TÍTULO: Development of an interaction system based on face processing

AUTOR: Laura Gálvez Jiménez

TITULACIÓN: Grado en Ingeniería de Sonido e Imagen

TUTOR: Iván Pau de la Cruz

DEPARTAMENTO: Ingeniería Telemática y Electrónica

Miembros del Tribunal Calificador:

PRESIDENTE: Inmaculada Almodóvar Martín

TUTOR: Iván Pau de la Cruz

SECRETARIO: Esther Gago García

Fecha de lectura: 12 de julio de 2018

Calificación:

El Secretario,
Development of an interaction system based on face processing

Laura Gálvez Jiménez
Julio 2018
Abstract

Nowadays, communication between humans and machines has become very recurrent due to computers are more and more present in all aspects of our lives and it’s very important for the machine to understand the person’s feelings if it’s wanted to have a conversation as real as possible.

This project studies several solutions for recognizing users’ feelings in the area of the facial expressions by using Artificial Intelligence methods, in particular, Machine Learning algorithms based on previous researches. From this study, it is made two algorithms with the same goal, to predict facial emotions by capturing the user’s face through webcam, but different methods. To achieve this objective is necessary to go through some steps, from capturing faces to predicting emotions, which are explained in detail in this research.

The final part in this project is testing both algorithms with the aim of getting their accuracies and levels of confidence and afterwards, comparing them and defining which solution is the best depending on the objectives to reach and the type of usage desired.
Table of contents

Abstract ........................................................................................................................................... i
Figures ........................................................................................................................................ iii
Tables ........................................................................................................................................... v
Glossary ........................................................................................................................................... v
1. Conceptualization ........................................................................................................................ 1
  1.1. Motivation................................................................................................................................ 1
  1.2. Goals ....................................................................................................................................... 1
  1.3. Methodology .......................................................................................................................... 1
  1.4. Structure ............................................................................................................................... 2
2. Theoretical background ............................................................................................................... 4
  2.1. Facial Emotion Recognition .................................................................................................. 4
  2.2. Facial expression recognition system .................................................................................... 7
  2.3. Machine Learning ................................................................................................................ 26
3. Datasets ....................................................................................................................................... 37
  3.1. The extended Cohn-Kanade Dataset ..................................................................................... 37
4. Implementation ........................................................................................................................... 44
  4.1. Program design ...................................................................................................................... 44
  4.2. Experiments and results ....................................................................................................... 57
    4.2.1. Support Vector Classification ......................................................................................... 57
5. Conclusions ............................................................................................................................... 68
6. Bibliography .............................................................................................................................. 70
I. Annex ......................................................................................................................................... 74
  Installation .................................................................................................................................... 74
  File organization .......................................................................................................................... 74
Figures

Figure 1. Plutchik Wheel of Emotions ................................................................. 5
Figure 2. Basic emotions with their AUs .............................................................. 6
Figure 3. Diagram of a facial expression recognition system .............................. 7
Figure 4. Bilinear interpolation ............................................................................. 8
Figure 5. Example of clipping the histogram ..................................................... 9
Figure 6. Application of Haar filters .................................................................. 10
Figure 7. Basic Haar filters ................................................................................. 10
Figure 8. Example of the application of a Haar filter ......................................... 11
Figure 9. Extended Haar filters .......................................................................... 11
Figure 10. Calculation of the integral image ...................................................... 12
Figure 11. Simple Classifier ............................................................................... 13
Figure 12. Adaboost classifier ............................................................................ 14
Figure 13. Example of sliding window ............................................................... 16
Figure 14. Pyramidal sliding window method ..................................................... 17
Figure 15. Example of gradient ......................................................................... 17
Figure 16. Calculation of intensity ...................................................................... 18
Figure 17. Application of the gradient ............................................................... 18
Figure 18. Image divided in cells with their gradients .......................................... 19
Figure 19. Range of orientations ....................................................................... 19
Figure 20. Histogram of gradient magnitudes ..................................................... 20
Figure 21. Blocks and their histograms ............................................................... 20
Figure 22. Getting the final descriptor ............................................................... 21
Figure 23. Gabor Wavelets ............................................................................... 22
Figure 24. Hybrid methods ............................................................................... 23
Figure 25. Decission trees types ....................................................................... 24
Figure 26. Examples of decision trees ............................................................... 25
Figure 27. Gradient descent ............................................................................. 25
Figure 28. Artificial Intelligence and its fields .................................................... 26
Figure 29. Example of SVM ............................................................................. 29
Figure 30. Example of samples not linearly separable ....................................... 30
Figure 31. Example of mapping ........................................................................ 31
Figure 32. Effect of the soft margin constant value ........................................... 33
Figure 33. Effect of the degree value .................................................................. 33
Figure 34. Effect of the gamma value ................................................................ 34
Figure 35. Example of a grid-search ................................................................. 35
Figure 36. Block diagram of the system .................................................................38
Figure 37. Example of a file in CK images folder ....................................................40
Figure 38. Example of landmarks file ....................................................................41
Figure 39. Example of FACS code .........................................................................41
Figure 40. Example of FACS code file ....................................................................41
Figure 41. Example of Emotion code ......................................................................42
Figure 42. Detecting faces ......................................................................................44
Figure 43. Drawing landmarks ...............................................................................45
Figure 44. Detecting face and smile .........................................................................45
Figure 45. First expressions detection app. ..............................................................46
Figure 46. Extract of the code. ................................................................................46
Figure 47. Mouth landmarks ..................................................................................47
Figure 48. Image processing ..................................................................................49
Figure 49. 68 facial landmarks ..............................................................................50
Figure 50. Captures from the application. .................................................................50
Figure 51. Gravity center considering AUs ...............................................................51
Figure 52. Block diagram of the ML model training ...............................................53
Figure 53. App diagram. .........................................................................................54
Figure 54. App solutions. .......................................................................................55
Figure 55. Grid-search resultant of the polynomial kernel ......................................58
Figure 56. Grid-search resultant of the RBF kernel ...............................................59
Figure 57. Grid-search resultant of the sigmoid kernel .........................................59
Figure 58. Example of a right classification - Sadness .........................................61
Figure 59. Example of a right classification - Surprise ..........................................61
Figure 60. Example of a wrong classification - Anger ..........................................61
Figure 61. Example of a wrong classification - Surprise .......................................62
Figure 62. Example of a wrong classification - Sadness ........................................62
Figure 63. Example of a right classification - Happiness ......................................62
Figure 64. Example of a wrong classification - Contempt .....................................64
Figure 65. Example of a right classification - Sadness ..........................................65
Figure 66. Example of a wrong classification - Fear .............................................65
Figure 67. Example of a right classification - Happiness .......................................65
Figure 68. Example of a right classification - Surprise ........................................66
Figure 69. Example of a wrong classification - Angry ..........................................66
Tables

Table 1. Kernel types. .................................................................................................................. 31
Table 2. Kernel types with optimization parameters ................................................................. 32
Table 3. Emotion description in terms of facial action units. ..................................................... 37
Table 4. Frequency of emotions in the dataset .......................................................................... 38
Table 5. Emotions and examples of the dataset ...................................................................... 39
Table 6. Linear kernel tuning .................................................................................................... 57
Table 7. Percentage of right classifications with emotions solution ......................................... 60
Table 8. Accuracy AU models .................................................................................................. 63
Table 9. Percentage of right classifications with AU solution ................................................ 64

Glossary

HMI: Human-Machine interaction
FACS: Facial Action Coding System
AU: Action Unit
CK+: Cohn Kanade +
AHE: Adaptive Histogram Equalization
CLAHE: Contrast Limited Adaptive Histogram Equalization
CDF: Cumulative Distribution Function
HOG: Histogram of Oriented gradients
LBP: Local Binary Patterns
PCA: Principal Components Analysis
LDA: Linear Discriminant Analysis
FLD: Fisher Linear Discriminant
AAM: Active Appearance Model
ASM: Active Shape Model
EBGM: Elastic Bunch Graph Matching
ML: Machine Learning
SVM: Support Vector Machine
RBF: Radial Basis Function
1. Conceptualization

1.1. Motivation

Machine Learning and computer vision are very present in daily life due to more and more companies are looking for introduce these technologies in their solutions to get more benefits. In addition, there are many studies related to the medical area with the aim of improving the quality of life of people by bringing these techniques to the detection of diseases and rehabilitation for people with disabilities. Specifically, the facial emotion recognition is very used in fields as neuro-marketing, games and social networks, which are at their peak. Keeping that in mind, it’s very important to have knowledge of these technologies and their applications to continue improving quality of life by searching other possible solutions.

1.2. Goals

The main purpose of this project is the development of an application based on computer vision techniques which allows the human-machine interaction through facial gestures. To reach this objective there are some secondary goals:
- Understanding human emotions
- Learning face detection and methods to do this task
- Learning how to extract characteristics from the detected face
- Understanding Machine Learning and its types
- Learning the ML algorithms which are going to be used

Before starting to work in this tasks, it is necessary to understand Python and OpenCV which are necessary to develop the algorithm and so get the main objective.

At the end of the project, we’ll be able to understand computer vision and ML and how to apply them in a facial emotion detection algorithm.

1.3. Methodology

The methodology defined for this work is divided in the next steps:

1. Study of the scientific and technological background.
- The first step is a research of the different concepts about humans’ emotions and some of the theories made by scientists and psychologists over the history.
Development of an Interaction System based on Face Processing

1. Elaboration of a scheme of how a facial emotion recognition system works.
2. Creation of a bibliography search plan based on the proposed scheme.

2. Development of the algorithms.

2.1. First steps with Python and OpenCV
2.2. Image processing and face detection
2.3. Characteristics extraction
2.4. Training the Machine Learning model

3. Test the application and gathering of results.

4. Evaluation of the results obtained.

1.4. Structure

The project is divided in 6 sections:

1. Conceptualization. In this part are presented the motivation, the objectives, the methodology and structure of the project.
2. Theoretical background. In this section is exposed the research about emotions and techniques uses for the development of the algorithm.
3. Dataset. This section explains the CK+ dataset which is a databased of images with their corresponding FACS and emotions codes.
4. Implementation. This section is divided in two subsections. The first one explains all the steps followed to get the final application from the first scripts to the main program. The second part exposes the different experiments done and the results obtained from them.
5. Conclusions. In this last section it’s made a summary of the project and it’s analyzed the results obtained with the final conclusions.
2. Theoretical background

2.1. Facial Emotion Recognition

Introduction

Facial expression is the most effective way of non-verbal communication for humans to provide information about their emotions and feelings, sometimes becoming more relevant in a conversation than words. This fact brings us to think that emotions are an essential part of the human experience: the brain is influenced by emotions to make a decision.

Due to the importance of emotions, when the objective is to create a machine expected to understand humans and communicate with them, its design must start with emotion recognition. These systems which simulates the communication between machines and humans are known as human machine interaction systems (HMI). To create that, the machine needs to learn to express itself in an oral and a mimic way, so the human-machine interaction seems to be as a human-human interaction. Despite oral communication is very important in the system, facial expression recognition is one of the best steps for improving HMI systems. For that reason, most of the studies focused on this area of investigation are centered on the improvement of the management the computer does with the user's information and the interface to provide that information via webcam, in the case of facial recognition.

The first step to get an intelligent communication is learning to classify emotions. Next sections explain emotions and emotion recognition theories.

Studies about facial emotion recognition

As said before, facial expressions play a very relevant role in the communication of feelings and emotions. This affirmation is ratified by Albert Mehrabian in his publication [1], a German psychologist whose theory sustains that, in a personal or intimal conversation, only the 7% of the information received by the interlocutor comes from the words said. The remaining 93% comes from non-verbal communication, that is, body language and voice characteristics, such as volume, tone, intonation, etc.

Since more than a century, there has been a debate about universality of facial expressions through cultures. In 1872, Darwin explained in [2] that ethnic or culture don’t influence on the ability of express the same way some emotions.
Also, he claimed that individual differences, such as personality and gender, affects the intensity and the way to reveal an emotion.

Later, in the 70’s, Ekman and Friesen expounded in [3], based on Darwin’s theory, there are 6 basic emotions and facial expressions for each one of them: anger, disgust, fear, surprise, sadness and happiness. In addition, these emotions are universal and innate. This emotions classification is usually used in HMI systems.

Other emotions classification used in HMI systems, is the one proposed on [4], by Parrott in 2001, who describes 136 emotional states divided into groups of primary, secondary and tertiary emotions. As in Ekman studies, primary emotions are 6: anger, disgust, fear, surprise, sadness and happiness.

On the other hand, Plutchik suggest a theory exposed on [5] based on the division of the emotions in primary, secondary and tertiary through a classification system called Plutchik Wheel of Emotions. In this case, the basic emotions are 8: anger, disgust, fear, surprise, sadness, joy, anticipation and trust.

![Plutchik Wheel of Emotions](Image)
There are other recent studies, such as [7] made by Vinay Bettadapura in 2012, who recognize more emotions than the 6 defined by Ekman, from posed expressions to micro expressions through spontaneous expressions. As posed expressions are artificial and micro expressions very difficult to detect, studies are focused on the detection and recognition of spontaneous expressions, which could be sometimes confusing.

Nowadays, most of the facial emotion recognition studies are based on Ekman’s Facial Action Coding System (FACS). The main purpose behind this system is to classify human facial movements based on their facial appearance. FACS measurement units are Action Units (AUs). These combine muscles in the face and are descriptive only. There are 46 different units which are combined to give expressions. For example, a disgust face has 3 AUs: 9+10+25, numbers that corresponds to nose wrinkle, upper lid raise and lips part. These numbers can be accompanied by letters from A to E which indicates the intensity of the gesture, being A the minimum and E the maximum intensity.

In the image below, it is shown the 8 expressions used in a simple facial expression recognition system and their corresponding Action Units. These FACS will be used in the CK+ dataset explained in the next sections.

![Figure 2. Basic emotions with their AUs. [8]](image-url)
2.2. Facial expression recognition system

Facial expression recognition systems are focused on detect human faces and distinguish what kind of emotion they are expressing. These systems have become the center of many studies due to they can be very useful in several investigation areas apart from psychology, like medicine, videogames, home automation or education.

In the following figure is described how a facial expression recognition system works and which are the steps it takes to get the right emotion in a general way:

1. Capture image of a participant. In this case, the system captures a sequence of images, that is a video from a webcam.
2. Face detection. From each frame, it detects the face of the participant by methods which will be explained later.
3. Remodeling and normalization. It is necessary that the image taken has the same format as the images in the dataset, so they must be standardized.
4. Characteristics extraction. It is selected the method to extract the information from the detected face to process it and get the emotion, such as facial landmarks.
5. Expression recognition algorithm. In this step, a machine learning algorithm is executed to process the information obtained in the previous step and get the correct emotion.

![Figure 3. Diagram of a facial expression recognition system](image-url)
REMODELING AND NORMALIZATION

The aim of this step is standardizing the user’s image by using image processing techniques and then apply the face detections algorithms and extract the landmarks. One image processing method is the Contrast Limited Adaptive Histogram Equalization, CLAHE [9].

The Adaptive Histogram Equalization is an image processing technique which improves contrast differing from other histogram equalizations in that this method considers local information by processing small blocks called “tiles” of the image while an ordinary histogram equalization is a global process. As the AHE tends to over amplify noise it is necessary to limit the contrast amplification, what it’s called CLAHE.

The AHE process starts defining a neighborhood of N×N pixels around a central pixel and obtaining the CDF. The central pixel is modified by mapping its intensity value in the CDF. Then, the region is moved to the adjacent pixel and the process is repeated.

To reduce computational time AHE is made for non-lapped sections and for each one it is determining its own function, but with this procedure appears a block effect that is undesirable. The solution is to make a bilinear interpolation.

1. Define reference points located in the middle of each section.
2. Each original pixel, r, is modified in a new pixel, s, by using neighbor transformation functions.

\[
S = (1-y)[(1-x)T_A(r) + xT_B(r)] + y[(1-x)T_C(r) + xT_D(r)]; \quad \text{where A, B, C and D are the central points of the neighbor sections with their respective transformation functions } T_A(r), T_B(r), T_C(r) \text{ and } T_D(r).
\]

The interpolation of each pixel in the image considers 3 different cases:

1. If the pixel belongs to an intern region, the interpolation is made with the four adjacent transformation functions: up left, up right, down left, down right.

Figure 4. Bilinear interpolation
2. If the pixel belongs to an edge region, the interpolation is made with the two adjacent transformation functions: up and down or left and right.

3. If the pixel belongs to a corner region, the interpolation is made with this pixel function.

As said previously, AHE has a disadvantage which is the overamplify of noise in homogenous sections due to the transformation function saturates the intensity, which generates a pick in the histogram. To solve this problem, it is used CLAHE technique. The algorithm has two parts: the generation of the transformation function for each tile and the bilinear interpolation.

This method clips the histogram with a defined value and redistribute uniformly all the cut values over the histogram to keep the total number of pixel in the image. From the obtained histogram it's generated the transformation function.

![Figure 5. Example of clipping the histogram](image)

**FACE DETECTION**

In this stage the objective is to detect the user’s face and then separate it from the background. To achieve this goal, there are several methods commonly used in detection algorithms.

- **Haar + Adaboost detector, HaarCascade Classifier**

This method was proposed by Paul Viola and Michael Jones in [10] and it is based on a Haar + Adaboost detector. This detector is composed by a descriptor, Haar, and a classifier method, Adaboost.

It is applied a series of filters, with different sizes and multiple positions in the image, which will calculate several characteristics of the input. To calculate in a quick way these characteristics it is going to be used the integral image.
Adaboost is a Machine Learning algorithm used for classification, which consists in a cascade of classifiers, each one more restrictive.

- **Haar descriptor**
  A set of Haar filters are applied over the whole image. Haar filters are a combination of rectangles of the same size, vertically and horizontally adjacent. In the image below, it is represented the basic Haar filters. Black rectangles represent areas with a positive contribution to the filter, while white rectangles represent areas with a negative contribution. White and black rectangles must have the same size.

\[
\sum_{(x,y) \in \text{Black}} I(x,y) - \sum_{(x,y) \in \text{White}} I(x,y)
\]

The result of applying one of these filters over an image is the difference between the addition of the intensity of the pixels in the black area and the addition of the intensity of the pixels in the white area.

To understand better, there is an example of the application of a Haar filter:
The filters are applied with different sizes and positions such as it was done with the gradient. Nevertheless, Haar filters are not only centred on the edges of the objects, but it detects other changes of intensity in the image in bigger scales.

Each filter is applied in every scale vertically and horizontally and, in turn, each scale is applied in every possible position. The result of each filter in each scale and position is a Haar characteristic.

There are other Haar filters, called extend filters which are rotations and new configurations of the basic filters to detect other type of characteristics.

As it was commented before, it is possible to calculate Haar characteristics in a quicker way using an integral image. This image is a transformation of the original image which has as result a new image of the same size so that
the value of each pixel will be the addition of all the pixels in the original image situated on the top left.

![Integral Image](image)

*Figure 10. Calculation of the integral image. [12]*

The calculation of the integral image is very efficient with only a round from the accumulated addition of the actual row and the value of the previous row in the integral image:

\[
II(x, y) = II(x, y - 1) + s(x, y),
\]

where \(II(x, y-1)\) is the pixel in the previous row and \(s(x, y)\) is the accumulation of the intensity in the actual row and is calculated by

\[
s(x, y) = \sum_{x' \leq x} I(x', y) = s(x - 1, y) + I(x, y).
\]

If it is required to apply extended filters, the integral image is calculated as the addition of all intensities at 45° with two steps. The first one is a round through the image from the top to the bottom and from left to right. The second one will be from the bottom to the top and from the left to the right.

- Adaboost
  In this classifier the classification limit is not found from a parametric function, but as a result of combining a set of simple classifiers. It provides a global classifier which minimize the classification error and have a great ability of generalization.
  Each simple classifier is learned by giving a different weight to each example, so that it is given a higher weight to the examples that are classified in a wrong way by the previous classifiers and a lower weight to the examples that are properly classified.
  Simple classifiers are called weak classifiers which will be a decision stump, a binary decision tree with depth 1. This calculates one function of
characteristic for each input image and determine a threshold which will distinguish positive examples from negative ones. Each classifier depends on one Haar characteristic.

\[ h(x) = \begin{cases} -\alpha, & f(x) < \theta \\ \alpha, & f(x) \geq \theta \end{cases}, \quad \text{where } \alpha = \{-1, +1\} \]

Parameters that determine the classifier are the threshold and the sign as it can be seen. To fix them, it must be analyzed in an exhaustive way every possible threshold values to finally select the one with a minimum classification error.

Given a set of samples, result of the previous steps of classification, each sample will have an associated relative weight which will be higher for the wrong classified examples in earlier steps.

To find the optimal value of the threshold, the training samples are ordered according to the value of the Haar characteristic that is being used for this classifier. The threshold with the lowest error is selected.

In each iteration of the learning process a weak classifier is trained for each Haar characteristic, considering that in an image it is going to be calculated many characteristics and Adaboost will select the most relevant.

Each characteristic will give a different classifier with its corresponding error. The result of a specific iteration is the classifier with the lowest error which corresponds to a particular Haar characteristic.
On the other hand, in each iteration of the process, the relative weight of the samples is actualized after learning a weak classifier and according to the error of this one.

\[
\begin{align*}
  w_i(t + 1) & = \begin{cases} 
    \frac{w_i(t)}{\epsilon_t^2}, & h(x_i) \neq y_i \\
    \frac{w_i(t)}{(1 - \epsilon_t)^2}, & h(x_i) = y_i
  \end{cases} \\
  \epsilon_t & < 0.5 \\
  \begin{cases} 
    \frac{1}{2\epsilon_t} > 1 & \text{wrong classified} \\
    \frac{1}{2(1 - \epsilon_t)} < 1 & \text{well classified}
  \end{cases}
\end{align*}
\]

The relative weight is actualized according to the previous relative weight and an actualization factor which depends on the classification error of the previous learned weak classifier.

As it is said, Adaboost is an iterative process, with the total number of iterations fixed to T, which combines all weak classifier in a global one. In each iteration a Haar characteristic is selected and a weak classifier learned. The result is a global classifier which is the addition of all weak classifiers weighed.

\[
H(x) = \text{sign} \left( \sum_{i=1}^{T} \alpha_i h_i(x) \right)
\]

Classifiers with lower error will be given more importance due to

\[
\alpha_t = \frac{1}{2} \log \left( \frac{1 - \epsilon_t}{\epsilon_t} \right) \\
\epsilon_t \rightarrow 0, \alpha_t \rightarrow \infty, \quad \epsilon_t \rightarrow 0.5, \alpha_t \rightarrow 0
\]

The objective of the cascade of classifiers is to reject the highest number of windows in an image which, in this case, doesn’t contain a face.

As seen in the image below, Adaboost is a sequential combination of classifiers.

![Figure 12. Adaboost classifier](image)
The first classifier receives as input all windows in the image. Then, those ones detected as faces will be sent to the second classifier and so consecutively until get to the last one. Only images recognized as faces by the last classifier will be recognized definitely as faces. Each classifier is centered on evaluate the difficult images which the last classifier could not classify.

The number of characteristics used in the first classifiers is very low, therefore images are discarded quicker.

The objective of each level, fixed at the beginning, is to reach a determinate efficiency index with the lowest number of characteristics, so that false detections vs. correct detections. To get to this goal, each level of the cascade will be a strong classifier trained with Adaboost and it is introduced a margin which is selected from an iterative search.

\[ H(x) = \text{sign} \left( \sum_{i=1}^{T} \alpha_i h_i(x) + s \right) \]

If the number of false negatives is higher than the objective, \( s > 0 \), thus, the margin is moved to the left. On the other hand, if the number of false positives is higher than the objective, \( s < 0 \), therefore the margin is moved to the right.

If it is not possible to find a value \( s \) that accomplishes the objectives, it is considered that strong classifier is not robust enough, so another strong classifier is trained adding more characteristics. However, it is possible that, although many characteristics are added to a level, the objective could be too restrictive, hence it is necessary to fix a maximum number of characteristics per level.
HOG + linear classifier

This method consists in sliding a window classifier over an image pyramid, in particular, it slides a linear classifier over a HOG pyramid.

As seen in the images, the sliding window method consists in a window that has a determinate size, and which is moved through the image with a specific shift on axis X and Y, thus, it is created a movement pattern.

To classify windows as faces or non-faces it is necessary to calculate their descriptors, such as LBP, HOG or Haar. However, if there are two windows overlapped it is calculated twice the descriptors of the areas which are overlapped. To avoid that, the solution is to modify the performance sequence, that is, to calculate first the descriptor of the initial image and then apply the sliding window process. In this case, the information which will reach the classification module will be windows coming from the image with the calculated descriptors instead of windows from the original image.

Another key aspect to consider is the size of the window. The classification process needs all the windows to have the same size, so it’s necessary to use interpolation methods.

It’s possible to have faces of different sizes, therefore if there are faces with a larger size than window has, it won’t be detected. The solution is to sweep the image with a determinate window and then repeat the process changing the size of the window. The strategy to come to the solution will be the **pyramidal sliding window method**. In this method the size of the window is not modified but the size of the original image.

The process starts with the original image which will decrease its size until the canonical window gets bigger than the image. In each level the image is re-scaled and softened and it will be calculated its related descriptor.
So, the faces which are located in the distance will be detected in the original image, that is in the pyramid base, and the faces located nearer will be detected in the pyramid peak, or rather, in the smallest image.

This is the general operation of the pyramidal sliding window method, but to detect faces, the pyramid will be formed by HOG and the window classifier is a linear classifier that will be explain with detail later.

HOG, Histogram of Oriented Gradients, uses information of the gradient, that is, the outline of the objects. This descriptor makes the most of the gradient information by combining it in local histograms calculated in cells which are distributed over the image. This information allows to distinguish shapes, so it's good at detecting and recognizing objects. The local histograms are assembled in bigger blocks to normalize the final representation and make it more invariant to changes and distortions.

1. Calculating the gradient

Gradient is based in two values: direction and magnitude. These values are calculated at the level of pixel; hence they provide local information.
The calculation in a HOG context is based on the difference of intensity between neighbor pixels in a vertical or horizontal direction.

\[
\begin{array}{cccc}
0 & 0 & 255 & 255 \\
0 & 0 & 255 & 255 \\
0 & 0 & 0 & 255 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

\(\text{dx} \quad \text{dy}\)

\(\text{Figure 16. Calculation of intensity}\)

\[\begin{align*}
\text{dx} &= l(x+1, y) - l(x-1, y) \\
\text{dy} &= l(x, y+1) - l(x, y-1)
\end{align*}\]

To calculate orientation and magnitude of the gradient:

Orientation: \(\theta(x, y) = \tan^{-1}\left(\frac{\text{dx}}{\text{dy}}\right)\)

Magnitude: \(g(x, y) = \sqrt{\text{dx}^2 + \text{dy}^2}\)

\(\text{(a)} \quad \text{(b)}\)

\(\text{Figure 17. Application of the gradient; a) Original image; b) Gradient of the image [15]}\)

2. Calculating the HOG

Due to classifiers usually require a global representation of the image as an input, it is necessary to transform the local gradients calculated for each pixel previously in a global descriptor of the image characteristics. To get to a global descriptor there are two steps to take:
1- Divide the image in cells and calculate for each one a histogram of orientations. There are some aspects to consider:

- The cell sizes.
- The division of the orientations range and the number of intervals which will be fixed.

![Image divided in cells with their gradients.][12]

- Assigning each pixel of the cell in an interval according to the gradient orientation. If it considers this orientation as it could be positive or negative, the range of orientations will be between 0° and 360°. In other case, the range will be between 0° and 180°. Another point to be contemplated is the number of intervals which the range of orientations will be divided into. In this example, there are 9 intervals, which is a typical value, but could be different.

![Range of orientations.][12]
Accumulating the gradient magnitude of every pixel assigned to an interval to get the histogram.

This calculation is defined by this expression:

\[ h(k) = \sum_{(x,y) \in c} w_k(x,y) g(x,y), \]

where \( k \) is the interval, \( c \) is the cell, \( g(x,y) \) the magnitude and \( w_k(x,y) \),

\[ w_k(x,y) = \begin{cases} 1, & \text{if } (k - 1) \delta \theta < \theta(x,y) < k \delta \theta \\ 0, & \text{if other} \end{cases} \]

2- All histograms calculated are combined to calculate, in turn, the global descriptor of the image. First, it is necessary to normalize due to the image could have different illumination, hence there could be differences in the values of the histograms. A uniform normalization over the image will not be enough because of these changes in the illumination could not be constant, so it will be preferable a local normalization for each zone. To do that, it is used blocks, which are groups of neighbor cells, \( b \times b \) cells. These blocks are defined with certain degree of overlapping. The histograms of each cell are linked and normalized with the L2 norm:

\[ \|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2} \]
Finally, to get to the final descriptor, all of the normalized histograms of the blocks are linked.

![Diagram showing the process of linking normalized histograms](image1.png)

*Figure 22. Getting the final descriptor. [12]*

- **LBP**

LBP method aims to get an associated code to each pixel of the image. This is the process to calculate LBP:

1. Define the pixel’s neighborhood and compare its intensity level with the neighbors. In the example, neighbors are the pixels that are in contact with the central one.

![Diagram showing the LBP process](image2.png)

*Figure 23. LBP Process*

2. Choose a “visit order”, that is, the order it’s going to be followed to compare central bit with a neighbor. In this case, it’s starting with the pixel with value 1 and continues to the right, but it could be different.

3. For each neighbor this rule is applied:
bit value \( b = \begin{cases} 
1 & \text{if neighbor's value} \geq \text{central bit value} \\
0 & \text{if other case}
\end{cases} \)

Once all the comparisons have been made, the binary value obtained is the following:

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
\end{array}
\]

4. The binary value is converted to decimal, in this case the resultant decimal value is 30, which will be the value of the central pixel. This value provides information about itself and the relation with its neighbors.

CHARACTERISTICS EXTRACTION

This stage consists in the acquisition of the user’s face characteristics, that is facial landmarks. There are different methods to get this objective:

- Methods based on facial features. These methods are divided in two types: Low level methods and methods based on facial geometry. Low level methods include edges detection systems or segmentation, among others. The main problem of these methods is the ambiguous information that it usually obtained.

On the other hand, methods based on facial geometry include Local Binary Patters (LBP), applied to specific areas; Gabor wavelets and methods based on deformable templates.

Gabor Wavelets technique is based on a Gabor filters bank, each one with a different frequency and orientation. It allows to obtain information of frequency about a specific area of the image, transforming the face in waves and making easier the characteristics extraction.

Figure 24. Gabor Wavelets. [15]
• Methods based on holistic techniques. These methods work with all the image or specific areas using all the image as a pattern. There is an outlined method, Principal Components Analysis (PCA), which is a technique that search the reduction of the number of variables in an environment without losing too much information. This technique derives others as Linear Discriminant Analysis (LDA) and Fisher Linear Discriminant (FLD).

• Hybrid methods. These methods combined the two previous and include, among others, Active Appearance Model (AAM), Active Shape Model (ASM) and Elastic Bunch Graph Matching (EBGM).

AAM corresponds to a statistic model created in a training phase which is made of images that include points situated on representative locations of a human face. With these points it’s created a 2D triangular mesh by doing the average of each one of the training images meshes.

ASM is also a statistic model used for the representation of deformable objects. Each shape is represented by a set of characteristics points that are looking for individually first and then it’s applied PCA to make the object deform in a unique way.

On the other hand, EBGM is a method based on graphs theory. In the same way as AAM, it is created a statistic model in a training phase but this time it’s made from a small collection of simple image graphs. The generation of the graphs is based on Gabor filters explained previously and the recognition of characteristics is grounded on the comparison of the graphs.

(a) (b)

Figure 25. Hybrid methods; a) ASM [16]; b) AAM [17].
- Boosting methods. These methods are built in series with sequential models which weights are adjusted based on the learning of the previous models. Each one of these models are called weak models that set up a strong model that will provide the decisive output. Two distinguished boosting methods are Adaboost algorithm, described before, and Gradient Boosting, which has been used in this project for the characteristic extraction task.

In particular, the characteristic extraction method used is the one created in 2014 by Vahid Kazemi and Josephine Sullivan named One Millisecond Face Alignment with an Ensemble of Regression Trees [18]. This technique consists in a cascade of decision trees which are used to detect facial landmarks from a sparse set of pixel intensities achieving a high quality in real time. Each decision tree is trained via Gradient Boosting with a squared error loss function which objective is to reduce the error sequentially.

Before explaining how the Gradient Boosting works, it’s necessary to understand the decision trees, and specifically the regression trees. Decision trees are prediction models which, from a set of data, build an output by making decision in different moments called nodes.

![Decision Trees](image)

*Figure 26. Decision trees types. [19]*

Depending on this output, the decision tree could be a regression tree or a classification tree. While the first one output takes continuous value, the second one takes discrete values. As the goal of the algorithm is to predict
Development of an Interaction System based on Face Processing

ETSIST UPM

facial landmarks, Regression tree is selected for the procedure. The following figures shows the differences between both types of decision trees.

![Decision Trees Diagram](image)

*Figure 27. Examples of decision trees; a) Classification tree; b) Regression tree. [19]*

The parameters of each regressor in the cascade are determined by Gradient Boosting method, which used a gradient descent that minimizes the loss function.

The main purpose of this technique is to get the minimum value of the cost function $J(w)$, $w^*$. first, it is necessary to calculate the gradient value from an initial point $w^{(0)}$ applying $\nabla J(w) = \left( \frac{\partial J(w)}{\partial w_0}, \ldots, \frac{\partial J(w)}{\partial w_n} \right)^T$. Then, it’s subtracted $w^{(0)} - \nabla J(w^{(0)})$, which means going down the side of the function. This process is done until reach the minimum value of $J(w)$.

![Gradient Descent Diagram](image)

*Figure 28. Gradient descent*

**Repeat:**

$$\begin{align*}
    w_0^{(k)} &\leftarrow w_0^{(k-1)} - \frac{\partial}{\partial w_0} J(w^{(k-1)}) \\
    & \quad \vdots \\
    w_n^{(k)} &\leftarrow w_n^{(k-1)} - \frac{\partial}{\partial w_n} J(w^{(k-1)}) \\
    & \quad \text{Until convergence.}
\end{align*}$$

**EXPRESSION RECOGNITION SYSTEM**

This is the last phase to get the objective of the program: detect the emotion. To detect the emotions, the classifiers used to accomplish this step are based on Machine Learning algorithms, in particular, in this project it’s been used Support Vector Machines.
2.3. Machine Learning

As it can be seen in figure, Machine Learning (ML) is a subfield of Artificial Intelligence and can’t be confused with Deep Learning, which in turn is a subfield of ML. ML can be defined as “… field of study that gives computers the ability to learn without being explicitly programmed”, explanation given by Arthur Samuel in 1959. This means that ML is the practice of using algorithms to parse data, learn with it and then be able to make a prediction without a human performance. These algorithms generate models based on the input data to produce an output with a set of predictions. In other cases, the model created would be only useful for a determinate problem, therefore for each new requirement needs the programmer to update the software. But this is not the case, considering that the ML model will be able to handle with these new requests without modifying any code.

On the other hand, Deep Learning is a type of Machine Learning inspired by the structure of the human brain and implicates feeding the computer on a model which can evaluate examples and a set of instructions to modify this model when an error occurs. This is very effective in feature detection, but this project is focused on Machine Learning to recognize emotions.

There are different types of Machine Learning, of which most common are:

- **Supervised Learning**: The model receives a set of input data which is mapped with the corresponding output. This is the one selected in this project so in the next section it’s going to be explain in detail.

- **Unsupervised Learning**: The model receives a set of inputs without being labeled, trying to learn from data by exploring patterns of them. The model works on its own to discover information that may not be visible to the human eye.
• Semi-Supervised Learning: Hybrid learning between unsupervised and supervised learning.
• Adaptive Learning: Based on a previous model which parameters are modified using new training data.
• On-line Learning: There is no distinction between training and testing phases. System learns during the prediction process where there is a human supervision validating and correcting each output according to inputs.
• Reinforcement Learning: Hybrid system between on-line learning and semi-supervised. It is based on punishes and rewards, a type of learning based on “argumentum ad baculum”, used in animals’ education.

**Supervised learning**

As said previously, supervised learning consists in a training algorithm that uses a labeled dataset with information about input and output characteristics and has to generate a set of rules which will be used to obtain a coherent output when the input is non-labeled. One typical example of supervised learning is detecting if an email is spam or not, hence to do that, the training set includes emails labeled as “spam” and emails labeled as “not spam” to help the algorithm learn to classify future emails. In a formal way, supervised learning can be described as a system were given a set D of learning examples described with features, X, the goal is to find a function that predicts target variable, Y.

To design a supervised learning algorithm to resolve a specific problem, is required to:

• Define the task. In this step the problem is evaluated, and after that, it is set a determinate database. This is always the first step that must be taken before start designing the algorithm.
• Decide on the machine learning algorithm, which introduces specific inductive bias. This bias is added to every set of characteristics and has a value of 1. It is used to obtain an output different to 0 when value of characteristics is 0.
• Decide on the score or cost function.
• Find a function that describes the relation between X and Y. With the given set of learning examples described with features, X, the goal is to find a function that predicts variable results, Y. \( f(X) \rightarrow Y \)

Regarding the operation system, as machine learning algorithms, there are several steps followed:
1. Gathering and preparing data. First, it's important to have a good database with a considerable quality and quantity of the labeled data, which will determine good results of the model. This data has to be prepared to the training and evaluation phases, what means that has to be normalized with a random order. Data will be divided in two parts: the fist one will be the majority one, used in training phase and the second one used to evaluate the model once it's trained. There are several methods to make the division, two of them are Cross-Validation an Hold-Out.

In Hold-out method a percentage, A%, of the samples are used for training and the remaining percentage, (100-A)% , is used for evaluation. All the samples are divided in two groups, training and test, in a random way, therefore this process can be made several times and then average the evaluations. It is not guarantee the use of all samples.

On the other hand, in Cross-Validation method, it’s selected an integer K which is factor of the number of samples M, that is $\frac{M}{K} \in N$. Once K value is chosen, samples are divided in K subsets of M/K samples. Then, K-1 subsets are trained, and the remaining subsets are used for evaluation. This process is repeated K times to use all the samples, so this method has a costly procedure comparing with Hold-out method.

2. Choosing a model. Depending on the problem to solve, a specific machine learning algorithm has to be used, hence it’s necessary to study the different ML algorithms to choose the correct one and get better results.

3. Training. This step uses the data iteratively to improve the model’s predictions. $y = m*x + b$; where y is the output, m is the slope, x the input and b the y-intercept. In this formula, the values that can be adjust are m and b. There are as many m’s values as features are, so the collection of m values is formed into a matrix, denote W (weights). In the same way for b, which are assembled and named biases. Initially, the process starts with random values for W and b trying to predict output with those values. Then these predictions are compared with the outputs that should be produced and W and b values are adjusted to have more correct predictions.

4. Evaluation. Once training phase is finished, the model has to be evaluated. This step allows to test the model with data that has not be used for training and how it will response to other new real inputs in a future.
5. Parameter adjustment. The evaluation shows how precise is the model and if it’s necessary to modify something in previous steps, such as the ones used in training set or the number of iterations done during the process. These parameteres are named “hyperparameters” and their adjustment is more an experimental process that depends on the requirements of the dataset, model or training process.

6. Prediction. When the model is totally trained and evaluated with a significant quality, it would get the predictions and solve the initial problem.

**SUPPORT VECTOR MACHINE**

The SVM is a linear classifier based on determining the maximum margin between two classes from certain vectors called support vectors. As the SVM is a linear classifier the decision limit will be a hyperplane which divides the space of characteristics in two different regions.

*Figure 30. Example of SVM*

**How to find the limit line or plane?**

The separation limit is found from the samples. Not all the samples are used to do that, only a limited number of vectors are considered, and they are called support vectors. These samples are selected to make the distance between the hyperplanes which contain the support vectors of both classes be maximum.
The objective is to optimize this maximum margin, therefore the SVM solution provides the major distance between classes, free of training samples. If samples which are not support vectors are moved, the solution does not change. This fact offers statistic robustness and makes the model robust to overfitting. On the other hand, approximating support vectors reduces overfitting.

**Set of samples not linearly separable**

Most part of the real cases are not separable in a linear way, so this option must be studied to get an enough efficient model.

![Figure 31. Example of samples not linearly separable](image)

To make the SVM be efficient in these cases, there are 2 alternatives:

- It is permitted certain overlapping between classes by relaxing the margin condition. This technique implicates an error tolerance in which there are vectors violating the margin condition. It allows SVM to be robust to noise associated to these samples and to work with sets not linearly separable. The tolerance is controlled by slack variables and has a regulation factor which looks for a compromise with the error tolerance.

- The model is spread in a way it is possible to work with these type of sets, hence the space of characteristics that is not linearly separable is turn into another space that is linearly separable. This technique is called **kernel trick**.
In this case, it is not necessary to define the mapping function, only the scalar product of the kernel function, \( f(x) = \text{sgn} (\sum y_i \alpha_i k(x, x_i) + b) \), which can be one of those defined in the table.

**Table 1. Kernel types.**

<table>
<thead>
<tr>
<th>Kernel type</th>
<th>Scalar product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>( k(x, z) = \langle x, z \rangle )</td>
</tr>
<tr>
<td>Polynomial</td>
<td>( K(x, z) = \langle x, z \rangle^d )</td>
</tr>
<tr>
<td>Radial Basis Function</td>
<td>( k(x, z) = e^{-|x-z|^2 / 2\sigma} )</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>( k(x, z) = \tanh(\kappa(x, z) - \delta) )</td>
</tr>
<tr>
<td>Inverse of Multi-quadratic function</td>
<td>( k(x, z) = \left(|x - z|^2 / 2\sigma + c^2\right)^{-1} )</td>
</tr>
<tr>
<td>Intersection kernel</td>
<td>( k(x, z) = \sum_{i=1}^{n} \min(x(i), z(i)) )</td>
</tr>
</tbody>
</table>
**Multiclass classifier**

A multiclass classifier differs from a binary classifier in the number of classes to classify. As in a binary case, each sample is assigned to one label. As the SVM classifiers only support binary classification, the solution to these problems is to decompose them in binary classification problems. To do that, there are two different methods:

- **One vs the rest.** This method consists in making a classifier for each class to classify, that is, the class is fitted against the rest of the classes. This is the most used method.
- **One vs one.** This method consists in making a classifier for each pair of classes. In this case, the class receives votes when predicting and the class with the most votes is selected. If there is a draw between two classes, it selects the class with the highest confidence by summing the confidence levels per pair calculated in each binary classifier.

**Selection of kernel parameters**

To find the correct margin hyperplane, it is necessary to adjust parameters $\alpha_i$ and $b$, but there are other parameters which kernel function may depend on called hyperparameters. In this project, hyperparameters modified are defined in the table below.

<table>
<thead>
<tr>
<th>Kernel type</th>
<th>Scalar product</th>
<th>Optimization parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>$k(x, z) = \langle x, z \rangle$</td>
<td>$C$</td>
</tr>
<tr>
<td>Polynomial</td>
<td>$K(x, z) = (\gamma \langle x, z \rangle + r)^d$</td>
<td>$C, \gamma, d$</td>
</tr>
<tr>
<td>Radial Basis Function</td>
<td>$k(x, z) = e^{-\gamma |x-z|^2}$</td>
<td>$C, \gamma$</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>$k(x, z) = \tanh(\gamma \langle x, z \rangle + r)$</td>
<td>$C, \gamma$</td>
</tr>
</tbody>
</table>

Explanation. $C$: soft-margin constant, $\gamma$: width Gaussian kernel, $d$: degree polynomial kernel, $r$: coefficient.
Beginning with the optimization parameter which is common to all kernel functions, C corresponds to the soft margin constant. The behavior of this parameter is observed in the Figure 33. As seen, a smaller value of C increments the margin, so points closed to the limit are ignored and becomes margin errors. While C value is increasing, the margin gets thinner thus the system has more freedom to select more samples as support vectors.

![Figure 33. Effect of the soft margin constant value. [21]](image)

On the other side, degree of the polynomial kernel, d, and width parameter of the Gaussian kernel, γ, have a relevant effect on the decision limit. The first one performance is shown in Figure 34, which demonstrates that the polynomial degree affects the flexibility of the decision boundary, therefore a higher degree implicates a more flexible limit.

Degree 1 of the polynomial kernel corresponds to a linear kernel, which is not enough when the relation between samples is not linear. In the example, a polynomial degree 2 would be sufficient to classify the features.

![Figure 34. Effect of the degree value](image)
The second parameter, $\gamma$, is the inverse of the radius of influence of the samples selected by the model as support vectors. The system is very sensitive to the behavior of this parameter. If $\gamma$ value is high, the area of influence of the support vectors will be reduced so it will include the support vector itself and, although $C$ is regularized, it will be overfitting, which can be seen on the two pictures at the bottom of the Figure 35. On the opposite case, if $\gamma$ is very small, the model will be limited and won’t catch properly the shapes of the data. The area of influence of the support vectors will include all the training set, hence the model behaves such as a linear model with a set of hyperplanes which will separate the centers of high intensity of any pair of classes. This last behavior is appreciated on the picture at the upper left of Figure 35.

Those smooth models where $\gamma$ is lower, can be transform to more complex models by increasing $C$ value, therefore, selecting more support vectors. With intermediate $\gamma$ values it is obtained models which behavior is the same when $C$ becomes to have higher values, therefore it is better to use the lower $C$ value that secures goods results so that models will use less memory and will be faster to predict.
Model selection

The type of kernel is dependent to the data the system has, hence it is necessary to try with several kernels. It’s usually to start with linear kernel tuning C values and then see if accuracy can be improved by changing to a nonlinear kernel, such as polynomial or RBF. Linear kernels generally provide good results and, since there is only one parameter to adjust is easier to tune. Furthermore, RBF and polynomial kernels often have more probabilities to overfit. Once tried with linear kernels, it can be used as baseline for an improvement with a nonlinear kernel.

As the SVM decision limit depends on the hyperparameters, the classifier accuracy is also dependent to them. In the case it is using a linear kernel there is only one parameter that has to be tuned, the soft-margin constant, C. But in the case of the polynomial or the RBF kernel, there are two parameters to be regularize, so the best way of selecting the best values of these hyperparameters is via grid-search, which is the standard method of exploring a two-dimensional space. The figure below is an example of the classifier cross-validation accuracy as a function of C and γ. The grid points are selected on a logarithmic scale, usually is enough a logarithmic grid between $10^{-3}$ and $10^{3}$ but can be extended as in this case. Once the accuracy is estimated and represented in the grid, it can be chosen the best values of the hyperparameters, C and γ in the example, and the model can be trained.

![Validation accuracy](image)

*Figure 36. Example of a grid-search. [22]*
3. Datasets

3.1. The extended Cohn-Kanade Dataset
This dataset is an extension of the Cohn-Kanade Dataset [23], created as a consequence of the limitations and difficulties the first model presented, which has been used for AU and emotion detection. One of these difficulties were due to emotions associated to each image of the database didn't coincide with the emotion given in the image.

The first CK database includes 486 sequences of 97 subjects. Each of these sequences contains images from the beginning expression, that is a neutral expression, to the last frame containing the peak expression. As CK dataset, CK+ peak expression is FACS coded but emotions labels were revised and validated with reference to the FACS Investigators Guide. The results are based on the experiments made with the Active Appearance Model (AAM) and the Support Vector Machine (SVM) system.

Extended Cohn-Kanade database added 107 sequences to the 486 already existed besides another 26 subjects, in total 593 sequences from 123 subjects, who are 18 to 50 years, 69% female, 81% Euro-American, 13% Afro-American and 6% other groups. With these images the CK+ was labeled according to the FACS coded emotion labels. There are 3 steps in the selection process:

1- Images are compared with the Emotion Prediction Table from the FACS manual, which lists AU combinations and variants of each emotion, except contempt. If a sequence satisfies the criteria for a specific emotion, it is provisionally coded as belonging in that emotion category.

Table 3: Emotion description in terms of facial action units. [23]

<table>
<thead>
<tr>
<th>Emotion</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angry</td>
<td>AU23 and AU24 must be present in the AU combination</td>
</tr>
<tr>
<td>Disgust</td>
<td>Either AU9 or AU10 must be present</td>
</tr>
<tr>
<td>Fear</td>
<td>AU combination of AU1+2+4 must be present, unless AU5 is of intensity E then AU4 can be absent</td>
</tr>
<tr>
<td>Happy</td>
<td>AU12 must be present</td>
</tr>
<tr>
<td>Sadness</td>
<td>Either AU1+4+15 or 11 must be present. An exception is AU6+15</td>
</tr>
<tr>
<td>Surprise</td>
<td>Either AU1+2 or 5 must be present and the intensity of AU5 must not be stronger than B</td>
</tr>
<tr>
<td>Contempt</td>
<td>AU14 must be present (either unilateral or bilateral)</td>
</tr>
</tbody>
</table>
2- A less accurate comparison is performed in this step, meaning if an AU is not included in the listed variants, it is determined if they are consistent with any of the emotions.

3- The third step consists in determine if the expression in the images coincides with the emotion detected. This step is necessary because the FACS code only describes the expression at the peak image of the sequence without considering the facial changes before that peak expression.

As a result of this process, 327 of the 593 sequences were found to meet criteria for one of seven discrete emotions. This result is shown in the table presented below.

<table>
<thead>
<tr>
<th>Emotion</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Angry</td>
<td>45</td>
</tr>
<tr>
<td>Contempt</td>
<td>18</td>
</tr>
<tr>
<td>Disgust</td>
<td>59</td>
</tr>
<tr>
<td>Fear</td>
<td>25</td>
</tr>
<tr>
<td>Happy</td>
<td>69</td>
</tr>
<tr>
<td>Sadness</td>
<td>28</td>
</tr>
<tr>
<td>Surprise</td>
<td>83</td>
</tr>
</tbody>
</table>

Regarding the baseline system, as it is said before, all sequences are AAM tracked with 68 points landmarks for each image. Then, SVMs are used to classify the facial expressions and emotions.

![Block diagram of the system](image)
CK+ DATASET CONTENT

The CK+ database contains four and a ReadMe document that explains all the content:

1. The images. There are 593 sequences of 123 subjects which are FACS coded at the peak frame. All of the sequences start with the neutral face and ends with the peak expression.
2. The landmarks. All sequences are AAM tracked with 68 points landmarks for each image.
3. The FACS coded files. For each sequence there is only 1 FACS file, which is the last frame. Each line of the file corresponds to a specific AU and then the intensity.
4. The emotions coded files. Only 327 of the 593 sequences have emotion sequences due to these are the only ones that fit the prototypic definition. As the FACS files, there is only 1 emotion file for each sequence which is the last frame. There should be only one entry and the number will range from 0 to 7.

Table 5. Emotions and examples of the dataset.

<table>
<thead>
<tr>
<th></th>
<th>Emotions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Neutral</td>
</tr>
<tr>
<td>1</td>
<td>Anger</td>
</tr>
<tr>
<td>2</td>
<td>Contempt</td>
</tr>
<tr>
<td>3</td>
<td>Disgust</td>
</tr>
</tbody>
</table>
Table 6. Emotions and examples of the dataset.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Fear</td>
<td>![Fear Image]</td>
</tr>
<tr>
<td>5</td>
<td>Happy</td>
<td>![Happy Image]</td>
</tr>
<tr>
<td>6</td>
<td>Sadness</td>
<td>![Sadness Image]</td>
</tr>
<tr>
<td>7</td>
<td>Surprise</td>
<td>![Surprise Image]</td>
</tr>
</tbody>
</table>

To understand this, this document gives an example:
All the file name and structure should be the same. For example, an image at:
cohn-kanade-images/S005/001/S005_001_00000011.png
This file will have the corresponding landmark at:

```
Landmarks/S005/001/S005_001_00000011_landmarks.txt
```

**Figure 39. Example of landmarks file**

FACS code at: `FACS/S005/001/S0005_001_00000011_facs.txt`, which has

| AU9d | 9.0000000e+00 Â 4.0000000e+00 |
| AU17b | 1.7000000e+01 Â 2.0000000e+00 |

**Figure 40. Example of FACS code**

Note: if an AU is present but the intensity is 0 this means that the intensity was not given, e.g. 1.20000000e+00 0.000000000e+00, just means AU12 (intensity not given).
Emotion code at: Emotion/S005/001/S005_001_00000011_emotion.txt, which has 3.00000000e+00, that is disgust.
4. Implementation

4.1. Program design

The algorithm is based on a ML classifier which use several databases made up of images of people who are called participants. In this project two possible solutions have been selected. One of them uses a multiclass SVM classifier which classifies emotions directly and another one that uses a binary SVM classifier which identifies an AU from a neutral face. This is the final technology used in both solutions in this project but not the only one. These are the steps made to finally reach this point in the development.

FIRST STEPS

Before starting the design and implementation of the final algorithm, it was required to study how OpenCV works by testing some of its functions that would be implemented in the principal program.

The first testing program, Figure 43 in the left, consisted in a simple application that opens the webcam and detects the face putting it in a rectangle. The second one, Figure 43 in the right, added to the first program the detection of both eyes, drawing two points on the upper corners of the left eye rectangle. Both programs were made to have the first contact with OpenCV testing Haar Cascade models and VideoCapture functions which are also used in the final algorithm. Apart from that, it was the first contact with image processing in Python, transforming a RGB image to a grey image.

Once Haar Cascades were not a problem, it was the time to use dlib library, which would be very useful lately. Figure 44 shows the goal of this algorithm: identify facial landmarks. The reason why it was better to use dlib is that this library has a face detector and a shape predictor function which makes the task easier.
At first, and comparing both systems to recognize faces, it seems to be dlib functions the best option, but finally they worked together in different parts of the final algorithm.

Figure 44. Drawing landmarks

The following step was to think about how can be implemented the emotion recognition function.

The first idea before using Machine Learning algorithms was to use Haar Cascades in the same way it was used to recognize faces, so as there is one Haar Cascade file to recognize a smile, happiness and sadness emotions were the first ones to be detected. As this algorithm works pretty good, the intention was to create some Haar Cascade for each emotion, but looking for information about that, the conclusion was it would take too much time, therefore the idea was discarded.

Figure 45. Detecting face and smile.

The second idea was to play with landmarks coordinates so it was required to study the movements of the face and therefore landmarks movements. In Figure
46 is shown the operation of the program based on this study. It’s seen that the algorithm works in a right way if the face is centered but, in the case of the fourth picture where the face has a different angle, the detection is wrong.

![Figure 46. First expressions detection app; a) right detection of blinking; b) Right detection of smiling; c) right detection of a left blink; c)Wrong detection of blinking.](image)

It is due to each expression is based on the position of determinate landmark coordinates respect other points which may change when the face is turned. For example, in the case of the smile detection, shapes 48 and 66 and shapes 54 and 66 are compared. An extract of the code related to this detection is shown below.

```python
elif ((shape[48])[1]<((shape[66])[1])-3) & ((shape[54])[1]<((shape[66])[1])-3):
    cv2.putText(frame, "Smile", (450, 380), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
```

![Figure 47. Extract of the code.](image)
In this situation, if the face position is such as the one shown in Figure 46 d, smile will never be detected because coordinates of shape 54 (landmark 55 in Figure 48) may not be lower than coordinates of shape 66 (landmark 67 in Figure 48).

Another inconvenient of this design is that the operation worked good enough to control eyes tracking with one person but due to each person has different eye shape it wasn’t a good way to get to the main objective.
Finally, and after some researches, the best solution was to apply Machine Learning algorithms.

PREPARING THE DATASET
The next step after studying algorithms and making a design of the application was preparing the dataset. Data used in this is CK+ database that has been explain before. As there are two different solutions, it was necessary to prepare two datasets:

- **Solution based on emotions**
  First, images and emotions folders from CK+ database are situated in the working directory. At the same time, another folder divided in turn in 8 empty folders, corresponding to each emotion, is created in the same directory. Once everything is ordered, each emotion folder is filled with the last frames of participants’ sequences considering their corresponding emotion code. Neutral folder is filled with all the first frames of the sequences.
  The images contained in the new folders have different formats, so the next step is standardizing them to include these images in the dataset that is going to be used in the algorithm.
  As the algorithm will get the landmarks, it’s easier to have a dataset of images that only have faces without a background, hence the first stage is to detect faces in the images.
To detect faces on the participants’ images it’s used HaarCascade classifiers, in particular, five frontal face detectors: `haarcascade_frontalface_default`, `haarcascade_frontalface_alt2`, `haarcascade_frontalface_alt`, `haarcascade_frontalface_alt_tree`, `haarcascade_eye_tree_eyeglasses`. Before applying the classifiers, each frame is converted to grayscale and after the detections, all of them are cut to size of the rectangle containing the face. Once the monochromatic image has been cut, it is resized for having all the images with the same size and therefore all of them have the same format, in this case 350x350. Then, each group of images belonging to an emotion is saved in its respective folder in the final dataset.

- **Solution based on AUs**
  In this case, the procedure is almost the same, but instead of situating CK+ emotions folder in the directory it’s used FACS folder. The folder that has to be created is divided in 41 folders that correspond to the number of AUs that appear in the participants’ images. Once the image processing required is done, each group of images belonging to a determinate FAC is saved in its respective folder in the final dataset. Differing from the previous solution, this dataset hasn’t got 41 folders, it has 18 folders which correspond to AUs used to detect the emotions by following the instructions in CK+ document.

**IMAGE PROCESSING**

The steps to prepare images from the dataset are the following ones:

1. **Equalize images.**
   
   As images were kept in the dataset with a standardized format, it is not necessary to convert to greyscale or resized them. In the case of images coming from webcam, they have to be standardized in the same way as did with dataset images. Once they are normalized an equalization is required, so, to do that, a CLAHE (Contrast Limited Adaptive Histogram Equalization) object from OpenCV library is created and it is applied to each charged grey image. This type of histogram has been selected because with a global equalizer it is possible to lose information due to over-brightness, thus it is better to apply histograms in particular regions of the image.
   
   The pictures below show the equalization process. First image corresponds to the input without any modification; in the second one appears the detected
face on the first image turned into greyscale and resize to 350x350, which is the standard size; and the third one is the equalized image where it can be appreciated the increase in contrast that will allow the algorithm to predict better where are the specifics points to get the landmarks in the second step of the process.

![Image processing]({})

(a) Original image; b) Resized Grey scale image; c) CLAHE image

2. Get landmarks.

First, it is required to detect faces in the dataset images, so the frontal face detector is used for each image. Once faces are detected, facial landmarks are draw using the shape predictor class for all of them.

Shape predictor class is a tool that has as input an image section with an object contained and its output is a collection of points that define the pose of this object. In this case the input is an image where appears a human face and the objective is to identify the coordinates of the most important face landmarks such as eyes, eyebrows, mouth or nose. To achieve that, it’s needed a pretrained model available in dlib library, shape_predictor_68_face_landmarks.dat. This model was made by using dlib’s implementation of the paper created by Vahid Kazemi and Josephine Sullivan and was trained on the iBug 300-W face landmark dataset.
3. Vectorize landmarks

Figure 50. 68 facial landmarks. [24]

Figure 51. Captures from the application; a) Angle of the vector between central point and one landmark; a), b) and c) are different captures of a face in several positions.
As seen in the pictures above, coordinates might change as the captured face in the webcam moves, so the person could show the same expression while being on the top right or on the bottom left of the image but the value of the coordinates change. Nevertheless, the relationships between the different points are invariant, thus, the best way to work with landmarks is to calculate the relative position among them. To do that, it is calculated first the center of gravity and then, the distance and the angle between the rest of the shapes and this point. The process done by the designed algorithms is described next:

- In first place, X and Y landmarks coordinates have to be stored in two lists. Then, coordinates of center of gravity are found by obtaining the mean of X and Y.
  
  In the case of the multiclass classifier, it stores all the landmarks, that is 68 points. With the binary solution the number of landmarks depends on the AU to be detected, for example, to identify AU4 it’s only required to store eyebrows landmarks.

- After that, it is calculated the distance between gravity center and the rest of the shapes in both axes. When working with multiclass classifier solution, this center of gravity corresponds to the one shown in Figure 51, but with the binary solution the point changes depending on the AU to be predicted. It is due to the movements made by each part of the face. In some cases as shown in Figure 52, it is better to calculate gravity center considering only the landmarks belonging to the AU, but in other cases it’s better to calculate the gravity center of the whole face and then store the suitable points.

![Figure 52. Gravity center considering AUs](image)
Once all distances have been estimated, it is created an empty vector which will store the vectorized landmarks. This array is formed by the direction and the angle of the vector, as well as each coordinate X and Y.

- To get the distance it is used function linalg.norm which returns the norm of a vector or a matrix. In this case it is applied to the difference between the coordinates of the central shape and the desirable point.

\[
Distance = \sqrt{\text{mean\_coordinates\_shape}^2 - \text{coordinates\_other\_shape}^2}
\]

- To calculate the angle, it is used function math.atan2, which calculates arctangent in radians between \(-\pi\) and \(\pi\). The result of this function has to be divided by \(\pi\) and multiply by 180 to get the angle in grades.

\[
Angle = \left(\tan^{-1}\left(\frac{\text{ymean\_shape}}{\text{xmean\_shape}}\right)\right) \times 180 / \pi
\]

4. Store landmarks.

It is created an empty dictionary of values that will be fill with the vectorized landmarks.
The required steps to train the SVM model are described next:

- The primary step in the training process is loading files from the dataset and classify them in two sets in a random way. This division is made following the Hold-out method: the first set is made up of 80% of the input data, which corresponds to the training set, and the second one will be made up of the remaining 20% of data, that corresponds to the test set.

- It is created two lists for each set, one for data (training_data and prediction_data) and one for labels (training_labels and prediction_labels). Data list will be filled with vectorized landmarks and labels list with the corresponding emotion or AU depending on the algorithm. Before filling these lists, it is applied the image processing explained previously to all images contained in each set. Then, vectorized landmarks are stored in the data lists and their related label in the labels lists. Once the four lists are completed with the suitable information, the training process is initialized.

- On the one hand, training data set and training labels set are turned into arrays so that the classifier will use them to train the model using fit function from Scikit-learn library. Depending on the type of classifier that has been defined, the training will be different. In this case, this phase corresponds to a SVM training explained in previous sections.
- On the other hand, the test data set and the test labels set are turned into arrays so that the classifier will use them to get the accuracy of the model using score function from Scikit-learn library.
- This process is done iteratively with the objective of achieve the best results, thus, the more it is repeated the more it will get better accuracies. In this case, the number of iterations was fixed to 10 due to the time it takes to complete one. Once all the iterations are finished, the model is stored therefore it can be used later to predict emotions in the main program.

EMOTION PREDICTION

As there are two different solutions, the processes to run the created models and predict the emotions have their similarities but there are some differences between them. In Figure 54 is shown the general diagram of the app which share both solutions.

1. First, the ML models are loaded. On the one hand, the solution based on emotions only loads one model which has been trained considering all the emotions as explained before. on the other hand, the solution based on AUs loads a model for each required AU, in this case it’s needed 17 AUs as described in Figure 2 and Table 3.
2. The next step is to start the webcam and get each frame of the video to process it and obtained facial landmarks.
3. Once landmarks are found they are stored in arrays that will be the input of the model predict function.
4. To predict values there are two predict functions: predict and predict_proba. The first one performs the classification of the samples and gets the emotion that the
user may show. `Predict_proba` computes the probabilities of each emotion to be the one expressed by the user.

Solution based on emotions predicts the right emotion with a unique call to `predict` function, whereas solution based on AUs requires to call this function as many times as AUs and models there are. Then, depending on the AUs predicted, the algorithm will select as a solution one emotion or another. For example, if AU6, AU12 and AU25 are active the emotion will be happiness.

5. The last step is to show results in the window. The first solution shows the probabilities of each emotions and the selected one as seen in Figure 55 left. The second solution shows the active AUs and the selected emotion as can be observed in Figure 55 right.

![Figure 55. App solutions; a) Solution based on emotions; b) Solution based on AUs.](image)
4.2. Experiments and results

4.2.1. Support Vector Classification

MULTICLASS CLASSIFIER BASED ON EMOTIONS

Selection of kernel parameters
To test the SVM model it was used the SVC four types of kernel functions: linear, polynomial, radial basis function and sigmoid. Scikit offers the possibility to defined customized kernels, but because of time it wasn’t tested. For all the experiments the number of iterations was 10.

The first kernel to test was, as it’s recommended in Machine Learning algorithms, linear kernel. The parameter which can be modified is C, hence the first value selected was the default one C=1.0.

As seen in the Table 7, C value was increased until 10000, obtaining similar accuracies. Attending to results, changing C value does not improve quality too much, so it is enough with C=1.0.

<table>
<thead>
<tr>
<th>Linear kernel</th>
<th>C</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0.849</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.828</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.843</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>0.833</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>0.837</td>
</tr>
</tbody>
</table>

As the objective was to reach a classifier with the best accuracy, the next step was trying with a nonlinear kernel and see if the accuracy could be improved, so the following kernels to test with was polynomial. In this case, apart from C value, degree parameter was modified, thus, the best way to see which values provide the best results was via grid-search. The graph attained is shown below.
The results obtained show that there is not a significant improvement in the accuracy changing linear kernel to polynomial, so at this moment linear kernel was still being the best option. The next step was to test the classifier with RBF kernel. The parameters which could be modified in this kernel are C value again and γ. The process was the same as the polynomial kernel.
As it can be seen in the graph obtained, there is an improvement on the accuracy compared to results with polynomial kernel, getting a maximum accuracy of 0.84 with $C=10000$ and $\gamma=1e^{-6}$. As in the case of polynomial kernel, there is no important differences between RBF and linear kernel, therefore the next step is trying to achieve better accuracies with a sigmoid kernel.
The sigmoid kernel doesn’t provide higher accuracies, getting the best one with a value of 0.84 with $C=100$ and $\gamma=0.001$.

In view of the results, sigmoid and RBF are more accurate than polynomial which gets worst results. However, comparing them with linear kernel there are no big differences, so, as well as linear kernel is simpler than the others, its results are good enough to make a good classifier, so this option was the one selected to be used in the final algorithm.

**Testing the classifier**

Before starting to create the final app interface, it was made a script to test the SVM classifier with the linear kernel. When running the script a window is created and, at the same time, webcam starts to film the user’s face, which is shown in the window. Apart from the image, all of the emotions with their respective probabilities appear on the right side of the image and the emotion predicted, that is with the highest probability, appears on the left side.

<table>
<thead>
<tr>
<th>Emotions</th>
<th>Percentage of right classifications (%)</th>
<th>Wrong classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neutral</td>
<td>90</td>
<td>-</td>
</tr>
<tr>
<td>Anger</td>
<td>40</td>
<td>Disgust</td>
</tr>
<tr>
<td>Contempt</td>
<td>95</td>
<td>Neutral</td>
</tr>
<tr>
<td>Disgust</td>
<td>100</td>
<td>-</td>
</tr>
<tr>
<td>Fear</td>
<td>20</td>
<td>Disgust</td>
</tr>
<tr>
<td>Happiness</td>
<td>99</td>
<td>-</td>
</tr>
<tr>
<td>Sadness</td>
<td>30</td>
<td>Disgust</td>
</tr>
<tr>
<td>Surprise</td>
<td>80</td>
<td>Sadness</td>
</tr>
</tbody>
</table>

As seen in Table 8, the classifier has not an accuracy of 100%, there are cases where classifications are wrong. In view of the results, fear, sadness and anger are the three emotions with less number of right classifications, 20%, 30% and 40% respectively. This fact could be better by training again the classifier with a larger dataset or changing the number of iterations.

To test the algorithm, different people, angles and light were required. The pictures below show some of the results obtained, both right and wrong classifications.
This face shows a sadness expression and the algorithm predicts sadness with a probability of 26.6%, hence the classification is correct.

The user expresses surprise and the expression predicted is surprise with a probability of 32.6%, so the classification is right.

The user's face indicates anger but the emotion selected is disgust with a probability of 94.5%, hence the classification is wrong.
This face express surprise but it is predicted disgust with a probability of 28.4%, therefore the classification isn’t correct.

The user expression indicates sadness but the emotion selected is contempt with a probability of 49.1%, so the classification is wrong.

The user shows happiness and the emotion predicted is happiness with a percentage of 99.4%, hence the classification is right.
BILINEAR CLASSIFIER BASED ON FACS

Selection of kernel parameters
Considering the results obtained with linear kernel and C=1.0, presented in Table 9, it was not required to test other kernel or parameters due to the high accuracies.

<table>
<thead>
<tr>
<th>AU</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inner brow raise (AU1)</td>
<td>0.860</td>
</tr>
<tr>
<td>Outer brow raise (AU2)</td>
<td>0.907</td>
</tr>
<tr>
<td>Brow lower (AU4)</td>
<td>0.877</td>
</tr>
<tr>
<td>Upper lid raise (AU5)</td>
<td>0.914</td>
</tr>
<tr>
<td>Cheek raise (AU6)</td>
<td>0.974</td>
</tr>
<tr>
<td>Lid tighten (AU7)</td>
<td>0.918</td>
</tr>
<tr>
<td>Nose wrinkle (AU9)</td>
<td>0.979</td>
</tr>
<tr>
<td>Upper lip raise (AU10)</td>
<td>0.969</td>
</tr>
<tr>
<td>Lip corner pull (AU12)</td>
<td>0.967</td>
</tr>
<tr>
<td>Dimple (AU14)</td>
<td>0.941</td>
</tr>
<tr>
<td>Lip corner depress (AU15)</td>
<td>0.886</td>
</tr>
<tr>
<td>Chin raise (AU17)</td>
<td>0.834</td>
</tr>
<tr>
<td>Lip stretch (AU20)</td>
<td>0.966</td>
</tr>
<tr>
<td>Lip tighten (AU23)</td>
<td>0.897</td>
</tr>
<tr>
<td>Lip press (AU24)</td>
<td>0.932</td>
</tr>
<tr>
<td>Lips part (AU25)</td>
<td>0.925</td>
</tr>
<tr>
<td>Jaw drop (AU26)</td>
<td>0.964</td>
</tr>
</tbody>
</table>

Testing the classifier
Following the same procedure as in the previous case, a script was made to test the SVM classifiers with the linear kernel. In this case, the window does not show any probability. Instead of this information and depending on the emotion detected, which is also displayed, it is shown the different AUs activated in each case on the right side of the image.

The difference between this algorithm and the previous one is that, in this occasion, AUs are predicted with the classifiers and emotions are selected from these AUs, while the other one predicts emotions directly. For this reason, before making a script with all the
AUs’ classifiers and emotions, it was needed to test separately different groups of AUs corresponding to each part of a face (eyes, mouth, brows…) and then join them into one script to predict the emotions. In spite of the different tests that the application goes through, there are wrong classifications. The Table 10 confirms this circumstance by showing the percentage of right classifications. Now, anger, disgust and sadness are the emotions with less number of right classifications but it could be solved by training again the classifiers which affects to AUs implicated in the expression of the emotion.

Table 10. Percentage of right classifications with AU solution.

<table>
<thead>
<tr>
<th>Emotions</th>
<th>Percentage of right classifications (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neutral</td>
<td>100</td>
</tr>
<tr>
<td>Anger</td>
<td>20</td>
</tr>
<tr>
<td>Contempt</td>
<td>99</td>
</tr>
<tr>
<td>Disgust</td>
<td>50</td>
</tr>
<tr>
<td>Fear</td>
<td>89</td>
</tr>
<tr>
<td>Happiness</td>
<td>98</td>
</tr>
<tr>
<td>Sadness</td>
<td>40</td>
</tr>
<tr>
<td>Surprise</td>
<td>90</td>
</tr>
</tbody>
</table>

To emphasize this fact, the next pictures show some examples of right and wrong classifications with different users, light and angles.

Action Units predicted:
- Dimple

Action Units activated:
- Brow lowered
- Lip tighten
- Lip press

The user expresses anger but the emotion selected is contempt, therefore the classification is wrong.
Development of an Interaction System based on Face Processing

**Action Units predicted:**
- Brow lowered
- Inner brow raised
- Lip corner depress

**Action Units activated:**
- Brow lowered
- Inner brow raised
- Lip corner depress

The user express sadness which is the emotion selected, so the classification is correct.

**Action Units predicted:**
- Outer brow raised
- Inner brow raised
- Lips part

**Action Units activated:**
- Nose wrinkled
- Upper lid raised
- Lips part

The user express disgust but the emotion predicted is fear, hence the classification is wrong.

**Action Units predicted:**
- Cheek raised
- Lip corner pull
- Lips part

**Action Units activated:**
- Cheek raised
- Lip corner pull
- Lips part

The user express happiness that is the emotion displayed, therefore the classification is right.
Action Units predicted:
- Upper lid raised
- Jaw drop

Action Units activated:
- Upper lid raised
- Jaw drop

The user express surprise that is the emotion selected, so the classification is correct.

Figure 69. Example of a right classification - Surprise

Action Units predicted:
- Brow lowered
- Lip tighten
- Lip press

Action Units activated:
- Nose wrinkled
- Upper lid raised
- Lips part

The user express disgust but the emotion predicted is anger, so the classification isn’t correct.

Figure 70. Example of a wrong classification - Angry
5. Conclusions

This project has dealt with Machine Learning and Computer Vision by applying them to the design of an algorithm with the objective of detecting the emotions expressed in real time by a user whose face is captured through a PC webcam. First, the methods were defined. There were three parts in which the algorithm could be clearly divided: faces detection, characteristics extraction and emotion classification.

After this division in parts of the algorithm, it was made a research about the possible solutions, starting with a management of the landmarks coordinates and ending with an optimal solution based on ML, delving into SVM for classification, HOG + linear classifier techniques for face detection and boosting methods for characteristics extraction. Like every supervised ML algorithm, a dataset is needed, so the one that fitted, among others, for the main objective was CK+ dataset. Once the methodology was decided for each part of the algorithm, it was made a study of the different solutions. In this occasion, this project presents two solutions studied and implemented: a solution based on emotions and a solution based on FACS. Both of them uses boosting methods for tracking facial landmarks and CK+ dataset.

The first one uses a multiclass SVM which classifies the seven emotions directly. Before testing, it was subjected to a study of its kernel parameters, with the purpose of improve the accuracy, following the procedure recommended for ML algorithms via grid-search. With this enquiry it was conclude that the linear kernel is enough to get suitable accuracies due to the fact that the differences between the possible kernels weren’t significant. This classifier has a good accuracy and predict the emotions almost in real time but during the testing some difficulties shown up while predicting certain emotions.

The second one uses a binary SVM which classifies AUs based on FACS. In this case, it wasn’t required to try different kernels since the linear kernel provided high accuracies. On the other hand, the classifier does not use all the landmarks when training due to each AU is focus on a specific part of the user’s face, differing from the multiclass SVM. This solution includes one classifier for each AU which is activated in each emotion, so the main program calls all the models created in the training phase, which means a reduction in the runtime speed but an increase in the number of right classifications.

The best solution between the two proposed algorithms depends on the desired aim. In the event that the runtime speed takes the importance over the accuracy, the solution based on emotions will be the best option, but if it’s wanted to be centered on the functional part of the algorithm with high accuracies and number of right classifications the best decision will be the solution based on AUs.
Bibliography


I. Annex

Installation

The IDE used to develop the algorithm was Spyder which is included in Anaconda Navigator, so the first step was to download it from the official website of Anaconda distributor with Python 3.6 version.

Once Anaconda Navigator was installed with all its packages and services (Anaconda Prompt, Jupyter Notebook, Spyder...), it was necessary to install the required libraries used for the development of the algorithm. The installation of the packages is made using pip tool:

*Python pip install package*

The packages used are:

- Numpy
- OpenCV
- Scikit
- dlib

File organization

The project is divided in three folders based on the steps taken during the development:

- First steps. In this folder there are scripts which were allocated towards testing OpenCV and python.
  - Webcam.py. Script that test webcam.
  - Photo.py. Script that takes a photo from webcam and save it in a folder.
  - Attempt1.py. Script which test dlib library and gets landmarks.
  - Attempt2.py. Script that gets landmarks and manage their coordinates to acquire emotions.

- Emotions solution. This folder contains all the scripts and other folders required for developing the solution based on emotions detection.
  - Dataset. This folder is divided in 8 folders corresponding to 7 emotions and neutral face which contain the standardized images from CK+ dataset.
  - Source images. This folder is divided in 123 folders which contain the participants’ images from CK+ dataset.
- Source emotion. With the same division as the previous folder, it contains the emotions code for each participant’s face.

- Sorted set. This folder is divided in the same way as dataset folder and contains the same images with the difference that these images are not normalized.

- Models. Folder that contains the trained models.

- Sources.py. This script divides images from CK+ dataset in emotions and save them in sorted set folder.

- Faces.py. Script which standardizes images in sorted set and save them in dataset folder.

- Train_model.py. Script which trains the SVM model.

- Emotions_cam.py. This script starts the webcam and gets the landmarks. It is called in main.

- Main.py. Displays the user’s face and the predicted emotions.

- PlotKernelParameters.py. This script is used for the selection of kernel parameters.

- FACS solution. This folder contains all the scripts and other folders required for developing the solution based on AUs detection.

  - Dataset. This folder is divided in 18 folders corresponding to 17 AUs and neutral face which contain the standardized images from CK+ dataset.

  - Source images. This folder is divided in 123 folders which contain the participants’ images from CK+ dataset.

  - Source facs. With the same division as the previous folder, it contains the FACS code for each participant’s face.

  - Sorted set. This folder is divided in the same way as dataset folder and contains the same images with the difference that these images are not normalized.

  - Models. Folder that contains the trained models.

  - Sources.py. This script divides images from CK+ dataset in AUs and save them in sorted set folder.

  - Faces.py. Script which standardizes images in sorted set and save them in dataset folder.

  - Train_modelX.py. Script which trains the SVM models. X is the number of the AU wanted to train.
- FACS_cam.py. This script starts the webcam and gets the landmarks. It is called in main.
- Main.py. Displays the user’s face and the predicted AUs and emotions.