in a dense grid. Generally objects can be characterized by the distribution of local intensity gradients or edge detections, even without knowledge of the exact value or position. The image is divided into small regions (cells), and a 1-D histogram of gradient directions (edge orientations) is computed for the cell. Previously, however, the local responses are contrast-normalized to achieve a higher invariance to illumination and shadowing by measuring the “energy” of blocks of the image (larger regions than cells). These normalized descriptor blocks are the HOG descriptors. A dense and overlapping grid of HOG descriptors and the combined feature vector are used together with a SVM based window classifier to build the human detector. The overview of the blocks can be seen in figure 3.

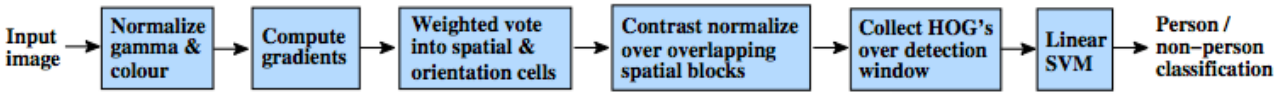


Fig 3. Overview of the feature extraction and object detection chain of HOG. Extracted from [19].

Different variations of the HOG are presented, each offering different accuracies and run times, but the results are better than the previous approaches. A drawback, however, is that blurred images are not properly detected, as detection of abrupt edges is critical.

In the ‘dlib’ library [12] (it will be introduced in 2.4.1) this HOG approached is used to train several object detectors, including the face detector.

2.3.3. Comparison

Viola-Jones was an important innovation in the field and since then a number of proposals have improved their results by means of new approaches. First attempts extracted hand-crafted features and trained with traditional machine learning algorithms. This, however, requires expert in computer vision for crafting the features, and as each component is optimized separately the steps are generally sub-optimal [6].

With the increased attention on the field, several benchmark systems have been built to achieve a fair comparison of the different systems, as there are several dozens of methods that have been proposed in the last few years.

One of the most commonly used benchmarks is the FDDB [4] and it was built to provide a way to compare different face detection algorithms. First, a common evaluating scheme was not available; and second, existing data sets were missing some aspects of real-world scenarios.

So, this dataset was build to solve issues such as the amount of faces available, increasing it to 2845 images containing 5,171 faces. Also, images that are more difficult to analyze are included; with occlusions, difficult poses and low resolution or blur; and all of this is properly annotated. The face annotation is done with an ellipse, to account for rotation, and both grayscale and color images are available.

Other issues are how to decide whether the detection from the algorithm being tested is correct, and for that reason a solution that defines correctness is proposed. This includes an algorithm to find the correspondence between the annotations in the dataset and the predictions, two rigorous methods for evaluating the performance that cover different applications, and the source code to implement these procedures. Finally, the results of several standard algorithms have been compared following these guidelines.

Another posterior benchmark, the WIDER FACE dataset [5], includes Viola Jones as a reference, and shows that new technologies are significantly better, as it can be seen in figure 4. These plots present the precision versus the recall, and the Viola-Jones is the light blue line, which has the smallest recall for a certain precision in all three images.

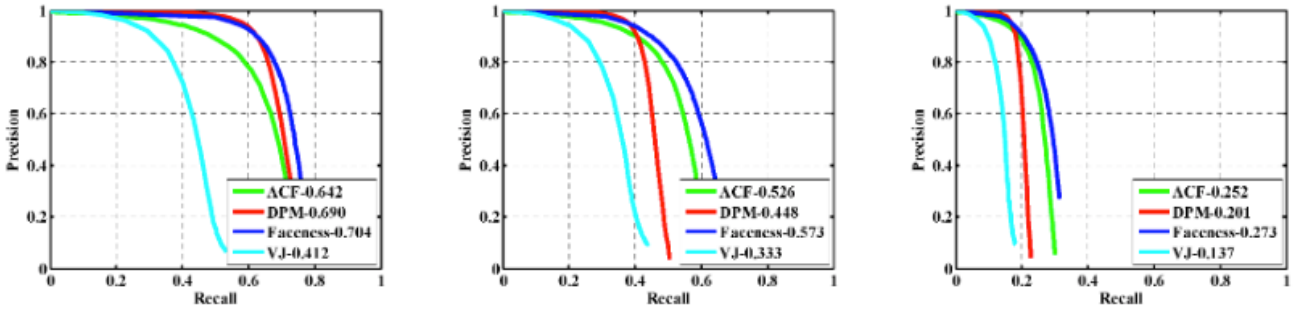


Fig 4. Results of the comparison. Easy set on the left, intermediate on the center, hard on the right. Extracted from [5].

According to [4], the face detection research does not perform well enough for real world applications, and for that reason they have built a data set for benchmarking that is 10 times larger than previous ones. Some examples of situations in which good-performing algorithms fail in real-life applications are with heavy occlusions, small scale, or uncommon poses. For this reason, this proposed dataset is further annotated than others, including occlusions, poses, event categories and, of course, the face bounding boxes. These extra annotations allow to build systems capable of performing well under more challenging circumstances.

It includes 32,203 images with 393,703 labeled faces in different scales, poses, occlusions and with 60 event classes. The goal is to encourage the community to use this more challenging dataset to build tools capable of detecting faces in more difficult conditions, such as surveillance cameras.

2.3.4. Tracking

Detecting faces on images is computationally expensive, and the previous algorithms generally allow between one and 5 frames per second —of course, it depends on the computational power, the size of the images, and more. An alternative approach is to detect faces once every few seconds, and in the frames in between track them, using the extra information provided by the detection in the previous frames. The simplest approach is to just suppose that the face is at the same position, but this can only be done when the movement is slow and the time between detections is short. More complex approaches include using optical flow to estimate the movement or other advanced algorithms. The key is that these algorithms are generally faster to run than the detection, thus achieving significantly higher frame rates (or saving power). If the time window is too large, however, the system will take some time to detect a new face.

2.3.5. State of the art

Recently exceptional successes have been achieved through deep learning and especially convolutional neural networks (CNN), as in several other computer vision fields.

An approach with a significantly good performance is the faster region-based CNN (faster R-CNN) [6]. They introduce a Region Proposal Network (RPN) that proposes regions almost cost-free, that are used by the Fast R-CNN for detection, afterwards. The RPN consists of a fully convolutional network that predicts object bounds and 'objectness' scores at each position, telling the R-CNN "where to look" (attention). The accuracy achieved is state of the art, while running at 5fps on a GPU. Current R-CNN have the bottleneck on the region proposals; therefore, making it almost cost-free is an important progress towards real-time.

In [7] they use [6] as basis and further improve it combining several strategies, such as feature concatenation, hard negative mining, multi-scale training, model pre-training, and proper calibration of key parameters. These improvements make this work the best model in terms of ROC according to the FDDB benchmark, at the time of publication.

2.4. Feature extraction

To know what emotion a face is showing, an algorithm that detects it is needed, but that algorithm needs information to work with. Although it is possible for such algorithm to use the whole image, usually a previous step attempts to extract the most important information so that the classification step is faster. This previous step will be described here, first comparing the relevance of the different parts of the face and then briefly describing some of the most commonly used approaches. Finally, the method use in this work will be explained a bit more thoroughly.

In order to detect emotions facial features need to be extracted, but it is convenient to know what features are more relevant than others. Some features are always visible, such as eyes, lips, brows and cheeks. Other features such as the facial lines or brow wrinkles appear with some expressions and disappear with the neutral face.

Regarding importance of features, in [32] different occlusions of face features are tested and it is learnt that the mouth provides more information than the eyebrows, and these two also provide more information than the rest of the face. In [34] it is shown that sadness is mainly expressed (or

detected) through the mouth, as it can be seen in figure 5. It is important to note, however, that this conclusion is obtained on a posed database, and may not hold true for spontaneous expressions.

Another research based on occlusions [35] shows that the occlusion of the mouth worsens the accuracy detecting anger, fear, happiness and sadness. On the other hand, covering the eyes region deteriorates the performance with disgust and surprise.

The features of interest and the value of each one have been discussed, and now the different approaches to extract them will be presented.

In [17] a comparison is presented with several publications from 2001 until 2010. Some of the most used methods for feature extraction in that comparison are the Gabor transform, the DNMF, the Candide model, and simply extracting points with the position of relevant features.

Gabor filters are generally used for texture representation and discrimination in which frequency and orientation is represented similar to how human beings see. For facial feature extraction, several Gabor filters at different orientations and spatial frequencies are used, and the outputs are fed into the classifier. In figure 6 we can see the results of training with this method in [37], and the different

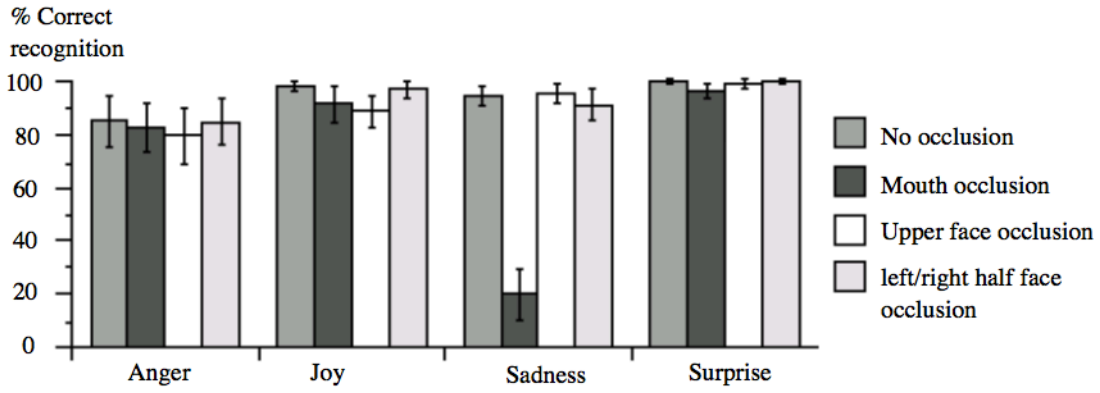


Fig 5. Effect of partial occlusion from [34]. Error bars show the 99% confidence intervals.

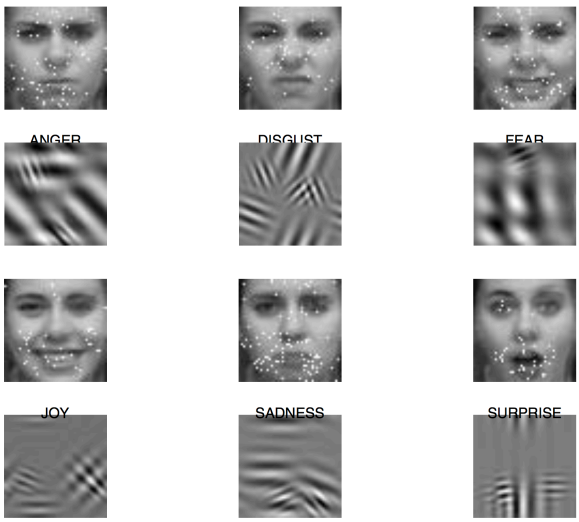


Fig 6. Example of Gabor method for feature extraction from [37].

emotions seem easily distinguishable in that example. However, in this method it is difficult to know what is occurring, as it is not intuitive to look at.

The DNMF approach is based on NMF (Non-Negative Matrix Factorization) that tries to approximate the facial expression image by a linear combination of a set of basis images. The DNMF method incorporates discriminant constraints to the weights matrix (which works as a projection to a lower dimensional space) and will be used in a minimization problem to train the classifier. In figure 7, an example of the DNMF implementation developed by [35] is shown. In this figure it can be seen that in most cases what the algorithm considers the most important are the eyes and mouth. This is coherent with what has been mentioned before, that several researchers have concluded.

The third method that is frequently used is the Candide method, through which a 3D model of the face is estimated. As it can be seen in figure 8, this is similar to extracting the position of certain features (landmark extraction), since there is a model of the face, and it locates certain positions in the face that will be provided to the classifier. Regarding the differences, in the 3D model the points

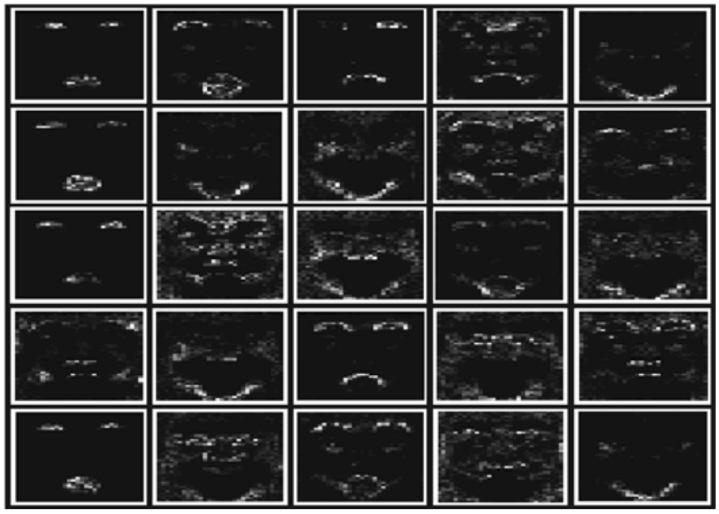


Fig 7. Example of DNMF method for feature extraction from [35].

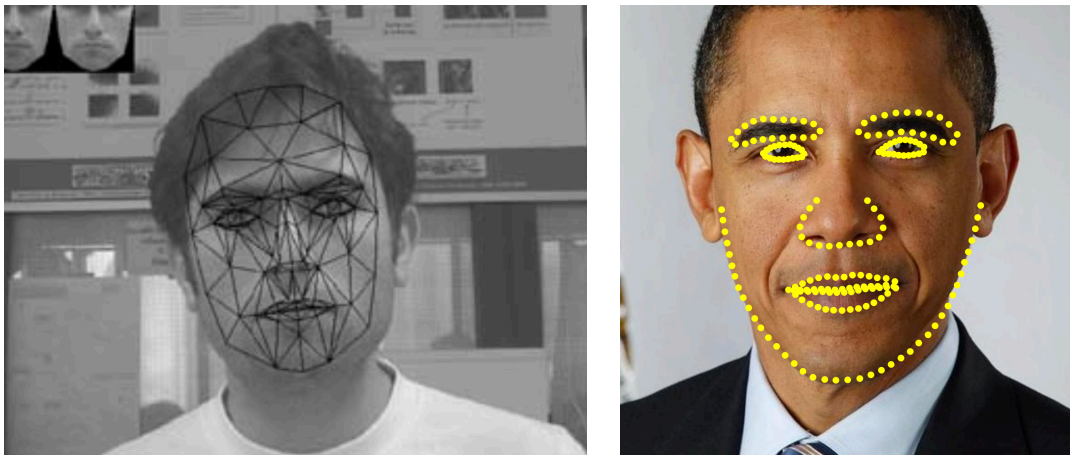


Fig 8. Candide 3D model from [38] on the left, landmark extraction from [13] on the right.

cover all the face, whereas the landmark extraction aims at certain points that are relevant and easy to extract due to the presence of abrupt changes.

Recently there has been a shift towards using Convolutional Neural Networks with the whole face and classifying in one step, as in object detection systems. Although these methods obtain very good results, they are still too expensive computationally for a smartphone and a real time application.

In the comparison there was not a method that seemed to be clearly superior in performance. It was decided to use the landmark extraction because it is easier to understand what the system is doing, and why it may fail or succeed. It was also desired to use a method that had already been implemented in a library and was fast computationally —so that it can run on Android and more or less in real time—, because there is a limited amount of time and resources available that have to be allotted to building the entire system. Implementing a feature extraction method would require an amount of time that would impede doing the rest of the tasks. The library used will be described next.

2.4.1. DLIB facial landmarks

The dlib [12] library is a machine learning library, but it is well known for its face landmark extraction. It is the implementation of a [13] using a larger database for training (the iBUG 300W dataset, which can be found in [48]), which places landmarks in key positions of a human face with an accuracy equivalent or superior to the state of the art, while processing each image in approximately one millisecond.

In [13] they identify the crucial components of other alignment algorithms include them in a cascade of regression functions learnt by gradient boosting. The regression functions in this cascade estimates the shape and intensities from an initial estimate. One reason for the good performance of the algorithm is the fact that it incorporates two key elements that have been used in other face alignment algorithms.

The first one is concerned with how pixel intensities are indexed relative to the current estimate of the shape. Accurate shape estimation using extracted features in vector representation is difficult because they greatly vary with shape deformation and other factors such as changes in lighting conditions. These extracted features are necessary to estimate the shape, but a good estimation of the shape is also necessary to extract features that are reliable. Given this relation between features and shape estimation, an iterative approach seems appropriate, and it is what some previous researchers and [13] use. The image is transformed to a normalized coordinate system based on the shape estimate, and then the features are extracted to predict a new shape. This iterative process is repeated until convergence is reached.

The second key element incorporated handles the difficult prediction problem. Finding the (high dimensional) vector that best agrees with the image data and the model is a non-convex problem with many local optima. As this is too expensive computationally, the number of potential shapes considered needs to be reduced to disregard most local optima. [13] uses two approaches that have been used in previous work to achieve this reduction. The first one consists in assuming the estimated shape will lie in a linear subspace, which for example can be discovered by finding the principal components of the training shapes. Then, the second approach uses the fact that a certain class of regressors produce predictions that lie in a linear subspace defined by the training shapes, so no additional constraints are needed.

Finally, another reason that contributes to the good functioning of this approach is their efficient regression function learning. An appropriate loss function is optimized and the feature selection is done in a data-driven manner. Regressors are learned through gradient boosting with a squared error loss function, which is the same function that will be minimized when testing.

The final system obtained from the previous developments has an accuracy comparable to the state of the art with a significantly shorter computation time.

In the dlib [12] library, a landmark extraction already trained is provided. The iBUG 300-W [14] dataset has been used for the training, and it contains more than 10,000 images annotated with the landmarks.

2.4.2. Landmark alternatives to dlib

One of the most popular libraries for landmark extraction is the dlib described before, for how fast it runs and its good accuracy. Nevertheless, there are other libraries that are also used such as STASM or CLM-Framework among others; but these are slower, less accurate or both.

In [15] we can see some interesting approaches that are probably not good enough yet, but might be the direction to follow for the next landmark detectors.

The idea behind these other methods is the same that has been mentioned for face extraction: processing video, the landmark extraction is analyzing each frame independently than other frames. This means that useful information is being ignored, potentially spending more computations and obtaining worse results.

According to [15], tracking landmarks from one frame to the next one is more robust to changes in pose or viewpoint, and for that reason they compare four tracking algorithms with dlib and another library called Deformable Shape Tracking (DEST) to find the best option for real-time in mobile devices. DEST will not be taken into account, however, because its face detector fails often, thus leading to incorrect landmark results. Although another advantage of tracking algorithms is that they can be faster, they are also more prone to drifting (error driven by statistical properties' changes, that can be induced by illumination or background changes).

The conclusion in [15] is that the LK tracker is more robust, has better overall accuracy and it is significantly faster. The tests, however, are done using an ideal landmark detector (ground truth) for initialization. This is critical, because if the LK tracker is initialized with the DLIB detector it does not outperform a standalone DLIB, since it cannot correct an error in the initialization. So, although it brings speed improvements, it is at the cost of accuracy in real conditions.

The implementation of LK is a sparse iterative version of Lucas-Kanade [16]. It is one of the most popular methods, which has been implemented in OpenCV and relies on optical flow for the tracking.

In conclusion, the main problem with tracking algorithms is that a bad initialization leads to poor results, whereas detecting on each frame only affects the frame that has bad results.

2.5. Classifiers

In this approach of emotion detection —and most of them— the classifier is one of the main blocks that conform the system, which receives the most relevant information from the feature extraction and processes it to decide what emotion is present, based on a previous training.

Classifiers typically used for this problem can be divided into static and dynamic. The former use information from a sample in a single time instance, whereas the latter use several frames to take into account the evolution in time. As it has been explained in the emotions section, facial expressions follow a set of stages that consist on onset, apex and offset. Dynamic classifiers exploit this information to help improve the results, but do not work or perform worse when some stages of the expression are missing, or when a single image is used. On the other hand, static classifiers do not have these disadvantages, also miss the information provided by time and perform poorly in transitions between emotions.

Cohen et al. covered some of these classifiers in [39]: a Naïve-Bayes with Cauchy distribution and a Tree Augmented Naïve Bayes for the static classifiers, and single and multi-level Hidden Markov Models for the dynamic ones. The Naïve-Bayes requires independency between samples which cannot be assumed in facial expressions, and for that reason they test the TAN that improves the results.

Regarding static versus dynamic classifiers, Cohen et al. conclude that static classifiers, which are easier to train and implement, can be unreliable on continuous video sequences, especially on those frames that are not on the apex of an emotion but the transition. For that reason, they believe that dynamic classifiers are better for person-dependent systems because they observe temporal dynamics; but they are more complex, require more samples for training and have more parameters to be tuned.

An interesting research that compares different classifiers is [40]. It includes Bayesian classifiers such as the NB and TAN mentioned above, decision tree inducers, an SVM with Vapnik's polynomial kernel, kNN —robust but slow— and others. They feed the classifiers with motion vectors and the one with best performance is the kNN (k=3), while the SVM has poor results. However, they also claim that the kNN is computationally slow and needs a lot of memory, which would be a bad option for a smartphone application.

In [41] they compare SVMs with an RBF kernel and a MLP with a single back propagation network. The MLPs reaches the best absolute recognition rate in their tests, but based on the ROC curves and statistical analysis they consider the SVM to be better.

In [17] a summary of the classifiers being used is presented and some of the most common static classifiers are SVMs and MLPs. These two were selected and will be further described afterwards. Many researchers have used SVMs achieving decent results, and are easier to understand. They are also available in most libraries, including the dlib library that is used in this work, and for the kind of problem to be solved the dlib map of classifiers recommends using the SVM [42].

2.5.1. SVM

One of the frequently used machine learning methods is the Support Vector Machine. The SVM is first trained through different methods with labeled data (supervised learning), and once the model has been built, predicting new samples is straightforward.

To understand how the basic SVM works, the labeled training samples can be represented in the space, such as in figure 9 on the left, and then an hyperplane that correctly separates those samples is found. As there may be many (infinite) hyperplanes that separate the samples, the one that is the furthest from the closest sample with each label is chosen, such as in figure 9 on the right. These images are simplifications, as this model is generally applied with a large number of

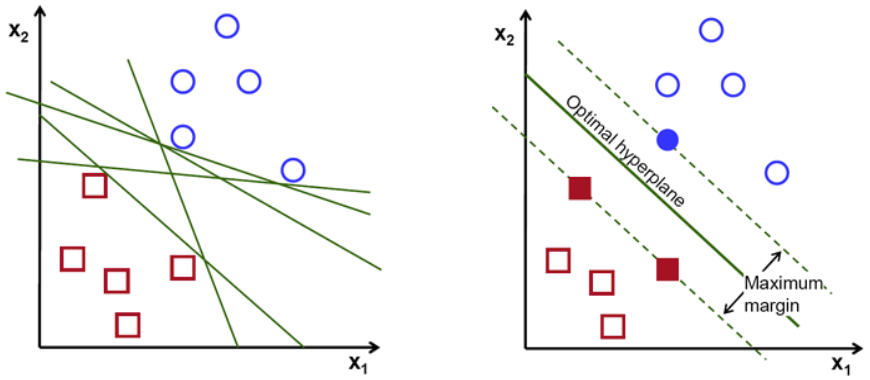


Fig 9. SVM training examples from [22]. Several possible separations on the left, maximum distance on the right.

dimensions. There may be complications, such as having no possible hyperplane that correctly isolates all the samples or having more than two different labels to recognize. Then, to know whether a new sample that needs to be classified belongs to a label or another, it simply needs to be located in the correct region of the space.

When not all samples are separable, there are methods to find the best hyperplane that correctly classifies most samples instead of all. But in some cases the samples are not linear and an hyperplane becomes a poor solution. The first approach intuitively might be to substitute the hyperplane and find other non-linear separators that can fit the training samples, but using an hyperplane is mathematically and computationally easy. For that reason, it is preferred translating the samples to a new space where an hyperplane can separate them —or most of them. This is achieved with the kernel method, that maps samples into a higher dimensional-space where an hyperplane may work better.

A common example to illustrate this is a set of samples in two dimensions with two labels and with the first class (samples with a common label) within a circle; then, the second class covers the samples that are out of that circle. In other words, two concentric circles. The ideal separation would obviously be a circle, but an hyperplane is needed for the SVM. If the samples are converted into a paraboloid —in the correct manner, of course—, an hyperplane will successfully divide the samples [23].

A common kernel is the Radial Basis Function kernel (RBF), which is also called Gaussian kernel because it is defined with the Gaussian exponential function. It has a gamma parameter that needs to be adjusted to the data samples, and it projects the n-dimensional space into an infinite-dimensional space [24].

Translating to a new space with higher dimensionality is the solution that has been presented to solve non-linear problems. This does not guarantee that the RBF will be the solution to all problems, but it is to many at least, and it is easy to adjust —it has one parameter, gamma. For these reasons, the RBF kernel is used in this project.

The kernel ridge regression is also used in the classifier of this program, which is the version of the ridge regression that is used with the kernel trick described before. This algorithm regularizes avoiding overfit, which can be achieved by penalizing the norm of the cost vector.

These characteristics described for SVM classifiers are used in a one versus one classifier. This consists on training N*(N-1)/2 binary classifiers, where N is the number of classes to be separated.

Then a voting scheme is applied to decide the class with the highest number of +1. It has ambiguities, as some regions of the input space can receive the same number of votes.

2.5.2. TensorFlow DNN

Artificial Neural Networks are mathematical models that learn through examples, following a structure that resembles biological neural networks (the brain) from animals. The initial intention was to mimic the behavior of the brain, but some functions were incorporated that differ from that behavior. An example is back-propagation, where information travels in the opposite direction and adjusts the network to reflect that information.

In this resemblance to the biological neural networks, ANN also have interconnected units called artificial neurons (parallel to axons in the brain). These units have inputs and outputs (connections) that are multiplied by weights, computed by a mathematical function, and sometimes thresholds. Modifying these weights and thresholds is how the desired output can be obtained from each input. However, given the size of these networks, assigning weights by hand is infeasible; instead, different learning or training mechanisms are used to do this automatically, with the goal of minimizing an error or cost function.

The structure consists of a first layer called input layer, with the same size as the number of inputs. Then, the last layer or output layer has the number of outputs as size, and the hidden layers are between the input and output. A particular case of ANNs are the deep neural networks (DNN), where there is more than one hidden layer allowing for modeling complex non-linear relationships. An advantage of these DNNs is that, once trained, they can be implemented with matrix operations, and an example of the structure in the case of two hidden layers can be seen in figure 10. The Tensorflow [20] example that will be described next is a DNN.

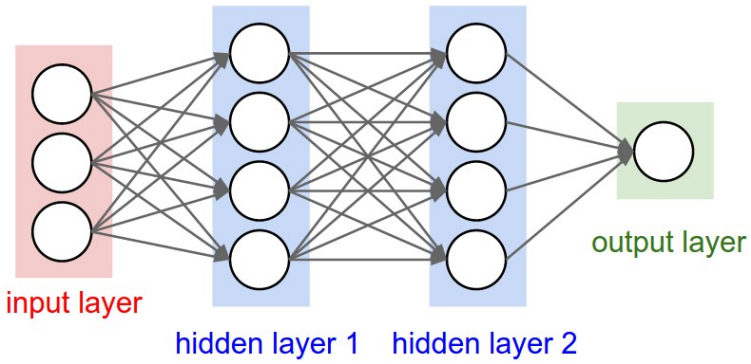


Fig 10. Structure of a deep neural network with two hidden layers. Extracted from [47].

Tensorflow offers machine learning tools that allow non-expert users to build classifiers easily. It is easy thanks to a library that has methods to do most tasks, tutorials and sample code that cover a wide range of cases. The DNN in the sample code, that will be used as a reference, uses an AdaGrad [21] optimizer.

The AdaGrad —Adaptive Gradient— optimizer consists on a family of sub-gradient methods that dynamically incorporate the knowledge of the geometry of the data observed previously to improve the gradient-based learning. It is based on the fact that data frequently has a high dimensionality, but a particular instance has most inputs to zero. However, when one of these inputs

that are most frequently at zero take a different value, this event is generally highly informative. So, this optimizer learns what features are more frequent than others and assigns lower learning rates to those that are more frequent. However, when an infrequent feature appears, the learner should adapt faster.

An activation function is also used to add non-linearities, the “tf.nn.relu”, that stands for rectified linear unit and sets to 0 values that are negative.

3. METHODOLOGY

With the state of the art explained in the previous section, the relevant methods to develop this project will be described here. Although the results will be occasionally mentioned, their full analysis will not be exposed in this section, but in the next one.

This project started with the intention of using an existing solution to the emotion recognition problem, and working on an evaluation and possible improvement for Android devices.

The Microsoft Azure emotion detection technology [1] was proposed to me as a possible solution to start with. Given the powerful brand offered by Microsoft, their tools were expected to be a good start; however, they did not fit this project. The first problem is that it is not a library, but an online API. The software that uses it sends the image to their servers, and the servers reply with the results of their analysis. Although Microsoft claims that it can achieve almost real-time by sending some frames of the video, that will only be if the user has a really good internet connection, and would consume data plans if not over WiFi. The other problem is that the tool does not offer any changes to improve the performance or adapt it to specific problems, so it does not suit the purpose of this project. Furthermore, after a certain amount of server requests a payment is required to continue using the API.

Given these restrictions, it was necessary to find an alternative that could be downloaded as a library and also worked on Android devices. The alternative that was found is Affectiva [2], which has both an online API and an offline library. It distinguishes between seven emotions: anger, contempt, disgust, fear, joy, sadness and surprise; and also offers other tools. For example engagement —how expressive the subject is—, facial expressions —attention or head orientation, brow furrow, cheek raise, etc— or appearance estimations —i.e. age, ethnicity, gender and whether glasses are being used.

To estimate the emotions, they rely on the facial expressions they manage to detect. Then, they have a mapping between emotions and expressions, and all expressions detected increase or decrease the likelihood of each of the emotions. This mapping they are using is based on the FACS [31] that has been described in the State of the Art section.

Through inspection of the documentation it was assessed that although the library works offline, it does not allow customization; only enabling or disabling features. However, it could be useful to test how accurate the results were, so a simple application using the SDK was built.

Given that they provide samples and that the library is not complex, it did not require a lot of time. A button is set to toggle between the front and back camera, which calls a method of the library that does all the work required. Another button starts and stops the camera and the system and a ‘SurfaceView’ displays the images that are being recorded by the camera. Finally the detector from the Affectiva library on each frame returns a list of the detected faces with the information requested, of the confidence level for each emotion. If there is no face a ‘TextView’ displays “NO FACE” and if there is at least one face, the confidence of each emotion is displayed. For visualization purposes, the emotions with higher than 20% confidence levels are displayed in green, and those with a lower value are displayed in red.

The features of ‘ethnicity’ and ‘age’ were briefly tested and return extremely poor results that actually seemed random guessing, even with good lighting conditions. However, it was only tested with one subject as this is not the goal of this project.

Regarding emotions, the results were better, but still not as good as it would be expected from a product that has been released to the public as complete.

Given that no good option was found that could be used and modified to build an Android app to detect emotions it was decided to build the whole system to obtain the images from the camera, process them and decide the emotion, all while being compatible with the Android API and the limited computational resources and battery.

With the amount of hours available for this project, and the fact that neither the author nor the advisors are experts in the field, the system is built as an initial approach that can be potentially improved by future researchers, without aiming at obtaining better results than the state of the art. Current projects in the field are developed through longer periods of time, and generally by researchers with years of expertise. Instead, the goal is to provide a structure and proof of concept of an emotion recognition application, that tries to obtain the best results given the limitations described.

The structure of the system, as it can be seen in figure 11, has a first step that extracts faces from the current frame captured by the camera. These faces are the input of the next step, that performs the feature extraction. This feature extraction consists of obtaining the information that can be useful to determine the emotion, from each of the faces, such as the aperture of the eyes or the shape of the lips. This information or features are then fed to the classifier, which transforms the feature information into emotion information.

Fig 11. Structure of the system. This structure is similar to what most researchers do.

3.1. Hardware

When the system is run on a smartphone, the smartphone in which it is tested is a Nexus 5, which is almost 4 years old at the time of presenting this work. Therefore, its computational power is fairly limited.

To build the database most of the videos have been recorded with this same device. However, two subjects were recorded with a Moto G smartphone, which has a front camera with worse characteristics than the Nexus 5, and also records at a lower frame rate.

3.2. Databases

Given that the next steps rely on the use of face databases, the comparison and decisions taken regarding database selection will be presented next.

For the first training and tests of the system it was decided to work with a database that contained static images, so that it could work with only one image instead of requiring a video.

Because of the limited time and resources it was decided to limit the project scope to frontal-facing subjects, without occlusions and without paying attention to different illumination conditions. The JAFFEE database [44] was immediately discarded because it only contained Japanese subject, and the FERG [45] one for being made of computer-generated drawings. The AR and CMU-PIE were also set aside because they focused on occlusions and illumination.

Between the Cohn-Kanade and the Radboud, both seemed to be suitable, and the latter was selected because it was labeled with emotions, whereas the former was labeled with AUs and required extra work for the conversion from AUs to emotions. Also, the fact of having two ethnicities and also kids seemed positive to obtain a training that was more generalized; however, later it was observed that the database is probably not large enough to generalize, and no real attempts have been done to obtain a more general training because this requires a significant amount of hours, collecting, merging and adapting multiple databases.

With respect to video databases, the initial intention was to use a video database after the first tests had been done with the pictures one. This would allow to add more images to the training, and also to develop a system that would take into account previous samples to aid in the prediction for the current sample. However, as it was requested to test the system with subjects from the laboratory group and to try to include them in the training, the use of a video database was substituted by building a simpler one with known subjects.

As a summary, initially the Radboud database was used, and later on a small database with videos was built and also incorporated. How this small database has been built will be described later in this section.

3.3. Face detection

This first component of the system has the goal of capturing all the faces that are visible in a picture or video frame, assuming a reasonable size. Then it obtains rectangular boxes that comprise the faces, and sends these boxes to the next step.

The first version of the face detector was based on the Viola-Jones [18] described in the state of the art, using the implementation from OpenCV. It is one of the most used approaches and it was easy to use because it is implemented in the library.

After using the ‘dlib’ library for the landmarks, its face detector based on the Histogram of Oriented Gradients [19] —also described in the State of the art— substituted the OpenCV approach. It was equally simple to use and offered better results, allegedly.

As it has been explained in the State of the Art, there are better approaches to face detection. However, as this project’s goal is to build a prototype instead of a final product, the focus is on emotion recognition and these detectors have good enough accuracy, other options have not been tested. In the future, substituting this block would be easy and allow for higher accuracy or less computation time, depending on the method selected.

Also, another interesting improvement for the future is adding a tracking algorithm so that not all frames need to be detected, and instead an algorithm with shorter computation time speeds up the system, as it has been presented in the state of the art.

3.4. Feature extraction

This block of the system receives a rectangle that contains the face of a subject, occupying the whole rectangle and cut from the original image or video frame. Since this rectangle contains a lot of information, having a classifier trained with all this information will be computationally too expensive for a real-time application, especially if it is run on a smartphone. Therefore, the goal of the feature extraction is to collect the most important information in the face (regarding emotions) in an efficient manner, time and computation wise. This information has to be extracted and formatted so that a classifier can work with it without requiring too much time or complexity. In this case, an array of 136 float values has been chosen, and it could even be reduced.

As we have seen in the state of the art, the mouth and eyebrows are the most relevant parts of the face for detecting emotions, and for that reason these are the main features being used. However, the image contains more useful information that would contribute to improve the classification. Some features that could have been extracted without adding a lot of complexity to the classifier might be the wrinkles between the eyebrows, or the wrinkles next to the eye. But this would require converting the amount of wrinkles into a single value, and as it deviated from the goal of the project it is left for possible future improvements.

As it has been explained in section 2.4.1, the ‘dlib’ library contains a machine learning based algorithm that has been developed to extract face landmarks while running extremely fast, and offering reliable results. As a simple test to verify that the results are accurate, 200 images from the database were tested where only three small errors were found. Two of these errors were the lips a bit out of place, and the third was the position of the eyebrows that also was a bit moved.

As we can see in the figure 12, it detects the contour of the face, as well as the nose, eyes, eyebrows, and lips. Since this library runs in C++ and is also available for android (NDK) it is a very good option for this project, as it also obtains the most relevant information for the emotion classification —i.e. the mouth, eyes and eyebrows.

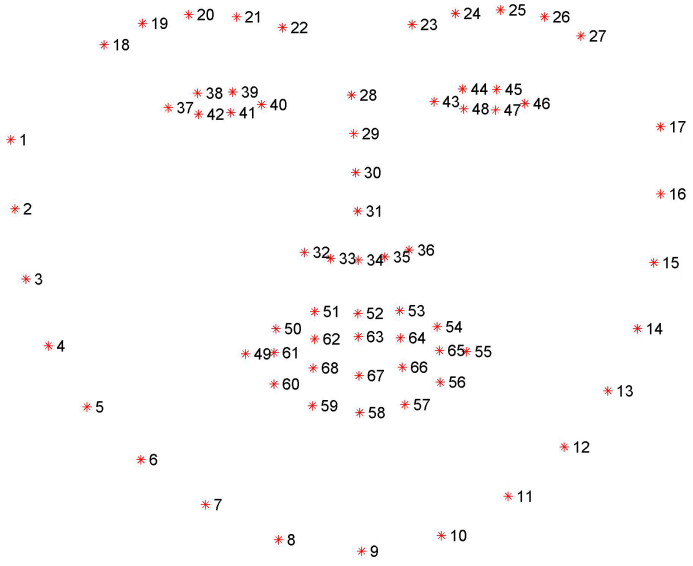


Fig 12. Landmarks extracted by dlib with the corresponding IDs. Extracted from [48].

The algorithm, based on a shape predictor, returns the 'x' and 'y' coordinates of each point in the face image, where each point has the ID that we can see in figure 12. Having these IDs consistent is also quite useful, as it will be shown afterwards.

All faces are different in shape, aspect ratio, and several other characteristics, and the system needs to be able to ignore these differences so that when it predicts the emotion in a new subject who has his or her own different characteristics, the emotion predictor should be able to function without further training. In other words, it should be able to generalize as much as possible.

One step to help with the generalization is to provide normalized values for the landmarks, instead of a value of the pixel position. So, the normalization ensures that all values will be between -1 and +1 for both the 'x' and 'y' axis, by dividing with the furthest point from the center —tip of the nose. This normalization is done independently in each axis, so that the aspect ratio —i.e. height and width— of the face is similar for all subjects, as seen by the classifier.

And in this step the consistent IDs become useful (see figure 12 for identifying the following IDs). First, because the center of the face is the point number 31[1], which is more general than using the average of all points. Then, the right most point will be the maximum 'x' of points number 15, 16, 17 or 27. For the left most point it is only necessary to examine points 1, 2, 3 and 18, and obtain the minimum 'x'. Finally, the top will be the minimum 'y' value —the origin is at the top of the image— of points 1, 17, 19, 20, 21, 24, 25 and 26; and the bottom will be the maximum 'y' of points 7, 8, 9, 10 and 11. So, to normalize horizontally, we divide the 'x' value of each point by the maximum of the absolute value of the difference between each extreme —left most and right most values— and the center —tip of the nose. To normalize vertically we apply the same computation, but using the top and bottom extremes.

Therefore, the next block of the system, the classifier, will receive each point respect a new origin —the tip of the nose— and normalized between -1 and +1. But this is not enough for generalization, at least for the real-time video processing. Although all the faces in the database are front-facing and straight, in video it is common to tilt the head, so that it appears rotated in the two-dimension frame; or to keep it still but with a bias to one side. Therefore, it is also necessary to compute the angle and apply a rotation to each point.

The angle is computed using average of points 22 and 23 for the top of the face reference, and point number 9 for the bottom. Then the points are rotated using the formulae:

$$x' = x \cdot \cos(\theta) - y \cdot \sin(\theta)$$

$$y' = x \cdot \sin(\theta) + y \cdot \cos(\theta)$$

It is important to rotate the points before applying the normalization; otherwise the normalization may be incorrect and it might deform the face —and therefore, the position of features. The positions of the landmarks and result of the normalization and rotation process can be seen in figure 13.

When this block is run in the whole system, for a real-time application, all the values are placed together on a vector. However, for training purposes, these are written in a text file, so that they can be read multiple times by different training attempts.

[1] IDs of the different points start in 1 in figure 12. In the code, however, the first ID has the index 0, so this difference needs to be considered when reviewing the code.

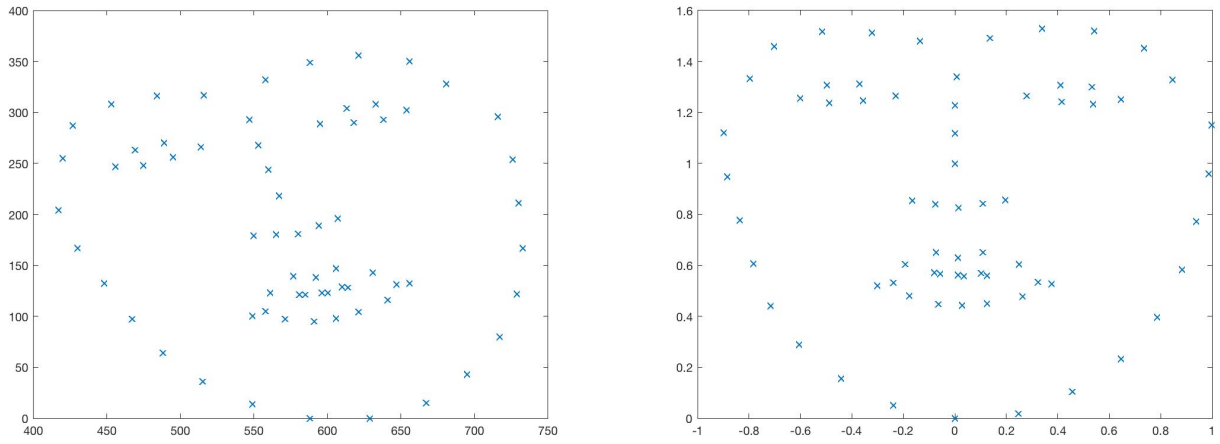


Fig 13. Landmark positions in pixels on the left, result of rotating and normalizing on the right.

3.5. Classifiers

The goal of the classifier is to process some information, the observation, and generate an output that tries to identify or classify this observation into a category or class. The basic classifier in this particular system receives as features —the information— a set of 68 points in a two dimensional space, i.e., a vector with a length of 136. This is the vector obtained in the feature extraction, that consists of different points of the subject's face, after normalization and other modifications.

The classes in this basic classifier are the different emotions, represented with an integer from 1 to 8 that corresponds to angry, contemptuous, disgusted, fearful, happy, neutral, sad, and surprised respectively.

3.5.1. Binary classifiers

As a first attempt, it was decided to try a simple binary classifier to see how it behaves with the database, given that this case is easier than trying to separate the 8 emotions at a time. This attempt, however, is not just a useless approach for the system, as we will see later.

The binary classifier is built with C++, using the same 'dlib' library used for the feature extraction. Since 'dlib' is a machine learning library, using it for the classification allowed building the whole system with only one library, which is more convenient.

A 'typedef' matrix of doubles is declared to contain the samples, with 136 rows —two for each point, to represent the 'x' and 'y' axes. Another 'typedef' is declared with the kernel type, in this case a radial basis kernel. Then two vectors of type "sample" are declared, one for the training set and another for the testing; and two vectors of type double are also declared for the labels.

Given that the images of each emotion are stored in a different folder, the label is chosen according to which folder is being read. So, as the classifier is binary, the desired emotion will have one label (-1) and all other emotions will have another (1).

With the sample sets already built and before starting the training, we reorder the samples randomly, using a method provided by the library. The maximum value for 'nu' is obtained, also through a method in the library that checks the ratio between +1 and -1 labels in the samples.

The classifier relies on two parameters that define its behavior, the 'nu' mentioned before, and the 'gamma'. To find appropriate values a grid search is used that starts with initial values for each parameter increments it by multiplying by 5 on each iteration with two nested '*for's*'. On each iteration a three-fold cross validation is used to assess the goodness of the parameters, and the best results are chosen. These best results are selected giving the same cost to positive and negative errors.

This approach for optimizing the parameter selection is not necessarily the one that will provide the best results, it is simply a method that is fast and yields decent enough results. So, with these parameters each classifier is built, and then evaluated on the test database set. The results will be presented on the appropriate section.

The whole process is repeated for each emotion.

3.5.2. All emotions SVM

Once the binary had been tested and produced reliable results, building a classifier to separate all emotions seemed appropriate. To build the basic classifier that separates all eight emotions, there were several options. As it has been explained in the state of the art, according to the literature and the resources and scope of this work, using a SVM seemed the most appropriate machine learning tool for the problem. Based on that, and 'dlib's own recommendations, the most reasonable options were using a multiclass linear trainer, or a Kernel Ridge Regression one versus one trainer, with a radial basis kernel (which has been briefly described in the state of the art). Ideally, both approaches should be compared, and potentially other options too. However, given the limited time and the fact that the latter was easier to implement, the former was not used and will be left for future work.

Regarding the data used for training, the features are read from a text file and assigned different labels according to the folder in which they are contained. However, instead of using -1 and +1 values, the labels will go from 1 to 8, one for each emotion.

About 20% of the database is separated for testing; however, this requires some considerations. First, the same subjects are selected as testing samples for all emotions, otherwise the subject in testing would not be "unknown" to the classifier, and the results might be better than they should be. And second, the particular database being used has 3 images of each subject and emotion where the eyes look at different directions. Given that these images are almost equal, and especially considering that no feature considers the direction at which the eyes are looking, there is a high dependence between these images. So again, the images need to be in groups of three and not separated to avoid having a test where the subjects are known to the classifier.

To build this classifier the same matrix 'typedef' is used for samples as in the binary classifier, with 136 rows and 1 column. Also, the same four vectors store the samples and labels for the training and for the test sets.

Two more 'typedefs' define the trainer of the type "one versus one trainer" and the kernel, of type "Radial Basis". This RBF depends on the parameter 'gamma', that will be optimized in a similar fashion as for the binary classifier. The values will be tested iteratively and a cross validation function will return the results of a 5-fold validation. The parameter value with best results will be used for the actual classifier.

The SVM is built with a KRR one versus one trainer using the RBF kernel with the optimized gamma parameter and the accuracy is computed for each emotion, dividing correct predictions of

each emotion by total samples of such emotion. This is done, besides the global accuracy, to verify that there is no emotion that is having significantly worse results than other emotions. Finally, the training is stored in a file so that it can be reused by the real-time desktop system or by the Android app.

3.5.3. All emotions deep neural network

As it was proposed by the advisors, an alternative classifier based on neural networks was also built in order to be compared with the SVM described in the previous step, and potentially be used in the final system. Again, with no expertise in neural networks and a limited time a simple network was built, using three layers, with the sizes being double than the amount of inputs in the first layer, four times the amount in the second and again double the amount in the third one. It is a simple approach that should yield approximately an acceptable network for having an estimate of the accuracy that can be potentially achieved.

Another of the benefits of the neural network is that using it to classify, once trained, requires simple matrix products and additions, so it can be implemented without difficulty. Also, given a network that is not too deep such as the three layers used in this case, the speed of computation is not excessive. This speed allows implementing the neural network on a real time application in a device with limited resources.

To train and build the Deep Neural Network it was decided to use TensorFlow. Google has provided tools and examples that allow non-experts to build a first DNN in a very short time, and for that reason it was a suitable tool for the task.

In the case of the neural network, the data format is modified to be adjusted to how it is used in the example. Two CSV files are created —one for training and one for test—, where each line represents a sample with the values separated by comas, and the last column contains the label. To separate between training and test it is done with the same considerations and exact same division as in the previous SVM case.

The code is based in Google's example, which is in python. The constructor provides the sizes of the layers, the number of classes and where to store the model. This location is important because as long as there is a file there, the training will improve the current file. Therefore, when performing a new training it is critical to remove the previous training or the results will not be what they are expected to be. The amount of steps is set when doing the training, and a value of around 10000 steps seems to be adequate for this problem, to avoid overfitting. To evaluate the results, a confusion matrix is printed comparing the test labels —'y'— with the predictions of the test set.

Both the training and the predictions are done automatically through methods provided by Tensorflow, according to the parameters decided. However, if this network needs to be used in a different program, such as the Android app, the matrix values can be obtained, and implemented as a series of products. This would take some time, but not too much.

The results were similar to those of the SVM approach, and for convenience the favored method was the SVM. First, as the 'dlib' library implements it and already needs to be installed for the feature extraction, it requires less time to build it in Android. Second, given the nature of how how each system works, doing changes to compare different approaches is significantly faster with the SVM. This is exacerbated by the difference in knowledge programming in C++ and python by the author.

3.6. Whole system

Once the different blocks of the system have been developed, it is time to see how they work together, and also verify that they can work in real time.

To build the complete system, the process is similar to what has been described in the previous sections. The matrix of samples, the trainer and the kernel are defined and then the frontal face detector, the shape predictor for the face landmarks and the SVM classifier are loaded from the respective files. An openCV video capture is opened to record images from the camera in the computer and a while loop obtains the frames in each iteration.

These frames are stored in an openCV 'mat' and flipped so that the viewer or subject sees it as a mirror, to avoid confusion with left and right. So that the 'dlib' library can work with the openCV 'mat', it is wrapped using a 'dlib' method —which does not copy the image.

The frontal face detector finds the faces in the current frame, and the landmark detector is run on the located faces. For doing the tests only one face is being analyzed, but since all the faces are detected by the face detector analyzing the other faces only requires adding a 'for' loop. Also, it is important to verify that at least one face has been retrieved, or the program will crash from trying to access an empty array.

The features or points obtained by the landmark detector are rotated, translated to have the origin at the tip of the nose, and normalized to have all values between -1 and +1, as before. An array with the processed features is built and fed to the classifier, that predicts the emotion in the current frame. This prediction is printed on screen.

Finally, as a last step done for visualization, the frame is shown with the landmarks printed on top.

3.7. Android app

One of the reasons for using the 'dlib' library was that it has been ported to Android, and it has a sample project that uses the landmark detection functionality. This simplifies the process of writing the Android application, that otherwise would have been too complicated. Hence, the work necessary to port the system to Android were mostly the classification block, since the first two blocks only needed minor adjustments. Of course, after understanding how the structure of the sample project works and learning how to develop with the NDK (native android development).

Generally Android applications are written using Java, and some parts —the user interfaces— with XML. However, in some projects, part of the code is written in C or C++ —native development. This generally occurs in those projects that are either computationally expensive and have speed requirements, or where an important part of the code has already been written in C/C++ in form of libraries that are not available in Java. So that these portions of the apps can be written in native code some tools, called Android NDK, are necessary.

In native development the structure of the application is generally written in Java. Then some native code is either written in separate files or imported as a library, and the Java Native Interface framework (JNI) allows the Java code to call native functions. This last part, the communication between Java and native code, requires some time reading the documentation and understanding how it works, as it is not straightforward.

This application has been designed with the native code built outside of the project and compiled into a library. The face detector, landmark detector and other 'dlib' library utilities are built in these C++ files that are external to the Android application. However, they cannot be completely independent, as there is communication needed between Java and C++, and this requires some methods in the C++ that have knowledge of the structure of the Android project.

Basically, each object —e.g. FaceDetector, EmotionDetector, etc— has a Java class that calls the C++ code, and a C++ file that manages the communication and object conversion. Then, a headers file contains the actual algorithms and it is called from the different files associated to the objects mentioned before.

For example, the emotion detection class —named SVM8 in Java— depends on the file that contains the training of the machine learning algorithm. The path to this file is obtained through Java code in the constructor and passed to the constructor of the C++ class by means of a native method. This method in C++ requires a specific nomenclature, and certain parameters, and receives an environment object and a '*jstring*', because C++ does not understand Java objects. In order to use this Java string in native code it needs to be converted into a C++ string, which is done using methods from the NDK tools. After this conversion, the classifier can be initialized loading the file from the path provided.

The other relevant method is the "detect" one. Given an array of double values —the points provided by the landmark detector— the method has to process the values and predict the emotion. In the Android application the processing of the points (rotation and normalization) is performed on the classifier because the landmark detector was already built in the 'dlib' sample. This way the landmark detector could remain as it was, and the processing be done in the new classification class.

The "detect" method in C++ receive a '*jdoubleArray*' object as input, which also needs to be converted. In this case the process is slightly more complex, and requires obtaining the array as well as its length. Then it is passed to the code that normalizes and stores it in a C++ data variable, and it is released. Finally, now that the data has been updated, the method that performs the classification is called and a '*jdouble*' object is returned with the value.

Regarding the structure of the project the MainActivity contains the views setup, manages the permissions, and leads to the Camera activity on click of a button. There, a fragment shows the images from the camera in a TextureView —with the appropriate size, rotation and mirror— and displays the results in text. A small floating preview shows the frames analyzed by the landmark detector with the points extracted, updating at a slower frame rate. Finally, a listener is set up, which will be called each time a new frame is available.

In this listener, for each frame, the image is converted from 'YUV' to ARGB and sent to the landmark detector. This landmark detector has a similar structure to the emotion classifier, where it calls the methods in native code. The results from the landmark detection are drawn to the preview on top of the face and then are provided to the classifier to obtain a prediction. This prediction is updated in the TextView so that the user can read it.

3.8. Database app

As subjects from the laboratory can do a live test of the system, it was decided to build a small database with them. This would allow a comparison between using these users in the training database and not using them, potentially improving the results.

This is also interesting because the training done with only the database with still images, (the Radboud [8]) was not complete enough so that the classifier can generalize for unknown faces, as it will be further discussed in the results. The results without the subjects in the training set will be a worst-case scenario, in terms of generalizing; and the results with the subjects in the training set will be a best-case scenario.

Recording several minutes of video at 30 frames per second, or even at a slower rate, would require a significant amount of time and resources in labelling. And taking into account the possibility of this project being continued, extending the database is a likely step.

To make this process more scalable and more pleasant, it was decided to build an application for Android that would record video and label frames automatically. Given that there is open source code [49] for the video recording and storing in Android, the time requirement is reduced, making this approach feasible.

Since there is no system for labelling emotions that is good enough to be trusted, the label is decided and given to the subject so that the emotion is posed. A real emotion that has been induced would be a better approach but again, this requires more resources, time and expertise than those available, as it has been described in the state of the art.

So, the application was designed to have green (for contrast) text in the top left of the screen, so that it is close to the camera, showing which emotion should be played by the subject. Then, a button to start and most of the screen with the image captured by the camera.

Three sequences of emotions are prerecorded, and one is chosen randomly, so that not all subjects and not all videos for each subject are doing the same. This is written so that it can be easily extended or modified.

In order to have a recording that was not too long, and for simplicity for both the subjects and the classifier, only some emotions were used: neutral, sad, happy and surprised; all appearing twice in the sequence.

When a new emotion is requested, the user requires some time to process the message and make the transition into the new emotion. Given this delay, and the fact that transitions between emotions are not taken into account by this system, an interval of 1.5 seconds is labelled as “no emotion” in each transition. This interval is followed by 3 more seconds where the emotion is actually recorded, adding up to a total of 4.5 seconds.

There is a label for each frame, and all the labels consist of an integer that is associated to an emotion. These are stored in a text file separated by spaces, with the same name as the video —the absolute time in milliseconds. This way it is easy to read by the training code, and the dubious frames are discarded.

The final database built consists of six subjects, five of which are male. Four subjects are caucasian and the two remaining are asian. It is also important to note that two subjects show expressions that are more similar to the prototype that can be seen in Radboud and other posed databases, and the other four subjects present expressions that are more difficult to distinguish. All videos are recorded on the frontal camera of a handheld smartphone —Nexus 5— except for two that are recorded on a Moto G and show significantly worse quality and half the frame rate. Finally, two videos —one on the Moto device— have the source of light behind the subject, which also worsens the quality of the image.

As a summary, the database, although small, has variability that will make the results worse than if all conditions were perfect.

3.9. System modifications

To try to improve the results, and test the effect of some changes, the previously described system was slightly modified, also using the videos in the database with subjects of the laboratory.

Building a complete database with all emotions is significantly time consuming. Furthermore, some subjects do not know how to pose for certain emotions, and no AUs expert was available to teach and verify the poses. For that reason, and to simplify the testing, the subjects only performed a smaller set of emotions: happy, sad, surprised and neutral.

The first system modification was to add samples from this database to the training. Given that no videos were used in the previous training, and that the database with static images was not large enough, it was decided to have the test subjects in the train database, but not the same videos.

Therefore, the code to analyze the videos had to be added to the training code. This works similarly to how it has been described in “whole system”. The ‘VideoCapture’ from openCV reads the video —instead of the camera— frame by frame and stores it in a ‘Mat’ to be processed. As the video is recorded at a higher frame rate that the real-time system can work, only one frame of every ’N’ is analyzed. This ’N’ value can be changed, but it has been set to 3, which are still more frames than the real-time system in order to have more training frames. Then the frames are processed as usual, by finding the rectangle that encloses the face, extracting the features, rotating and normalizing them, and adding the final values to the training or test set.

As the labels are stored in separate text files, these also need to be read along the videos. Through this reading, the values are stored in a vector, where each position represents a frame. Each time a frame is read, the corresponding label is read. If the label equals -1, it is a transition frame and therefore, that frame is not added to any database. On the other hand, if the label does not equal -1, both the values extracted from the frame and the label are stored in the appropriate set.

As each user produced two videos, one is added to the training set and the other to the test set. However, some tests are also done adding all videos to the training set and half to the test set, to see how the classifier performs. In those instances, the accuracy is almost 100%.

As it was decided to test less emotions, and the built database only contains happy, sad, surprised and neutral faces, some tests have been done only with these emotions. In these tests, the images from the static database that correspond to an emotion that is not in the list of four emotions are not added to the training. This improves the results, as it will be seen in the results.

Finally, another training that changes how the system works was introduced. The reason behind this change is that some literature uses the neutral face of the subject in one way or another, to help the system. There are several reasons that lead to believe this may help build a system with better results. For example, typically Asian subjects will open more their eyes, than Europeans, or some subjects have a neutral face that seems sad.

So, the approach consists on using a neutral face from each user in the database, and this is done through subtraction. The idea is that the points obtained from each frame or image are subtracted from the points obtained from the first neutral face that appeared for that particular subject. This way, we obtain the vector of movement for each point from the neutral state to the current face, which is analogous to some of the AUs (how the muscles in the face affect the shape, in each emotion).

For the static image database, each subject has three pictures for each emotion. The first of these is subtracted to the other two, as well as to all the pictures from non-neutral emotions for that

subject. Therefore, there will be a an entry with all values to zero for each subject. For the video database, the first frame is used as neutral (all videos start with a neutral face), and that frame is subtracted to all other frames from that user.

This helps in the generalization but also has drawbacks. The final system will require either that the user starts with a neutral face (which is generally true, but not always) or that the system waits until a neutral face is detected in some way to work. Also, it will be necessary to perform face tracking or recognition in the case of analyzing multiple faces.

This is also one of the ways to obtain the AUs (explained in the state of the art) that are critical for a good system. Another way is to build a system that takes into account several samples adjacent in time.

3.10. Testing

The different classifiers need to be tested to compare their performance. In order to do this, all the images and frames in the database are processed by the steps of the system prior to the emotion classifier and the values obtained are assigned to the label and stored, some in the training dataset and some in the test dataset.

When testing, the classifier is set up and it analyzes all the images isolating for this purpose. The results are compared with the label and different counters are incremented when appropriate. Counters are used both for a specific video and for the overall performance, and in each of these two categories there are also counters for each emotion and for the overall results. Both the correct predictions and the total number of frames in each category are added up, and the accuracy is then computed by means of a division.

To know what frames correspond to a video or another, the number of frames that have been processed and added to the test dataset for each video have been recorded, so it is only a matter of counting the number of frames tested and when it reaches the amount of frames from one video, it means that the next frame will correspond to the next one.

3.11. Challenges

Some challenges have been encountered during the development of this project. The main one is that emotion extraction is an extremely complex topic to work with, with an enormous amount of publications. Covering all of these publications was not possible.

Another difficulty was the unfamiliarity with computer vision and machine learning. Despite having a background in signal processing, having more knowledge in these two fields would have been helpful.

Finally, another challenge to overcome was the lack of experience with most tools required. It was not a total impediment, but it required significant time. These unfamiliar tools were Python and especially C++, regarding programming languages used in the project. And then, OpenCV, dlib and the Android NDK were also unknown.

4. RESULTS

In this section the results of the different experiments will be presented. But first, some points about how this is done will be made.

As it has been discussed in the state of the art, emotions generally produce facial expressions that go through three stages: onset, apex and offset. The apex consists of a constant facial expression, and the other two stages are the transitions between a facial expression and another. Dynamic classifiers take into account the temporal patterns of facial expressions, thus being capable of perceiving those transitions. On the other hand, static classifiers such as the one used in this work do not have this information. Generally static classifiers are tested on the apex of emotions instead of through all the stages, and this is what will be done in the results presented: transitions will not be considered.

In a complete system capable of detecting emotions in real time this approach would yield poor results each time there is a transition. For that reason it is necessary to have either a dynamic system or a training that takes into account transitions besides emotions. In the future development section, these options will be further discussed.

Another limitation is that only frontal facing subjects will be evaluated. For predicting emotions on subjects that look 45 degrees away from the camera or 90 degrees away, more training and images would be required. As it is similar to the frontal faces but requires more time, it was considered preferable to occupy the available time on performing more tests in this one case than trying to take all angles into account. With the landmarks extracted it should also be possible to estimate the pose of the subject and use different classifiers accordingly.

Finally, all the images taken into account are of posed expressions. Spontaneous expressions would be preferable, but there are few databases with spontaneous images. Recognizing spontaneous expressions is also more complex, and building a spontaneous database is not possible with the available resources.

4.1. Feature extraction

The feature extraction consists of obtaining the position of 68 points or landmarks on certain positions in the face of the subject, such as the mouth, nose, eyes, etc. This library has been selected for several reasons, and one of them was that it is considered an accurate landmark extraction library. For that reason, it was not thoroughly tested, only enough to verify that it is actually reliable.

The short test that was performed consisted in manually reviewing the landmarks placed in approximately 200 images of the Radboud database. Of these images tested only three errors were found, involving the position of the lips in two cases, and the position of the eyebrows in the third. These mistakes in the positions mean about 5 milimeters of displacement in the real world, and may affect the results or the training of the classifier. Nevertheless, a 1.5% images with errors, where the error does not involve all the points but only some seemed good enough.

On the database built with subjects from the laboratory there is one subject who is not looking completely straight to the camera, but a bit to the side. In this case the detection of the end of the face was significantly out of place in several frames. Although the outer part of the face should not

directly affect the classifier when deciding between emotions, the fact that it is used for the normalization may produce mistakes. In another subject the position of the chin is sometimes incorrect too. These two cases are subjects filmed on a smartphone with a considerably worse camera, worse lighting conditions and also worse performance of the autofocus.

In some other frames, also, there were errors in the position of the eyebrows. This typically occurs when the subject is surprised and has the eyebrows higher than in the resting position. The landmark extractor probably considers that the eyebrows are too high and decides it is lower than it actually is. This mistake is probably troublesome, because in the surprised expression the eyebrows are critical for detection of the emotion, especially on subjects that do not open their mouth.

The camera and lightning conditions are significantly worse in these tests, as they are performed with smartphones that do not have good front cameras, and the lighting conditions are far from ideal. This is probably part of the cause of worse results than expected.

4.2. Binary classifier

The binary classifier was initially built as a first test of the possibilities of the SVM as a classifier to detect emotions, but later it has received another use: to detect neutral faces as an initial training of the system for a new user.

In table 1 the results of the binary classifiers can be seen for each emotion and for both training and test sets, where approximately 15% of the database has been used for testing. The train accuracy values consist of the percentage of correct predictions where the image does not correspond to the current emotion for column ‘no’, in a 3-fold cross validation. Column ‘yes’, on the other hand, has the accuracy for those images that show the present emotion. The test accuracy is the same, but obtained from the test set. Finally, the ‘gamma’ and ‘nu’ values are the value found to be optimum for these parameters. A simple grid search was used, so probably better values could be found. However, doing a better optimization was not worthwhile and it could also lead to overtraining.

Overall, the errors for the absence of an emotion are much lower than for the presence for both train and test, with all accuracies in the ‘no’ case above 98%. For the ‘yes’ case the results are always above 82%, but there are some particular instances worth mentioning. First, the fact that happy, and disgusted have excellent accuracies, and angry and surprised also have very good results. Happy is the easiest emotion to predict according to most other researchers, so it is expected to have good results. Regarding disgusted, in spontaneous emotions it may be more difficult to predict, but for a posed database like Radboud, and particularly where some emotions such as this one are a bit exaggerated —see figure 13 j), k), l)— and performed with the same AUs by every subject, it is easier to predict. The same applies to angry and surprised.

Regarding the hardest emotions to predict, there is fearful, in which not all users have the same expression or AUs. The contemptuous faces in the database are almost like neutral faces, with an asymmetrical mouth, that in some cases is not so perceivable as others. As the features extracted only include the position of some points of the lips, it may not be enough. Extracting more information in the corner of the lips, such as the shape of the shadows, might help better classify this one.

	parameters		train accuracy		test accuracy	
	gamma	nu	no	yes	no	yes
angry	0.00125	0.03125	99.5 %	96.5 %	100 %	96.6 %
contemptuous	0.00125	0.00625	99.6 %	93.0 %	100 %	86.2 %
disgusted	0.0001	0.03125	99.9 %	93.6 %	100 %	100 %
fearful	0.00125	0.00625	99.4 %	93.6 %	99.5 %	82.7 %
happy	0.00125	0.00125	99.9 %	100 %	100 %	100 %
neutral	0.00125	0.03125	98.2 %	93.0 %	98.5 %	86.2 %
sad	0.00125	0.03125	98.8 %	88.9 %	99.5 %	89.65 %
surprised	0.00625	0.00125	99.5 %	98.8 %	98.5 %	96.6 %

Table 1. Train and test accuracies with the optimized parameters, for each of the binary classifiers.

4.3. SVM and Deep Neural Network

After the binary classifier was built with good enough results, the next logical step was to build a classifier that tested all emotions against each other, instead of one against all the rest. As it has been explained before, the SVM was the preferred option and the Neural Network was also developed for comparison, upon proposal from the advisors.

These systems were trained using the Radboud database that contains static images, with the same amount for each emotion and subject. The SVM is first tested with 15% of the database separated for test and yielding an accuracy of 94%. Then two more tests were done in both the SVM and the Neural Network, where two different divisions between train and test were performed, and a different training and testing was done on each of the separations. Nevertheless, the same samples were selected for SVM and for the NN in each of the two divisions to have a fair comparison.

In the first test the SVM has an accuracy of 93.5% and the NN has 94.9% of correct predictions. For the second the accuracies obtained are of 87% and 84.7% respectively. Although the NN has the best overall result, the SVM seems more consistent, with a smaller difference between the best case and the worst case. Furthermore, the worst case is not as bad. For these reasons, as well as ease of use, the SVM was selected to be used afterwards. Although results are not extremely good, they seemed good enough to continue with this approach. It is important to mention, however, that the optimizations and methods used for designing both the SVM and the NN are far from optimal, given the lack of expertise. These results can be improved, and with better optimizations or designs the results would probably be better, and the best tool might be different than the used one.

The fact that two different separations of the database yield results that are so different indicates that there are some problems with the database, and maybe also with the system. The database is probably not large enough for the results to be trustable, and definitely not large enough to generalize for other faces. Although there are subjects of only two different ethnicities, there are both adults and kids, which means that there is a a significant amount of diversity for the amount of

images. Also, having three images that are almost equal for each subject and emotion is better than having only one, but it is definitely not as useful as it would be without this high correlation in groups of three images.

In table 2 the cross correlation matrix of the NN for both tests can be seen —the SVM is similar. The first important conclusion to extract is that all emotions have acceptable results, instead of having the majority of mistakes in one emotion and the rest of emotions being perfect. It can be observed, however, that there are two facial expressions that are especially problematic: sad and neutral. It is important to note that the sad emotion is mostly detected because of the face, and other researchers have concluded that when occluding the mouth it is the emotion with worst results (see figure 5). For some subjects of this database —and in general— the mouth changes when representing the sad face are subtle, and this is probably the reason for misclassifying the sad face. Some images are even difficult to classify even for human beings (for example, observe the mouth of neutral and sad faces of subject 3, in figure 13 e) and f) respectively). And neutral can be confused with several other emotions. As all emotions have some part of the face neutral or almost neutral, and others with activated muscles, if the deformations produced by these activated muscles are not enough to be detected, there is a confusion with the neutral face —sad is a good example of this in the second test. The same occurs with the neutral face when the algorithm 'thinks' there is a subtle deformation, because it is trained to detect some contortions that are not exaggerated.

Since both the SVM and the NN offered similar results without any of the options being clearly better, and the SVM is easier to train and use with the libraries that are being used, the following experiments will be conducted using the SVM. The comparison between both systems is not exhaustive, and it was meant to change the classifier if a clear advantage was seen, although other researchers have found both systems to be similar (such as [41]). Performing a complete comparison is out of the scope of this work, and requires more expertise on machine learning.

cross-correlation	**First test, 94.9% accuracy**								**Second test, 84.7% accuracy**							
angry	39	0	0	0	0	0	0	0	36	1	0	0	0	3	0	0
contemptuous	0	37	0	0	0	1	1	0	0	32	0	0	0	5	3	0
disgusted	0	0	39	0	0	0	0	0	0	0	40	0	0	0	0	0
fearful	0	0	0	34	0	3	0	2	0	0	0	30	3	6	0	1
happy	0	0	0	0	39	0	0	0	0	0	0	0	40	0	0	0
neutral	0	1	0	1	0	34	3	0	2	1	0	5	0	28	4	0
sad	0	0	0	3	0	0	35	1	1	1	0	1	0	10	27	0
surprised	0	0	0	0	0	0	0	39	0	0	2	0	0	0	0	38

Table 2. Cross correlation tables for the DNN in both tests. The emotions for the columns are in the same order than the one shown for the rows.

4.4. Laboratory database

The previous tests have been done on the Radboud database that contains pictures —not videos. So, the next logical step was to try the results of the system on videos. It was requested to try the system on users from the laboratory, and as it has been explained before, a small database of labeled videos was built for that purpose.

These videos are labeled with a -1 on the transition between emotions because those transitions are not analyzed. It would require a significantly bigger database to train a system with the different possible transitions so that it is not confused by them, because there are many possible combinations of emotions to build a transition, different timings according to the subject and circumstances, and so on. Other emotions are labeled with the number that corresponds to the emotions, and that is what is evaluated to asses the results. However, as it has been explained in the proper section, not all emotions are present on this custom database, only neutral, happy, sad and surprised.

4.4.1. Basic experiment

The first experiment performed consisted on using the previous classifier trained with the static database. As it can be seen in table 3 the results were poor. The database is not complete enough to properly perform on video with subjects not present in the database. The accuracy is extremely low, especially for some emotions, while happy obtains decent results. There are also two extremes in performance where most cases are performing either quite well or quite poorly.

There are several reasons that can explain this poor performance. One of these is when the Radboud database, which is the only —and incomplete— source of training for the classifier has all the subjects expressing an emotion in the same (exaggerated) way. An example of this would be the fact that all happy faces in Radboud exhibit a radiant smile that exposes the teeth. On the other hand, subject one has a timid or false smile that does not separate the lips, and the classifier confuses the happy faces with neutral ones, having 0% accuracy.

	happy	neutral	sad	surprised	total
subject 1	0 %	80.9 %	28.8 %	0 %	26.7 %
subject 2	60.0 %	82.8 %	45.0 %	81.7 %	67.5 %
subject 3	100 %	0 %	56.7 %	60 %	54.23 %
subject 4	33.3 %	0 %	67.7 %	16.7 %	29.2 %
subject 5	100 %	1.6 %	62.3 %	24.6 %	47.1 %
subject 6	100 %	43.4 %	100 %	86.7 %	82.2 %
total	70.1 %	39.3 %	59.5 %	46.0 %	**53.7 %**

Table 3. Accuracy of the SVM classifier trained with the Radboud database, and tested on the test videos from the laboratory database. Results are shown for four emotions and six subjects, including the overall results of each subject and emotion.

Another source of mistakes can be a subject that plays or acts an emotion differently than most subjects or identical to how that same subject plays a different emotion. This will be further discussed later on, but a good example is subject 3 for whom the neutral face is classified as sad, also attaining 0% accuracy [see figure 13 e), f) and g)]. Confusing neutral and sad faces is one of the most common mistakes, and it can also be frequently seen in subjects 4, or 6, for example.

Finally, of course, the system can fail because it does not work properly, such as inaccurate processing of the image or missing important information like frowning or other wrinkles —adding more features such as wrinkles is one of the proposals for future work.

Other examples of mistakes in the classification are the confusion of surprise with fear, that occurs in 1, 5 and sometimes 6, and is so common that even human beings sometimes mistake these emotions. The last participant is interesting because it exaggerates the different emotions more and achieves better results. However, the neutral expression cannot be exaggerated, because it is precisely not activating muscles. For that reason, that is the emotion with most mistakes for that subject, as it is confused with sad —a common confusion—, and contemptuous, an emotion that the classifier also incorrectly predicts for other subjects.

The results are in general too poor to be further discussed. More interesting deliberations will be presented in the following experiments in which modifications are done to attempt to obtain better results and evaluate how the system functions.

4.4.2. Known subjects

The next experiment consisted on improving the training with the videos from the database. As generalizing the system requires a large database and resources it was decided that in the following experiments the users in the test set would also be included in the training database. As two videos were recorded for each user, one is used for training the classifier and the other for testing the results —so, the classifier 'knows' the subjects in which it tests.

In this experiment two different trainings were performed. The first training included all the images in the Radboud database and half of the videos recorded, and its results can be seen in table 4. However, the database contains 8 emotions or facial expressions, and the videos only 4. A second training was performed with only the images from the Radboud database of the emotions that are being tested, to see whether distinguishing between fewer emotions improved the results. The results of this training can be seen in table 5.

Looking at these two tables, several elements or results stand out. The first one is that there is a significant difference in accuracy between subjects. The emotions of some subjects are classified fairly well, with accuracies between 85% and 100%; while other subjects yield extremely poor results, with subject 1 having an overall accuracy of 45.1%.

Regarding emotions, there is a significant difference in performance for each emotion, too. On one side we can see 'happy', for which other researchers have stated that it is the easiest to detect, with an accuracy of 100%. On the other side, 'sad' reaches almost 60% on the first case, and 68% on the second. Neutral and surprised are in between, with neutral being better.

Then, there are other individual subjects with interesting results. For example, subject 1 has a rather good accuracy for the neutral face, but the worst by far accuracy in surprised, and a 0% for the sad face. For both sad and surprised faces, the classifier almost always predicts 'neutral' for this subject. The reason for the classifier failing with the prediction of the surprised expression is probably the fact that the subject opens his eyes but does not open the mouth. Actually, he keeps the mouth

	happy	neutral	sad	surprised	total
subject 1	100 %	93.6 %	0 %	16.4 %	45.1 %
subject 2	100 %	100 %	46.7 %	100 %	86.9 %
subject 3	100 %	50.0 %	63.3 %	56.7 %	67.8 %
subject 4	100 %	51.6 %	86.7 %	100 %	85.0 %
subject 5	100 %	100 %	77.0 %	88.5 %	91.7 %
subject 6	100 %	100 %	100 %	100 %	100 %
total	100 %	88.2 %	59.9 %	75.5 %	**80.7 %**

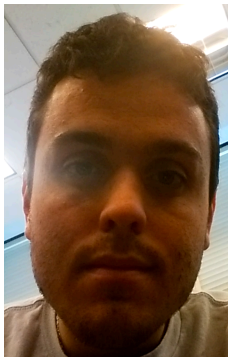
Table 4. Accuracy of the SVM classifier trained with the Radboud database and the training videos of the laboratory database. Tested on the test videos from the laboratory database. Results are shown for four emotions and six subjects, including the overall results of each subject and emotion.

	happy	neutral	sad	surprised	total
subject 1	100 %	83.0 %	0 %	34.4 %	48.0 %
subject 2	100 %	100 %	80.0 %	100 %	95.4 %
subject 3	100 %	63.3 %	46.7 %	56.7 %	66.9 %
subject 4	100 %	48.4 %	96.7 %	100 %	86.7 %
subject 5	100 %	100 %	88.5 %	83.6 %	93.4 %
subject 6	100 %	100 %	100 %	100 %	100 %
total	100 %	87.5 %	68.2 %	79.1 %	**83.3 %**

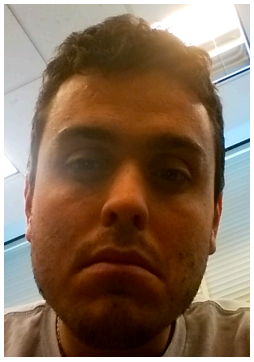
Table 5. Accuracy of the SVM classifier trained with the Radboud database —only the images of the 4 emotions tested— and the training videos of the laboratory database. Tested on the test videos from the laboratory database. Results are shown for four emotions and six subjects, including the overall results of each subject and emotion.

in a state similar to his neutral expression. Some studies that consider occlusion have seen that accuracies detecting surprise emotions are drastically worse when the mouth is occluded —other studies with different approaches disagree. Regarding the sad expression, the subject does change the mouth by pulling down the corners, but the rest of the face stays unchanged [see figure 13 a) and b)]. So this is a possible explanation for the failing of the algorithm.

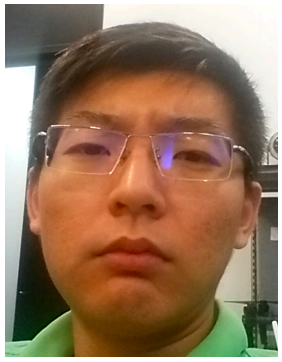
The second subject has an accuracy of 100% in all emotions except ‘sad’, where it fails in more than half of the frames for the first case. However, on the second one it improves up to 80%. When the sad expression of this subject is incorrectly classified, it is confused with the neutral face. It is important to note that of the two instances in which the sad face is requested in the video, for the first one most frames are correctly predicted, and for the second one most frames are incorrectly categorized as ‘neutral’. The subject’s face expression in both instances of the sad face is slightly



a) subject 1, neutral

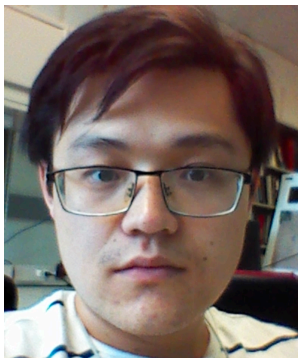


b) subject 1, sad



c) subject 2, sad, predicted

d) subject 2, sad, error

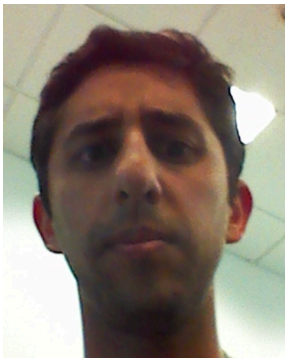


e) subject 3, neutral

f) subject 3, sad

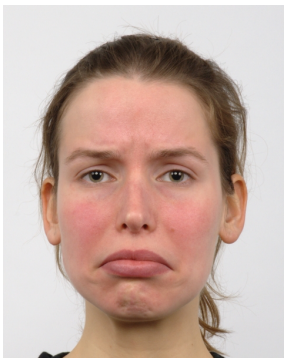
g) subject 3, sad

h) subject 4, neutral



i) subject 4, sad

j) Radboud neutral



k) Radboud sad

l) Radboud surprised

Fig 13. Face examples for commenting results; a-i from laboratory database, j,k,l from Radboud [8].

different, probably because in the first instance it comes after a neutral face, and in the second after a happy face. However, this difference is quite subtle, as it may be observed in figure 13 c) and d). No explanation for this behavior of the classifier has been found.

The fourth subject is also detected fairly well in three emotions, but almost half of the 'neutral' frames are mistaken for the sad expression. By looking at the images in figure 13 h) and i) the confusion is not surprising. First, because the video is recorded with a worse camera than most other subjects and the source of light is behind the subject, which adds a lot of noise and blur to the image. Secondly, the subject's expressiveness is fairly limited making the classification task complicated even for human beings.

Subject 3 is the second subject with worst performance —around 67% overall— and all emotions except happy —100%— have around 50% accuracy. This subject was recorded with the same camera as subject 4 but with better lighting conditions, which made the video a less blurry. 'Sad' is mostly confused with 'neutral', 'surprised' with 'sad' and 'neutral' with 'sad'. By looking at the images [figure 13, e), f) and g)] the sad and neutral faces look almost identical, with the sad one having a bit more frown, sometimes. Regarding the surprised emotion, for some frames he was distracted and changed the shape of the mouth to speak, which is where the system gets confused.

Finally, the last subject is strange on the other extreme. All four emotions are detected with an accuracy of 100%. This subject is familiar with the FACS as well as the Radboud database and other posed databases where the emotions are more clearly distinguishable, and often exaggerated. This and the good lighting conditions are probably the reasons for the algorithm always predicting correctly.

Posed databases generally have exaggerated or clearly distinguishable facial expressions, such as in Radboud [an example can be seen in figure 13 j), k) and l)], and many researchers working with posed databases use expressions that are deeply exaggerated [17]. As we have seen before, there is a huge difference between posed and spontaneous facial expressions, and the latter are notably more difficult to detect. In this database in which the algorithm is tested, subjects 1, 3 and 4 have expressions that, although posed, are not clear. Subjects 2, 5 and 6 have more exaggerated expressions that the system distinguishes appreciably better, with an average accuracy higher than 90% for the first case, and higher than 95% for the second one.

As it can also be seen in the pictures analyzed, building a database, even a simple and small one, is complicated and has the added difficulty of deciding whether the labelling is correct. In this case the labelling was not modified and the worst videos were not censored. But the consequence of this has been that several sequences of frames are dubious, and probably human beings would fail, or at least hesitate, when trying to classify them. Often, researchers who develop posed databases teach the subjects how to represent each emotion, or instruct them to make the emotions clear. Doing this would have probably produced more consistent results. These results would not be valid when using the system on real life conditions, but that requires a database which is purely *spontaneous*, which is difficult and expensive to generate.

One of the most common mistakes of the classifier is the confusion between the sad and neutral faces. Typical sad faces, when not exaggerated, are quite similar to their neutral counterparts. The main difference are the temporal or dynamic wrinkles from frowning. However, this algorithm is oblivious to the wrinkles, as it only analyzes the position of certain features in the face. Although the frown could be extracted from the movement of the eyebrows, it is often too subtle to be perceived by the system, especially with poor image quality, where the landmark extractor is not precise enough. Also, without a reference of the neutral face or that subject, such as in the last system

tested, it cannot be known if the eyebrows are lower than in the neutral position without seeing the wrinkles. For this reason, it seems that the classifier would perform better with more features that determine the horizontal and vertical wrinkles in the frown of the subject. Probably, the skin texture around the eyes, at the chin and at the corners of the mouth would also provide useful information.

In this section different tests have been reported and compared, to see possible improvements of the system and analyze its caveats. In the basic approach that consisted in training the classifier with the database that only contains pictures and testing it with videos the results showed that the system did not perform very well. A better training with a larger video-based database should produce a system that better generalizes to work with unknown faces, but this option has been discarded.

Instead a new classifier was trained using videos from the same subjects that would later be tested on. This shows results that are better than how the system would perform on unknown subjects, and has the drawback that building a system with these characteristics would not be autonomous, but shows sort of an upper-bound for how the previous system could work if it generalized very well for unknown subjects.

Finally, the results of a last system are shown, where a combination of both approaches is used. The system also learns information from the subject —not a lot, only the neutral face— but can be done autonomously with some —not absolute— reliability. It presents several difficulties that have not been analyzed. First, a binary classifier to detect a neutral face is critical to the system. Although it will have much better accuracy than a classifier that separates between eight different emotions, or four, it is still not entirely reliable. Second, this approach does not work until such face is detected, and therefore does not predict emotions, or does so with a less reliable system, until a neutral face is seen and correctly detected. Third, in a false positive of the binary classifier the system will not work at all; which means that the cost of false positives should be set higher, but this can lead to failing to detect truly neutral faces.

4.4.3. Subtracting the neutral face

This last approach is the one that uses the neutral face as a reference for each subject, subtracting its values from those of each frame as it has been described in the methods section. The goal is to find a method that improves the results offered by the first approach that does not train using the subjects in which it tests. It will also be compared with the other experiments that do use all subjects for training.

In tables 6 and 7 the results of this approach are shown. The first one displays the accuracies when the training is only done with the Radboud database and the test with the videos; whereas the second one shows the result of using both the Radboud and the training videos to learn, and the test videos to compute the accuracies.

Observing the tables, the first important remark is that there are subjects —or more specifically how these subjects perform the emotions— for whom this approach correctly predicts more accurately than the previous one, and other subjects for whom the results are significantly worse. The system becomes very good in some situations and very bad in others, but overall it yields better results than the previous approach both in the basic training (see table 3) and slightly better results than the training that also uses the videos (see table 4).

As in previous experiments, the last two subjects —the ones that have more exaggerated facial expressions— obtain much better results. It is also worth noting that the second subject does not have an especially good accuracy on the first test, but he does in the second one. This is probably due to his way of expressing emotions that changes a bit respect the subjects in Radboud —there are no asian subjects in Radboud.

The neutral emotion has the highest accuracy in this experiment, as opposed to the previous ones. This is expectable, since the method is based on using the neutral face of each subject as a reference.

The behavior when the system fails is similar to previous experiments, where the system think that an emotion is another instead of trying random values —with the exception of subject 4. For example, the first subject's prediction of the sad expression is neutral. The happy expression is sometimes confused with neutral and sometimes with contemptuous —error that has occurred before. Sad is also confused with neutral and surprised with sad.

	happy	neutral	sad	surprised	total
subject 1	0 %	87.0 %	3.4 %	0 %	20.5 %
subject 2	65 %	100 %	0 %	78.3 %	60.6 %
subject 3	100 %	31.0 %	60 %	56.7 %	62.4 %
subject 4	23.3 %	70 %	10 %	16.7 %	30.3 %
subject 5	100 %	100 %	100 %	78.7 %	95.0 %
subject 6	100 %	100 %	100 %	100 %	100 %
total	70.1 %	87.2 %	47.8 %	58.6 %	**65.4 %**

Table 6. Accuracy of the system that subtracts the neutral face. Trained only with Radboud images for comparison with Table 3. Tested on the test dataset of the laboratory database.

	happy	neutral	sad	surprised	total
subject 1	51.3 %	63.0 %	33.9 %	60.7 %	51.7 %
subject 2	100 %	100 %	86.7 %	100 %	97.0 %
subject 3	100 %	44.8 %	86.7 %	63.3 %	74.3 %
subject 4	66.7 %	100 %	0 %	56.7 %	56.3 %
subject 5	100 %	100 %	95.1 %	95.1 %	97.9 %
subject 6	100 %	100 %	100 %	100 %	100 %
total	89.6 %	88.0 %	71.9 %	83.1 %	**82.9 %**

Table 7. Accuracy of the system that subtracts the neutral face. Trained with both Radboud images and the training set, for comparison with Table 4. Tested on the test dataset of the laboratory database.

The second subject performs well but completely fails at sad, for the first table. It is also confused with neutral.

If in previous approaches expressions were commonly confused with ‘neutral’, in this one it is even more common by the nature of how the system works.

Finally, in the second table most cells show an acceptable or a good accuracy, with the exception of subject four’s sad that yields 0%. It is always surprised with neutral.

5. BUDGET

It has been estimated that 820 hours have been dedicated to this project, and considering the cost of a junior electrical engineer in the USA to be of $30 per hour, this part of the cost is a total of $24,600.

Considering the total cost of a computer to be of $800, its life expectancy of 5 years, and 7 months of use, the part of the cost related to the computer consists of $93.33.

Finally, the cost of the Nexus 5 is of $450, with a life expectancy of 4 years. 7 months of use yield $65.63.

Therefore, the total cost of this project is estimated to be $24,758.96.

Description	Cost	Quantity	Total
Hours of work	$30/h	820h	$24,600
Computer	$800	7/60	$93.33
Smartphone	$450	7/48	$65.63
Total	-	-	**$24,758.96**

Table 8. Budget estimation.

6. CONCLUSIONS

Now that the work done has been explained through different sections, some conclusions extracted during the process will be briefly presented.

First, building automatic systems to extract emotions from images is a complex task. This complexity can be seen as researchers generally build systems that work under certain restrictions, to solve the problem for a specific situation. For example, only analyzing frontal faces, working with overdramatized faces, only analyzing pictures, etc. Another example of the complexity is how there are dozens of different approaches, with many methods for extracting the information, and for classifying it.

Emotions go through three states: onset, apex and offset. Emotion detection systems can be static or dynamic; the former work better with the apex of emotions but have difficulties in the transitions, whereas the latter succeed in all stages. Dynamic systems, however, are more complex and have problems when an emotion is not entirely analyzed through all the stages.

Extracting facial expressions requires first detecting the face. This on itself is a problem with a lot of research. There are solutions that obtain good results, but the difficulty is having a method that works fast enough. In this project it is the slowest stage, allowing the processing of only two or three frames per second.

Currently, one of the problems to advance in this field is the absence of a database that can be used for properly training and benchmarking. Building one that satisfies all or most of the needs of researchers is complex, and the main impediment is constructing it with spontaneous expressions. Researchers have shown that posed and spontaneous expressions are different. Spontaneous are preferred for real-life applications, but these are also much more difficult to obtain —and to work with.

This work builds a prototype of a system that detects emotions. However, as the amount of hours available and the expertise were limited, several limitations were imposed. Faces analyzed need to be front facing, and expressions posed. Taking those restrictions into account, the goal of building an application to detect emotions has been fulfilled, with decent but not excellent performance. This will be further described next.

Two approaches have been tested. In both, 68 face landmarks are extracted, and in the first one those are fed to the classifier after normalizing. On the second approach a neutral face is stored for the subject and the landmarks are subtracted from the landmarks of each frame, to obtain the deformation respect the neutral face. This second approach would be problematic with multiple subjects because recognition would be necessary, but for only one subject it can work with a binary classifier that detects neutral faces and extracts it to be used as a reference.

These approaches have been tested in different ways. A small video database was build with subjects from the laboratory group. First the Radboud database was used for training the systems and then the videos from the small database were used for testing. This yielded poor results, with an accuracy of 53.7% for the basic approach and 65.4% for the approach that uses the neutral face. This database is not complete enough in terms of amount of images, variability of ethnicities, and in the expressions used; which reinforces the need of a good database.

To see how well the system could work in the event of a perfect training database, and what needed improvement, another test was performed using two videos for each subject. One of these

videos was for training and the other one for testing, both with the same set of emotions but sequenced in different order. The resulting accuracies were 80.7% and 82.9%, so subtracting the neutral face does not provide a significant benefit in this case. It would be interesting to see whether the neutral face as reference is worthwhile in an intermediate case, in which training is performed with a much better database than Radboud and the test subjects are not present in the training set.

From these tests several conclusions are extracted. The emotions that are easy and difficult to extract are generally the same that other researchers have reported. Sad is one of the hardest to detect, especially when the subject shows the exact same mouth shape as in neutral.

The neutral face can be confused with all, and all can be confused with the neutral face. As some emotions (or how some users express these emotions) have most of the face relaxed and some part not relaxed, it is easy to confuse an emotion with neutral. Also, sometimes the changes are subtle, which leads to the training of deciding it is not neutral as soon as some slight change is detected, in some particular occasions.

The happy emotion is the easiest to detect in both this system and the literature. An exception is the approach that subtracts the neutral face, for which the neutral face performs better in some cases. But that is because the neutral face is used as a reference, so there is bias.

Another important remark regarding the results is that in some cases (subjects, emotions) the performance is very good, higher than 90%, and in other cases it is quite poor. With the kind of mistakes done by the system, it can probably be concluded that more features need to be extracted. Detecting the wrinkles between and above the eyebrows, or between the eyebrows and the eye should help better recognize the movement of the eyebrows, improving the performance in some of the situations in which the algorithm fails. Extracting the texture of the skin (shadows) around the corners of the mouth may also be helpful.

SVM and a DNN with three hidden layers were also compared. The comparison was not complete, but both systems performed similarly, which is also consistent with the literature.

Future developments are described in the next section, as well as proposed ideas.

7. FUTURE DEVELOPMENT

Once more, due to the limited time available to develop this project, a prototype has been built. Therefore, there are several tasks left for future development that will be described here. Also, some proposals of possible changes or additions to the current system will be described.

Some of the possible improvements are related to the database. Throughout the document the importance of the database has been repeated several times. The most urgent step would probably be to use one or more video databases in addition to Radboud. More images, users, and ways of expressing emotions are needed for the robustness of the emotion detector. Also, having multiple ethnicities besides a larger number of subjects would contribute to creating a system capable of working in a more person-independent manner.

Also related to the database is the capability of analyzing faces that are not frontal. As it required more time and was not necessary for a first prototype, other facing angles were not used. To train for all angles it would probably be necessary to train with faces looking at 45 degrees and 90 degrees from the camera, maybe more angles too. Another possibility would be to have different trainings for ranges of angles and estimating the range with the landmarks.

The face detector used is far from ideal, as it is the slowest step performed in the analysis of a frame. There are some detectors that have better accuracy, and other detectors that are faster. Substituting the current detector by one that improves in results, in computational time, or both is an obvious improvement that could be made. However, it is probably not the most relevant or pressing.

In the results it has been mentioned that the accuracy could probably benefit from including more features. Some of the interesting additions could be wrinkles on top of the eyebrows and in between, shadows between the eye and the eyebrow, shadows in the corners of the mouth, or wrinkles in the corner of the eyes. Wrinkles might be simplified to a single number through a Fourier transform maybe, or other image processing tools that allowed to find the alternation between dark and clear. Also, not all features used are equally relevant, and some probably do not add information and could be removed. The study of the usefulness of each feature has not been conducted.

As it has been mentioned, the approach of subtracting the neutral face to enhance the performance was more useful in the training that only used the Radboud, that in the better performing case in which the subjects were known to the classifier. If a larger database is used, reassessing the usefulness of this method might also be of interest.

Another aspect not covered were transitions between emotions. This could be managed by labeling transitions as an extra emotion, and provided that the database is large enough the classifier should be able to know that a transition is occurring.

Another way of managing transitions is by having a dynamic classifier, as it has been explained in the state of the art. A proposed improvement is to at memory of past frames to the system with a buffer. The results from the classifier of the present and previous frames could be all fed to a new and simpler classifier that could improve the results and detect the emotion. This could possibly be improved if instead of using the result of the most likely emotion for every frame, the confidence level of all emotions at each frame were recorded. Then, the classifier would work with a value for each emotion, for each frame. Of course, both adding transitions as an extra emotion and having this voting system can be combined, too.

Finally, testing how many faces can be analyzed simultaneously without slowing down the frame rate is a task that has also been left for the future.

BIBLIOGRAPHY

[1] https://azure.microsoft.com/en-us/services/cognitive-services/emotion/ N.p., n.d. Web. 24 July 2017
[2] http://developer.affectiva.com N.p., n.d. Web. 24 July 2017
[3] https://ibug.doc.ic.ac.uk/resources/300-W/ N.p., n.d. Web. 24 July 2017
[4] Vidit Jain and Erik Learned-Miller. FDDB: A Benchmark for Face Detection in Unconstrained Settings. Technical Report UM-CS-2010-009, Dept. of Computer Science, University of Massachusetts, Amherst. 2010
[5] Yang, Shuo, Ping Luo, Chen Change Loy, and Xiaoou Tang. "WIDER FACE: A Face Detection Benchmark." Http://mmlab.ie.cuhk.edu.hk/projects/WIDERFace/support/paper.pdf. N.p., n.d. Web. 24 July 2017.[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks.
[6] Sim, Terence, Simon Baker, and Maan Bsat. "The CMU Pose, Illumination, and Expression (PIE) Database." International Conference on Automatic Face and Gesture Recognition (2002)
[7] http://www.pitt.edu/~emotion/ck-spread.htm N.p., n.d. Web. 24 July 2017
[8] Langner, O., Dotsch, R., Bijlstra, G., Wigboldus, D.H.J., Hawk, S.T., & van Knippenberg, A. (2010). Presentation and validation of the Radboud Faces Database. Cognition & Emotion, 24(8), 1377—1388. DOI: 10.1080/02699930903485076
[9] https://mmifacedb.eu N.p., n.d. Web. 24 July 2017
[10] http://www.engr.du.edu/mmahoor/DISFAContent.htm N.p., n.d. Web. 24 July 2017
[11] https://mug.ee.auth.gr/fed/ N.p., n.d. Web. 24 July 2017
[12] http://dlib.net N.p., n.d. Web. 24 July 2017
[13] Kazemi, V., & Sullivan, J. (2014). One millisecond face alignment with an ensemble of regression trees. 2014 IEEE Conference on Computer Vision and Pattern Recognition. doi:10.1109/cvpr.2014.241
[14] https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/ N.p., n.d. Web. 24 July 2017
[15] Wettum, Y. V. (2017). Facial landmark tracking on a mobile device. Retrieved July 24, 2017, from http://essay.utwente.nl/71696/1/van_Wettum_BA_EEMCS.pdf
[16] Jean-Yves Bouguet. “Pyramidal implementation of the Lucas Kanade feature tracker”. In: Intel Corporation, Microprocessor Research Labs (2000), p. 9.
[17] Bettadapura, Vinay. "Face expression recognition and analysis: the state of the art." arXiv preprint arXiv: 1203.6722 (2012).
[18] Viola, Paul, and Michael J. Jones. "Robust real-time face detection." International journal of computer vision 57.2 (2004): 137-154.
[19] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.
[20] https://www.tensorflow.org N.p., n.d. Web. 24 July 2017
[21] J. Duchi, E. Hazan & Y. Singer. Journal. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: Journal of Machine Learning Research 12 (2011) 2121-2159.
[22] http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html N.p., n.d. Web. 24 July 2017
[23] http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html N.p., n.d. Web. 24 July 2017
[24] http://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/svms/RBFKernel.pdf N.p., n.d. Web. 24 July 2017
[25] P. Ekman and W.V. Friesen, “Constants across cultures in the face and emotions,” J. Personality Social Psychology, vol. 17, no. 2, pp. 124-129, 1971.
[26] P. Ekman, “Strong evidence for universals in facial expressions: A reply to Russell's mistaken critique,”

Psychological Bulletin, vol. 115, no. 2, pp. 268-287, Mar. 1994.
[27] P. Ekman and W.V. Friesen, “Manual for the Facial Action Coding System,” Consulting Psychologists Press, 1977.
[28] MPEG Video and SNHC, “Text of ISO/IEC FDIS 14 496-3: Audio,” in Atlantic City MPEG Mtg., Oct. 1998, Doc. ISO/MPEG N2503.
[29] W.G. Parrott, “Emotions in Social Psychology,” Psychology Press, Philadelphia, October 2000.
[30] Y. Tian, T. Kanade and J. Cohn, “Recognizing Action Units for Facial Expression Analysis,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23,
no. 2, pp. 97–115, 2001.
[31] P. Ekman and E.L. Rosenberg, “What the face reveals: basic and applied studies of spontaneous expression using the facial action coding system (FACS)”, Illustrated
Edition, Oxford University Press, 1997.
[32] M. Pardas and A. Bonafonte, “Facial animation parameters extraction and expression recognition using Hidden Markov Models,” Signal Processing: Image Communication, vol. 17, pp. 675–688, 2002.
[33] P. Ekman and E.L. Rosenberg, “What the face reveals: basic and applied studies of spontaneous expression using the facial action coding system (FACS)”, Illustrated
Edition, Oxford University Press, 1997.
[34] F. Bourel, C.C. Chibelushi and A. A. Low, “Recognition of Facial Expressions in the Presence of Occlusion,” Proc. of the Twelfth British Machine Vision Conference, vol. 1, pp. 213–222, 2001.
[35] I. Kotsia, I. Buciu and I. Pitas, “An Analysis of Facial Expression Recognition under Partial Facial Image Occlusion,” Image and Vision Computing, vol. 26, no. 7, pp. 1052-1067, July 2008.
[36] N. Sebe, M.S. Lew, Y. Sun, I. Cohen, T. Gevers, and T.S. Huang, “Authentic Facial Expression Analysis,” Image and Vision Computing, vol. 25, pp. 1856- 1863, 2007.
[37] M.S. Bartlett, G. Littlewort, I. Fasel, and R. Movellan, “Real Time Face Detection and Facial Expression Recognition: Development and Application to Human Computer Interaction,” Proc. CVPR Workshop on Computer Vision and Pattern Recognition for Human-Computer Interaction, vol. 5, 2003.
[38] F. Dornaika and F. Davoine, “Simultaneous Facial Action Tracking and Expression Recognition in the Presence of Head Motion,” Int. J. Computer Vision, vol. 76, no. 3, pp. 257–281, 2008.
[39] I. Cohen, N. Sebe, A. Garg, L.S. Chen, and T.S. Huang, “Facial Expression Recognition From Video Sequences: Temporal and Static Modeling”, Computer Vision and Image Understanding, vol. 91, pp. 160-187, 2003.
[40] N. Sebe, M.S. Lew, Y. Sun, I. Cohen, T. Gevers, and T.S. Huang, “Authentic Facial Expression Analysis,” Image and Vision Computing, vol. 25, pp. 1856- 1863, 2007.
[41] K. Anderson and P.W. McOwan, “A Real-Time Automated System for Recognition of Human Facial Expressions,” IEEE Trans. Systems, Man, and Cybernetics Part B, vol. 36, no. 1, pp. 96-105, 2006.
[42] http://dlib.net/ml_guide.svg N.p., n.d. Web. 24 July 2017
[43] http://emotion-research.net/wiki/Databases N.p., n.d. Web. 24 July 2017
[44] http://www.kasrl.org/jaffe.html N.p., n.d. Web. 24 July 2017
[45] http://grail.cs.washington.edu/projects/deepexpr/ferg-db.html N.p., n.d. Web. 24 July 2017
[46] http://www2.ece.ohio-state.edu/~aleix/ARdatabase.html N.p., n.d. Web. 24 July 2017
[47] http://blog.christianperone.com/2015/08/convolutional-neural-networks-and-feature-extraction-with-python/ N.p., n.d. Web. 24 July 2017
[48] https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/ N.p., n.d. Web. 24 July 2017
[49] https://github.com/googlesamples/android-Camera2Video N.p., n.d. Web. 24 July 2017

GLOSSARY

Some expressions that appear on this document may need clarification.

Basic emotions: anger, disgust, fear, joy, sadness and surprise

Posed (expression): facial expression that is not spontaneous. It can be acted (often exaggerated) or simply the result of an external influence, such as being in an unfamiliar environment like a laboratory, or the knowledge of being observed, recorded, etc.

Spontaneous (expression): facial expression that is not posed. Real way in which a subject expresses emotions when not being conscious of it.

AU: action units. A way of denoting changes in facial behaviors produced by muscle groups. Proposed in FACS, and widely used for describing facial expressions.

dlib: [12] C++ library used in this project.

DNN: deep neural network. Particular case of NN described in

FACS: [31] facial action coding system. System to code the movement of regions or muscles of the face so that it can be analyzed objectively. It has become a de-facto standard.

FAP: [28] face animation parameter. Introduced by the MPEG group to be a standard for describing facial animations. Has later been used for describing facial expressions in human beings, similarly to the FACS system.

MLP: multi-layer perceptrone. Particular case of DNN described in

NN: (artificial) neural network. Machine learning technique described in

SVM: super vector machine. Machine learning technique described in