Enhancing face recognition by image warping

Jorge Garcia Bueno

Master Thesis in Electronics and Electrical Engineering No.77/2009
1st supervisor: Dr. B. Porr
2nd supervisor: Prof. A.C. Bryce
University of Glasgow, Scotland, 2009
Department of Electronics and Electrical Engineering

April 2009
"People are not remembered by the number of times they fail but for the number of times they succeed"

"Las personas no son recordadas por el número de veces que fracasan sino por el número de veces que tienen éxito"

Thomas Alba Edison, 1927.
This project has been developed as an improvement which could be added to the actual computer vision algorithms. It is based on the original idea proposed and published by Rob Jenkins and Mike Burton about the power of the face averages in artificial recognition.

The present project aims to create a new automated procedure applied for face recognition working with average images. Up to now, this algorithm has been used manually. With this study, the averaging and warping process will be done by a computer automatically saving large amounts of time. Through a clear user interface, the program that has been developed will receive a batch of face images of a person and will create an average picture of them deforming each one of them based on . Some settings (colours, size, etcetera ...) might be edited before any average is created and some options will be offered after the job is done to facilitate the addition of them to a face database. Is is demonstrated in previous studies that the average picture generated contains most of the information of the group of original faces and therefore, a system would recognise this person easily than with any single image.

After the development of the software, a computational study will be done to locate the quality in terms of accuracy and speed of this solution. The program will be asked to learn a batch of faces of a group of people and afterwards it will be tested and compared with actual works to demonstrate if the algorithm is at the same level of quality.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>Abbreviations</td>
<td>x</td>
</tr>
</tbody>
</table>

1 Introduction

1.1 Computer Vision ........................................ 1
1.2 Background ........................................... 1
1.3 Applications for the solution ....................... 2

2 Theoretical Aspects

2.1 Digital Image Processing ............................... 5
  2.1.1 Introduction ...................................... 5
  2.1.2 Colour Spaces ................................... 6
    2.1.2.1 RGB Space .................................... 6
    2.1.2.2 Gray Colour Space ........................... 7
    2.1.2.3 HSV colour Space ............................ 8
  2.1.3 Mapping ........................................... 8
2.2 Image warping ........................................... 9
  2.2.1 Introduction ...................................... 9
  2.2.2 Basic transformations ............................ 10
  2.2.3 Actual algorithms for image warping ............... 11
    2.2.3.1 Transformation with one pair of lines .......... 11
    2.2.3.2 Transformation with multiple pairs of lines ... 13
    2.2.3.3 Triangular warping ........................... 15
    2.2.3.4 Mesh warping .................................. 17
2.3 Objects detection ...................................... 19
  2.3.1 The problem. How to recognise objects? ............ 19
  2.3.2 The solution. Haar Cascades ....................... 19
2.4 Face recognition ....................................... 23
### 2.4.1 The location problem

- 2.4.1.1 Face location through *global* representation
- 2.4.1.2 Face location through *local* representation

### 2.4.2 Algorithms

- 2.4.2.1 Eigenfaces. Principal Components Analysis
- 2.4.2.2 Linear Discriminant Analysis
- 2.4.2.3 Elastic Bunch Graph Matching

### 3 Development of the application

#### 3.1 Language, libraries and environment

- 3.1.1 Programming Language: C++
- 3.1.2 Operative System
- 3.1.3 Graphic User Interface Libraries: Qt4
- 3.1.4 Computer Vision Libraries: OpenCV

#### 3.2 Schema of the application

- 3.2.1 Schema of the application
- 3.2.2 Interface objects and custom objects

#### 3.3 Description of the application

- 3.3.1 Manual average
- 3.3.2 Automatic average

#### 3.4 Warp algorithm

- 3.4.1 Implementation of transformation with multiple pairs of lines
- 3.4.2 Implementation of triangular mesh warping

#### 3.5 Looking for face features

- 3.5.1 Cascade files management
- 3.5.2 Implementation of the Haar Cascades algorithm

#### 3.6 Face recognition

- 3.6.1 Eigenfaces using OpenCV

### 4 Results and Discussion

#### 4.1 Database

- 4.1.1 The importance of warping the images
- 4.1.2 Manual and automatic average faces

#### 4.2 Experiments realized

- 4.2.1 First experiment: Average with included images
- 4.2.2 Second experiment: Average with not included images
- 4.2.3 Third experiment: Database formed with averages. Single faces included
- 4.2.4 Forth experiment: Database formed with averages. Single faces not included
4.2.2.5 Fifth experiment: Comparison between manual and automatic average faces .................................. 53

4.2.3 Automatic single faces and average faces .......................................................... 53

4.2.3.1 First experiment: Study comparing single faces with automatic averages including non-matched faces ........................................... 54

4.2.3.2 Second experiment: Study comparing single faces with automatic averages without non-matched faces ....................................... 57

4.2.3.3 Third experiment: Study comparing single faces with automatic averages without non-matched faces ....................................... 57

4.2.3.4 Forth experiment: Single non-frontal faces Vs. automatic average face .................................................. 57

4.2.4 Time comparisons .......................................................... 59

4.2.5 Evolution of average faces .......................................................... 59

4.2.6 Rejection of faces .................................................. 61

5 Conclusions and future works 62

5.1 Conclusions .................................................. 62

5.2 Future works .................................................. 63

A Cat-Mull Algorithm. Mesh interpolation 64

B HSI-to-RGB Conversion 67

C Schema of the Application 69

D User’s manual 70

D.1 Introduction to the application .................................................. 70

D.1.1 Average face creation .................................................. 70

D.1.2 Database search .................................................. 71

D.2 Manual face creation .................................................. 71

D.2.1 Text menu .................................................. 71

D.2.2 Main options toolbar .................................................. 72

D.2.3 Secondary options toolbar .................................................. 72

D.2.3.1 Single node administration .................................................. 72

D.2.3.2 Group of nodes administration .................................................. 73

D.2.3.3 All nodes administration .................................................. 74

D.2.4 Working Area .................................................. 74

D.2.4.1 Status bar .................................................. 74

D.3 Automatic face creation .................................................. 76

D.3.1 Face batch selection .................................................. 77

D.3.2 Output configuration .................................................. 77

D.3.3 Current information .................................................. 78

D.3.4 Input intermediate image before warping .................................................. 78

D.3.5 Output intermediate image after warping .................................................. 78

D.3.6 Progress .................................................. 78

D.4 Face recognition. Search a face .................................................. 78

D.4.1 Database selection .................................................. 79
D.4.2 Face to recognise selection .................................................. 79
D.4.3 Matched face report .............................................................. 80
D.4.4 Matched face ................................................................. 80

E Manual face mesh .......................................................... 81

F Complete list of average faces ................................................. 84

Bibliography ................................................................. 88
## List of Figures

2. Example of [http://www.myheritage.com](http://www.myheritage.com) – *Celebrity collage* .................. 4

2. Three-dimensional representation of the RGB colour Space ................................. 6
3. Examples of grey scales. Original, equal intensity, Craig’s and CIE model. .......... 7
4. Selection of a colour using HSV model in Gimp ............................................. 8
5. Mapping an image. (a) Forward mapping. (b) Inverse mapping. ....................... 9
6. Warping examples over the same image: Shear, affine, perspective and waves. ......................................................... 11
7. Single pair of lines method. Description of the variables involved .............. 12
9. Multiple pairs of lines method. Description of the variables involved .......... 14
10. Multiple pairs of lines method. Description of the variables involved ......... 15
11. Triangle Warping. Variables involved on a single triangle unit ............... 16
12. Triangle Warping. Variables involved in a single triangle ..................... 16
13. Mesh warping. Operations applied to each pixel. Original mesh (a) and desired mesh (b) .................. 17
14. Mesh Warping. Example using a cylindrical destination mesh. (a) Brooklyn Bridge (New York City, US) (b) Desired mesh with a cylindrical deformation applied (c) Result picture once the mesh warping is executed . 18
15. Set of Haar-like templates used for object detection. Edge features, line features and centre-surround features. .................................................. 20
16. Example of Haar-like templates used for object detection ...................... 21
17. Example of Haar-like templates used for object detection ...................... 21
18. Area of different regions of pixels. ......................................................... 22
19. Monalisa. Examples of simple features over a known face. ...................... 23
20. Cascade of Haar. Each classifier can reject or pass to the next classifier the image. .................................................. 23

3. Comparison between C and C++ performance .............................................. 28
4. Schema with the classes and the links between them ................................. 32
5. Screenshot. Manual average face creator .................................................. 33
6. Screenshot. Highlight of the different sections of the manual form: toolbars, menus and working area. .................................................. 34
3.5 Triangular mesh designed for the creation of a manual warped image of a face enhancing key features like eyes and mouth. 35
3.6 Screenshot. Automatic average face creator. 36
3.7 Triangular meshes designed for the automatic average face generator. Left: 34-nodes mesh. Right: 12-nodes mesh. 36
3.8 Results after the implementation of the algorithm transformation with multiple pairs of lines. 38
3.9 Interpolation of a pixel over a new mesh. 42
3.10 Results after the implementation of the algorithm triangular mesh warping. George Bush, Madonna and Tony Blair 43
3.11 Haar Cascade examples. Face location, face features location without occlusions. 45
3.12 Haar Cascade examples. Face location, face features location with glasses. 45
3.13 Haar Cascade examples. Face location inside a multiple faces in the laboratory. 46
4.1 Example of some single faces introduced in the face database to do the experiments. 48
4.2 Average images generated by the application. Left images are manual average faces and right images are automatic average faces. 49
4.3 Example of some non warped faces. 50
4.4 Results comparing the recognition rate between a single face and the automatic average face (1). 56
4.5 Results comparing the recognition rate between a single face and the automatic average face (2). 57
4.6 Results comparing the recognition rate between a single face and the automatic average face (3). 58
4.7 Results comparing the recognition rate between a single face and the automatic average face (3). 60
D.1 Screenshot. Welcome menu. 70
D.2 Layout of the manual face creator. 71
D.3 Example of a real face processing. The left area contains the desired warping image while the right one contains the help face with a red node advising us the last node selected. 75
D.4 Layout of the automatic face creator. 77
D.5 Multi-selection of face images. 77
D.6 Automatic face averaging. 79
D.7 Layout of the search face form. 79
D.8 Example of face recognition in real time. 80
# List of Tables

2.1 Description of parameters in multiple lines ........................................... 13
3.1 Description of the classes programmed for the application .................. 31
3.2 Haar-cascades files used to implement the application ......................... 45
4.1 Results of recognition rate in non-warped faces. The number of images used to generate the average changes .................................................. 50
4.2 Manual average face and automatic average face recognition rates for the first experiment ................................................................. 51
4.3 Manual average face and automatic average face recognition rates .......... 52
4.4 Database created with average faces using single included images to be recognised ................................................................. 55
4.5 Database created with average faces using single included images to be recognised ................................................................. 55
4.6 Correlation between both: manual and automatic faces ..................... 55
4.7 Single non-frontal faces Vs. automatic average faces .......................... 59
4.8 Single non-frontal faces Vs. automatic average faces .......................... 59
D.1 Description of the toolbar icons in the Manual Form ......................... 73
D.2 Landmarks/nodes referenced by colour in groups ............................... 74
D.3 List of most frequent messages displayed by the application ................. 76
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB</td>
<td>Red Green Blue</td>
</tr>
<tr>
<td>HSI</td>
<td>Hue Saturation Intensity</td>
</tr>
<tr>
<td>HSV</td>
<td>Hue Saturation Value</td>
</tr>
<tr>
<td>CCTV</td>
<td>Close Circuit TeleVision</td>
</tr>
<tr>
<td>CPU</td>
<td>Control Process Unit</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysys</td>
</tr>
<tr>
<td>LDA</td>
<td>Lineal Discriminant Analysys</td>
</tr>
<tr>
<td>EBMG</td>
<td>Elastic Bunch Graph Matching</td>
</tr>
<tr>
<td>CSA</td>
<td>Cross Segment Algorithm</td>
</tr>
<tr>
<td>GPL</td>
<td>General Public License</td>
</tr>
<tr>
<td>OCR</td>
<td>Optical Caracter Recognition</td>
</tr>
</tbody>
</table>
Dedicated to Rosalía, Jorge, Luis and Raquel who recognise my face without problem . . .
Introduction

1.1 Computer Vision

Computer face recognition is nowadays one of the most difficult tasks to be solved in terms of computer vision. During the last 10 years [1] this idea has been investigated and the results obtained have never been completely successful. In some cases, simple objects with straightforward shapes were identified correctly over carefully controlled backgrounds. In other cases like Optical Character Recognition, systems are able to detect and process texts with accuracies up to 95.38% [2].

Nevertheless artificial recognition has to deal with the real world and that is what makes this process extremely hard to be understood by machines. Faces are made up of wide varieties of lines, distances and shapes. As natural objects, human faces are formed by complex edges that make computers desperately difficult to approach them to any mathematical model. To make it more complex, human faces can change spontaneously through physical aspects like a beard or emotional expressions as a smile. Furthermore, the passage of time causes physical changes that barely can be predicted [3]. In spite of all these obstacles, face recognition appears to be a challenge not only for computers but also for human beings to teach computers how to deal with them.

1.2 Background

The earliest studies started in 50’s by psychologists. Afterwards, some years later engineers also took part in the investigations. A few face expression studies were followed
Chapter 1. Introduction. A general Overview

during the 60’s. The search on automatic machine recognition faces happened over the past 30 years by psycho physicists, neuroscientists and engineers within research on various aspects of face recognition by humans and machines [4].

At the same time, engineers have tried to formulate the problem as recognition of 3D objects based on flat images in 2D. As a result, they adopted typical techniques of pattern classification where the face characteristics were used as attributes, obtaining semi-automated systems. (See the Bledsoe experiment for details [5]). Others focused the problem with subjective marks like ear’s length or lip’s thickness complicating much more the automation of the procedure.

In the 80’s there was a surprisingly stop. Researchers were confused and all the results pointed a lack in technology support. Nonetheless during the early 90’s, the commercial attention, technology advances and security demand encourage engineers and psychologists to continue the investigations extending the paths of interest in the matter.

During the last 15 years, the main target of the researches was to create a fully-automated system eliminating the difficulties due to face location inside an input image or the extraction of facial characteristics (eyes, mouth, nose, eyebrows, etc ...). In the meantime, important advances where reached in algorithms like Eigenfaces or Fisher-faces for face recognition and patterns detection.

1.3 Applications for the solution

Applications in face recognition are widespread. Focusing on security and control aspects, it would be very useful to automate the detection of people entering in a building, track the people that are entering in a plane or directly manage the people that take a walk in the street. This new concept of people control is being promoted by several countries as United Kingdom [6] or United States of America [7] due to terrorism attacks. On the other hand, face recognition can be very useful as a biometric method of identification to have access to any system like a computer or a building.

It is important to emphasize that face recognition is one of the leading fields in security and control environments. Several governments have invested huge quantities of money and time to investigate for applicable solutions. For instance, the US government has performed multiple evaluations to determine the capabilities and limitations of face
recognition, and to encourage and direct future development. The *Face Recognition Technology Evaluation* (FERET), sponsored from 1993-1997 by the *Defense Advanced Research Projects Agency* (DARPA) was an effort to encourage the development of this technology [8]. Large firms like *Google Inc.* are also performing investigations in these aspects after the acquisition of *Never Vision Inc.* in 2006. For this reason, the face recognition improvement would be useful to improve actual applications in some fields:

- Security in airports, public places, main streets, underground . . .
- Fast identification by the police or army of potentially dangerous people
- Preventative systems to control the access to computers or personal systems
- Web based solution as an Internet images search engine
- Web based solution as an entertainment service
- Artificial intelligence enhancement with the implementation in robots (human - robot interaction)
- Supplanting of the actual identification systems such as passports or ID Cards
- Decrease the rate of *Identity Theft*
- Video games, virtual reality, training programs
- Advance video surveillance, CCTV control, building controls, etcetera . . .

Several enterprises like *L-1* offer devices installed at corporate buildings, banks, hospitals, airports and other kinds of entry points that scan faces and decide when you are welcomed [9]. Other companies like *My Heritage* (See Figure 1.2 and Figure 1.1 ) determine which celebrities you look like more or reveal if sons and daughters are more like their fathers rather than their mothers and vice-versa.

Another field of application is in automatic photo classification. That is the case of *iPhoto* by Apple 1, where the photo viewer software not only displays the personal album photos but also learns who are the people inside the pictures, permitting the user to view the photos that contains a specific person or groups of people using filters. This features, in conjunction with GPS tracking (Geolocation) permit to classify any personal image in terms of time and space, recording this data inside the image file header in a special format called EXIF 2. Nevertheless, the implantation of these systems in our life, the idea of been controlled and located anywhere at any time should be treated consciously and with care.


Figure 1.1: Example of http://www.myheritage.com – Look-alike meter

Figure 1.2: Example of http://www.myheritage.com – Celebrity collage
Chapter 2

Theoretical Aspects

2.1 Digital Image Processing

2.1.1 Introduction

An image could be defined as a bi-dimensional function $f(x,y)$ representing a light intensity, where $x$ and $y$ are the spatial coordinates being the value of $f$ in any point $(x,y)$ denominated intensity or grey level. In case $f$, $x$ and $y$ are discrete values, the image will be called Digital Image [10]. Then, any digital image could be represented as a $M \times N$ matrix where each element is called pixel.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

The image analysis transforms any image in attributes (e.g. textures, distances, shapes or borders). To do that two tasks are required: segmentation (divide an image in small regions) and description (extract values that define the object uniquely). Finally, the target of the recognition is to interpret these objects, taking their attributes and setting labels to the recognized objects, making the cognitive functions normally associated to the human vision.
2.1.2 Colour Spaces

Each one of the values of the matrix $f(x,y)$ represent a colour, generally made of two components: Luminance$^1$ and Chrominance$^2$. A colour space is a mathematical model that describes how to represent a colour just with numbers, typically as three components. During the following sections, most usual spaces will be described.

2.1.2.1 RGB Space

Most of the algorithms and applications are expected to deal with RGB colour space. That means, Red, Green and Blue components that define any colour can be separate independently in three identical dimension matrices. This model is based on a Cartesian Axis system, being the space a normalized cube where pure values red, green and blue are located in the vertices $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$ respectively:

![Figure 2.1: Three-dimensional representation of the RGB colour Space](image)

In relation with the software developed for this project, it is important to notice that all the algorithms have been developed using the RGB space because of the simplicity it offers. In the other hand RGB space has a weak point: changes in ambient light will modify rapidly the intensity of the pixels. Thus, more robust systems could be used to avoid abrupt changes due to ambient light.

---

$^1$Definition: Luminous intensity per unit area projected in a given direction

$^2$Definition: Signal used in images systems to express the colour information of the picture, independently from the accompanying luminance signal
2.1.2.2 Gray Colour Space

One of the first applications of RGB space is the conversion into Grey Space. To convert a RGB pixel into a grey scale, there is not a correct conversion, depending always on the human perception and the sensitivity response curve of the camera used. If the three intensities have the same radiance in the visible spectrum (2.1)

\[ f(x, y) = 0.333333 \times R + 0.333333 \times G + 0.333333 \times B \]  

(2.1)

then the green will appear the brightest of the three due to the luminous efficiency function peaks inside the green region of the spectrum.

Several proposals have been discussed, being the most known and used the Craig’s approach (2.2)[11] and the model defined by the Commision Internationale de l’ Eclairege (CIE) in 1931 based on human perception (2.3)[12][13]:

\[ f(x, y) = 0.300000 \times R + 0.590000 \times G + 0.110000 \times B \]  

(2.2)

\[ f(x, y) = 0.212671 \times R + 0.715160 \times G + 0.072169 \times B \]  

(2.3)

The results applying the previous methods are displayed in Figure 2.2.

Figure 2.2: Examples of grey scales. Original, equal intensity, Craig’s and CIE model.
2.1.2.3 HSV colour Space

The geometrical representation of this space is a cone (See figure 2.3). Vertical position corresponds to brightness or value, radial position means saturation and angular position is for hue.\textsuperscript{3} The main advantage of this system of representation is the abstraction of hue from the image colours, allowing intensity changes without modifying colour saturation\[13\]. However, the problem is that the conversion from RGB to HSV and vice versa is not lineal (See Appendix B). Images are by general rule stored in RGB format producing large computational cost if it is necessary to work on this space.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure2.3.png}
\caption{Selection of a colour using HSV model in Gimp}
\end{figure}

2.1.3 Mapping

There are several techniques for map an image. The classic two are reverse mapping and forward mapping. The difference between both is the way they are processed. The first one is called forward mapping because the pixels are scanned and copied from the source image to the destination image taking into account the result of the warping function. The second one is called reverse mapping because takes for every pixel in the destination a value from the source image (see Figure 2.4). Thus, for this option the inverse of the warping function must be provided. The characteristic of reverse mapping is that we ensure a value in all the pictures in the destination image while in forward mapping leaving some of the destination points unevaluated is possible. For this project application reverse mapping has been applied.

1. **Forward mapping**: Send each pixel \( f(x) \) to its corresponding location \( x' = h(x) \) in \( g(x') \)

2. **Inverse mapping**: Get each pixel \( g(x') \) from its corresponding location \( x = h^{-1}(x') \) in \( f(x) \).

\textsuperscript{3}Hue can be normalized between 0 and 100\% for some applications.
Another procedure to map an image is the Cross Segment Algorithm (CSA) [14]. This method pays more attention to the relationship between the reference and the desired images. Due to this method eliminates the searching cost, CSA became an accelerate method of inverse warping. However, sometimes is not possible to know the generalized disparity of the pixel in inverse warping and then the operation cannot be performed directly.

There have been proposed several optimized methods to avoid the problem of time and try to increase the speed [14]. All of them are classified in two ideas: increase speed according to the properties of the epipolar line or acquiring the generalized disparity of the desired image.

### 2.2 Image warping

#### 2.2.1 Introduction

Image warping is the process which transforms an image into a new deformed image following a pattern or function. This process is actually used to recover distorted pictures
projected into surfaces or to simply to create morphing effects for films or animations. Warping procedure estimate the new coordinates of an original pixel based on constraints defined beforehand.

Image warping has been used during last decades in medical imaging, computer vision, and computer graphics [15]. Main efforts have been focused on security, special effects within films and scientific investigations. These deformations have become an interesting aspect because of the apparition of new computers capable of calculate tedious and difficult operations in real time. The fast increase in powerful machines, embedded devices and real time communications has made feasible the treatment of images, its processing and of course their recognition.

There are a wide variety of algorithms that provide warped images. All of them are based on geometrical relations between the pixel and the deformation function to be followed. Depending on the objective necessarily to reach, some are more recommended than others. The critical features of the image warping algorithms are speed, accuracy and CPU resources (algorithm’s complexity). Because there are different geometric transformation techniques for digital images, for the development of the application, a study of most of them has been done and therefore it was decided which one fits better within the project requirements.

For this case, due to the quantity of images to be processed is high and the size of them might be adjusted, the best choice will be the faster assuming that the output image is truthful enough to be pre-processed later.

2.2.2 Basic transformations

The fundamental transformations involved in this processes are three:

1. Scale by a factor $s$
   
   $$
   x' = s \cdot x \\
   y' = s \cdot y
   $$

2. Rotate by an angle $\Theta$
   
   $$
   x' = x \cdot \cos(\Theta) - y \cdot \sin(\Theta) \\
   y' = x \cdot \sin(\Theta) + y \cdot \cos(\Theta)
   $$

3. Shear by a factor $h$ in one or both axis

   **X-Axis:**
   
   \[
   \begin{cases} 
   x' = x + h \cdot y \\
   y' = h \cdot y 
   \end{cases}
   
   **Y-Axis:**
   
   \[
   \begin{cases} 
   x' = h \cdot x \\
   y' = y + h \cdot x 
   \end{cases}
   \]
Each one of them is based on a different function that describes the destination axis 
\((x', y')\) for every location in the source \((x, y)\). This method can be applied vice versa if 
the lineal application is considered invertible.

These are the most relevant functions, but there are infinite equations that can be 
applied to create a deformation in the destination picture. Some samples are provided 
in the Figure 2.5.

![Warping examples over the same image: Shear, affine, perspective and waves.](image)

2.2.3 Actual algorithms for image warping

2.2.3.1 Transformation with one pair of lines

If exist a pair of lines, the first one referred to the source image and the second one 
referred to the destination image, any pixel in the source image can be transformed 
based on a mapping relation between both lines (See Figure 2.6). This algorithm has 
been extracted from [15], [16].

If \(X'\) and \(X\) are any pair of coordinates in the source picture and destination picture 
respectively and \(P'Q'\) and \(PQ\) are the lines defined by extreme points in the source and 
destination pictures respectively, it is possible to define the following equations:

\[
\begin{align*}
  u &= \frac{(X - P) - (Q - P)}{\|Q - P\|^2} \\
  v &= \frac{(X - P) \cdot \text{Perpendicular}(Q - P)}{\|Q - P\|^2}
\end{align*}
\]

(2.4)

(2.5)
\[ X' = P' + u \cdot (Q' - P) + \frac{v \cdot \text{Perpendicular}(Q' - P')}{\|Q' - P'\|^2} \]  

(2.6)

where in 2.5 and 2.6 \( \text{Perpendicular()} \) returns the vector perpendicular to, and the same length as, the input vector in the argument. The value \( u \) corresponds to the position along the line \( PQ \) while \( v \) is the distance from the line to the pixel in the source image. The range of values of \( u \) goes between 0 and 1 as long as the pixel moves from \( P \) to \( Q \) and can exceed these values outside this range. The value \( v \) corresponds to the perpendicular distance from the point to the line \( PQ \).

This option applies to each pixel of the whole image a coordinate transformation by rotation, translation and/or scale. The scale will be done only in the direction of the line while the rotation and the translation will depend on the coordinates of the pixel.

![Figure 2.6: Single pair of lines method. Description of the variables involved](image)

The method to be used with this algorithm is described as the following:

for each pixel \( X \) in the destination image
  find \( U, V \)
  find \( X' \) in the source image for that \( U, V \)
  destination image\((X) = \text{source image} \ (X')\)

Due to this process is done using reverse mapping, it ensures that all the pixels in the destination image will be defined one by one by any of the pixels in the source image. Some examples of this algorithm have been displayed in the Figure 2.7.

Since faces contains multiple lines to control all the relevant points, it is necessary to implement this method for multiple lines. It is described in the following section.
2.2.3.2 Transformation with multiple pairs of lines

The new innovation for this method consists on the management of multiple lines during the morphing process. In this case, the interaction of each pixel with all the lines will be appreciated. As [16], [17] explains, the closer the distance between the pixel and any line, the bigger would be the interaction between them. To fix this problem a weighting of the coordinate position for each line is applied following the equation:

$$\text{weight} = \left( \frac{\text{length}^p}{a + \text{distance}} \right)^b$$

(2.7)

Where length is the length of any line, distance is the distance between each pixel and that line and a, b and p are constants to determine the finest approach. The values of these constants have been studied and bounded in Table 2.2.3.2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Limits</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0-0.1</td>
<td>If a is near zero weight $\rightarrow \infty$ and points on the line will be displaced exactly in the direction of the line</td>
</tr>
<tr>
<td>b</td>
<td>0.5-2</td>
<td>Establish the relative strength between the lines decline with the distance</td>
</tr>
<tr>
<td>p</td>
<td>0-1</td>
<td>Relative strength of the lines depending on its length. The longer the line, the stronger it is if p equals one</td>
</tr>
</tbody>
</table>

Table 2.1: Description of parameters in multiple lines
The multiple line algorithm becomes as follow:

for each pixel \( X \) in the destination image

\[
\text{DSUM} = (0, 0) \\
\text{weightSum} = 0
\]

For each line \( P_iQ_i \)

calculate \( u \) and \( v \) based on \( P_iQ_i \)

calculate \( X'_i \) based on \( u \) and \( v \) and \( P_i'Q_i' \)

calculate displacement \( D_i = X'_i - X_i \) for this line

distance = shortest distance from \( X \) to \( P_iQ_i \)

\[
\text{weight} = \left( \frac{\text{length}^p}{(a + \text{distance})^b} \right)
\]

\[
\text{DSUM} = \text{DSUM} + D_i \cdot \text{weight}
\]

\[
\text{weightSum} = \text{weightSum} + \text{weight}
\]

\[
X' = X + \frac{\text{DSUM}}{\text{weightSum}}
\]

destination image(\( X \)) = source image(\( X' \))

The graphical representation of the situation helps for the understanding of the algorithm. In the Figure 2.9 the left illustration represents the coordinates in the destination while the right one represents the source.

\[\begin{array}{c}
\text{Figure 2.8: Multiple pairs of lines method. Description of the variables involved.}
\end{array}\]

where \( X' \) is the point we are sampling from the source picture and \( X \) is the destination position. This new point is computed with the weighted average of the two pixel locations \( X'_1 \) and \( X'_2 \).

The resulting graphic of this method reveals that points in the same line or very close to them are moved in the same direction as the line and pixels further from one a specific line are affected by each one of them. Some examples of this algorithm are illustrated in the next Figure 2.9.
2.2.3.3 Triangular warping

The principle of this method is to create deformed pictures based on triangular meshes. It is based on \textit{piecewise polynomial transformations}. Supposing that the original and final images have attached a mesh that define the most relevant points of the image, if both meshes are formed by simple triangles, the modification of any triangle can be studied as a change in the coordinates of the corresponding barycentre’s triangle. The following method can be applied by the thought that calculating the coordinates of the barycentre of the original mesh makes possible to obtain the new barycentre in the destination triangle. Then the alteration of the position and colour of all the pixels included inside one triangle would be warped to the final shape. So, with this method is possible to find for each triangle in original image an equivalent and copy its intensity value to the warped image through a triangle to triangle correspondence [31].

To find out where each pixel in new image comes from in old image is just necessary to follow the next steps:

1. Determine which is the triangle the pixel is inside of
2. Compute its barycentric coordinates
3. Find the equivalent points in the equivalent triangle inside the original image

The geometrical equations are, based on Figure 2.10

\[
X' = A' + \beta \cdot g + \gamma \cdot h
\]

where

\[
g = B' - A' \quad \text{and} \quad h = C' - A'
\]
So, operating
\[ X' = (1 - \beta - \gamma) \cdot A' + \beta \cdot B' + \gamma \cdot C' = \alpha \cdot A' + \beta \cdot B' + \gamma \cdot C' \]

And therefore,
\[ X' = \alpha \cdot A' + \beta \cdot B' + \gamma \cdot C' \text{ warps to } X = \alpha \cdot A + \beta \cdot B + \gamma \cdot C \]

This is true because of the next property:

\[
\begin{pmatrix}
  x \\
  y \\
  1
\end{pmatrix} =
\begin{pmatrix}
  a_x & b_x & c_x \\
  a_y & b_y & c_y \\
  1 & 1 & 1
\end{pmatrix}
\begin{pmatrix}
  \alpha \\
  \beta \\
  \gamma
\end{pmatrix}
\]

where \( \alpha + \beta + \gamma = 1 \)
Applying the equation 2.8 to each one of the pixels of the triangles of the input image, the deformation is created. This method is an step between multiple lines deformation and the following method, mesh warping.

2.2.3.4 Mesh warping

This process was pioneered by Douglas Smith for using it in the film Willow [15]. The process followed in this method is based in changes on points of different meshes instead of lines (see [15], [19] for more details). The initiative is to break up both images into small sections that are plotted onto each other deformed for the morph process. In order to apply this method, two arrays are needed. The first one contains the source coordinates of the mesh while the other one encloses the destination mesh. It is compulsory that both matrices have the same size to facilitate the one-to-one relation for the needed computations.

Two passes are needed to create the final image warped. This method is called two-pass warp. During the first pass the image is only warped vertically, but in the second pass the image is full transformed with a horizontally warp over the pre-warped image done before. Occasionally, in some parts of the picture could be recommended to do the first horizontally to maintain the picture quality [32].

\[\text{Figure 2.12: Mesh warping. Operations applied to each pixel. Original mesh (a) and desired mesh (b)}\]

If the original image is called \(I_S\) and the destination image is called \(I_D\). There are two meshes defined as \(M_S\) and \(M_D\) that contains the coordinates of both meshes (See Figure 2.12). It is important to notice that both meshes are constrained to be topologically equivalent [15]. That means that no folding, discontinuities or self-intersection between nodes in both networks are allowed. During this process intersection some intermediate images are created.
The algorithm to be used has to be the following one:

\[
\text{For each frame } f \text{ do} \\
\begin{align*}
\text{linearly interpolate mesh } M & \text{ between } M_S \text{ and } M_D \\
\text{warp } I_S & \text{ to } I_1 \text{ using meshes } M_S \text{ and } M \\
\text{warp } I_S & \text{ to } I_1 \text{ using meshes } M_S \text{ and } M \\
\text{linearly interpolate image } I_F & \text{ between } I_1 \text{ and } I_2
\end{align*}
\]

Interpolation of the meshes

There are several methods to implement the interpolation between two collections of coordinates. But, the most used method is the bi-cubic spline interpolation. More precisely, the usual algorithm is Catmull – Rom spline interpolation. A deeper explanation is attached at the end of the text for further analysis. (See Appendix A)

An example of the deformation created by using this method is displayed in Figure 2.13 based on the example of [21]. In this case the deformation has been created following a cylindrical pattern as a destination mesh.

![Figure 2.13: Mesh Warping. Example using a cylindrical destination mesh.](image)

(a) Brooklyn Bridge (New York City, US) (b) Desired mesh with a cylindrical deformation applied (c) Result picture once the mesh warping is executed

To obtain a smooth and continuous batch of pictures in the transition between both pictures is recommended to apply a cross-dissolve filter\(^4\). This action will provide a natural behaviour in the path between the original image and the final result.

\(^4\)That means, a simple linear transition between images
2.3 Objects detection

2.3.1 The problem. How to recognise objects?

Image analysis for object detection is concerned with the extraction of measurements, data or information from an image by automatic or semi-automatic methods. By definition, the systems to be analysed are not limited to the classification of scene regions to a fixed number of categories, but they rather are designed to provide a description of complex scenes whose variety may be enormously large and ill-defined in terms of a priori expectation [22].

The objective in this study will be to find as fast and accurate as possible facial patterns defined beforehand: both eyes, nose and finally the mouth for each face. It is important to take into account that the application developed starts from the idea that any input image contains faces but it could be changed simply to any other static object, converting it into a portable application for future projects.

The final usage for objects detection would be face deformation. Therefore, depending on the number of objects detected inside the face, the deformation will vary, being better as long as this quantity is increased. One of the requirements for this project is a pseudo real-time analysis. That means that the results must be shown just-in-time fast and ready to be used afterwards by other systems. That is the reason why the method selected was to use Rapid Object Detection through a Boosted Cascade of Simple Features, described in the next section.

2.3.2 The solution. Haar Cascades

Due to human faces are complex objects it is complicated to find features or heuristics that could confront the huge variety of instances of the object class (e.g. faces, eyes, mouths . . . ) that may rotate in any direction, captured in different light conditions or the simple apparition of glasses, beards or wrinkles. For such objects, statistical models (here called classifiers) may be trained and used to detect the desired targets.

To do that, statistical models will be taught using multiple instances of the object to be recognized (these instances are called positive) and also multiple samples of negative instances where the object does not appear. The collection of all these samples, positive and negative, form a training set. During the training process, face features will be
extracted from the training set and unique features of each image will be used to classify the object. It is important to remark that using this method, if the cascade does not detect an existing object it is possible to add the sample to the classifier training set and correct it for the next time.

The statistical approach used in this project has been defined using the OpenCV libraries (mentioned and explained later on during the following sections) based directly on the Viola & Jones publication [23]. This option applies simple Haar-Like features and a cascade of boosted tree classifiers as a statistical model. The classifier must be trained on images of the same size and detection is done using a window of that size moved along the whole picture. For each step, the algorithm check if the region looks like the desired object or not. Furthermore, to include possible sizes of the images to be detected, the classifier has the ability to scale the patterns. To make this method works, it is necessary just a batch of Haar-like features and a large set of very simple classifiers to classify the image region as the desired object or as a non desired object.

Each feature is determined by the shape of the feature, its position relative to the search window origin and the scale factor applied on the feature. In total, 14 templates shown in Figure 2.14 are used for the project module.

![Haar-like templates](image)

**Figure 2.14:** Set of Haar-like templates used for object detection. Edge features, line features and centre-surround features.

As in the previous Figure, each feature is designed using two or three black or white rectangles horizontal, vertical or rotated by 45°. To compute the Haar feature’s value, just a weighted summation of two components is needed: the pixel sum over the whole area of the feature and the sum over the black rectangle. Once this simple calculation is done, the weights of both components are of opposite signs and they are normalized dividing the summation over the black rectangle by the total area. As an example, for
the second feature:

\[ \begin{array}{c}
\text{Figure 2.15: Example of Haar-like templates used for object detection.}
\end{array} \]

\[ 2 \times weight_{\text{black}} = -4 \times weight_{\text{whole}} \tag{2.9} \]

or for the thirteenth feature:

\[ \begin{array}{c}
\text{Figure 2.16: Example of Haar-like templates used for object detection.}
\end{array} \]

\[ weight_{\text{black}} = -9 \times weight_{\text{whole}} \tag{2.10} \]

Now, instead of computing directly the pixel sums over multiple rectangles and make the process of detection incredibly slow, [23] defined a way to make the summations faster: Integral Image.

\[ ii(x, y) = \sum_{x' < x, y' < y} i(x', y') \tag{2.11} \]

where \( ii(x, y) \) is the integral image and \( i(x, y) \) is the original image. The summation of pixels over a single window \( r = \{(x, y), x_0 \leq x \leq x_0 + w, y_0 \leq y \leq y_0 + h\} \) can be done using the surrounding areas as showed in Figure 2.17.

where the sum of the pixels of the rectangle \( D \) can be computed with four array references as is demonstrated in ?? . The integral image at 1 corresponds to the summation of all the pixels included in rectangle \( A \). Just as the previous example, the value at 2 is \( A + B \), the value at 3 is \( A + C \) and finally the value at 4 is \( A + B + C + D \). Therefore, the sum within \( D \) can be done as \( 4 + 1 - (2 + 3) \). That means that the pixel sum over a rectangle can be done regardless of the size, just taking into account the corners of the rectangle.

\[ RecSum(r) = \underbrace{ii(x_0 + w, y_0 + h)}_4 + \underbrace{ii(x_0, y_0)}_1 - \underbrace{ii(x_0 + w, y_0)}_2 - \underbrace{ii(x_0, y_0 + h)}_3 \]
If a decision tree classifier is created taking into account each one of the feature value computed over each area of the image, as in 2.3.2 for two terminal nodes or 2.3.2 for three, where each $f_i$ will give $+1$ if the obtained value is inside a threshold pre-defined and $-1$ otherwise.

$$f_i = \begin{cases} +1, & x_i \geq t_i \\ -1, & x_i \leq t_i \end{cases}$$

(2.12)

$$f_i = \begin{cases} +1, & t_{i,0} \leq x_i < t_{i,1} \\ -1, & \text{else} \end{cases}$$

(2.13)

where $x_i = w_{i,black} \cdot \text{RecSum}(r_{i,black}) + w_{i,whole} \cdot \text{RecSum}(r_{i,whole})$

It is important to notice that each classifier is not ready to detect an object by itself. It notices simple feature in the image (like a border or a point). For instance, the eye region is often darker than the cheeks, or the iris is darker than the rest of the eye. (supposing a correct size and orientation of the feature as in Figure 2.18)

After the creation of these classifiers (called weak classifiers) a list of complex robust classifiers is built out with the union of all the weak classifiers iteratively as a weighted
sum of weak classifiers, being each one increasingly more complex. Afterwards, a cascade is created where first positions are for simple classifiers and final positions for the most complex. As far as the window is scanned by each classifier, it can be rejected or sent to the next $F_i$ as in Figure 2.19 is explained.

![Figure 2.18: Monalisa. Examples of simple features over a known face.](image)

![Figure 2.19: Cascade of Haar. Each classifier can reject or pass to the next classifier the image.](image)

### 2.4 Face recognition

#### 2.4.1 The location problem

Most of the techniques used for facial recognition assume the availability of *ideal images* (good light location and intensity, uniform background, smooth variations in the object’s shape, sufficient resolution, uncompressed, etcetera ...). However, real scenarios are completely different: poor defined shapes, diffuse or moved objects and complex
backgrounds that force the use of previous processing (pre-processing) to acquire a better image ready to extract face regions from the environment.

Probably this is one of the most exciting and difficult tasks in face recognition: to extract face coordinates with low probability of failure. Until the middle 90’s the face was searched under highly controlled backgrounds (one shade). These methods included the use of neuronal networks, skin texture or deform meshes based on facial characteristics.

### 2.4.1.1 Face location through global representation

Among all the actual techniques used to detect faces the one that has been mostly emphasized is the rules codification that describes the characteristics of the face and its relations (relative distances, positions or even areas described by a group of points). This method presents two particularly difficulties: It is necessary to know beforehand that there is a face in the scene and secondly the generation of too much false positives in complex backgrounds [24].

Other alternatives have been studied to improve accuracy and quality rates like global techniques based on localization of faces treating them as a pattern location problem assuming that the face is the pattern. When the templates are used, a pattern is searched defined manually in the input image. That is, face and non face categories are determined by an expert. Methods based on appearance learn the templates from the training images. In general, these methods use statistical analysis and automatic learning to find the relevant features of the categories face and non face.

The disadvantage of algorithms based on templates is the difficulty for generalize the algorithm to be used in different situations of images (illumination, spatial hiding or position). To fix that, the expert have to define a high number of templates to cover any possible situation and therefore represent any case. Both methods develop an exhaustive search of patterns for any position or scale extracting the output region and classifying the result with the classical methods.

The last group of methods provides the face location based on colour distribution techniques. Because faces maintain a narrow colour band inside the colour space and most information is concentrated in chrominance and not in luminance, this component can
be rejected using segmentation models: $RGB, HSV, YCrCb, YIQ, CIE, XYZ$ [25]

### 2.4.1.2 Face location through *local* representation

These methods try to locate faces detecting facial characteristics like eyes, eyebrows, mouth, face contour, etcetera... and then they are combined with other proceedings that identify the relations and verify the existence of the face. These procedures are based on the idea that humans are able to recognize faces in different positions or luminance situations because there should be a group of distinctions or properties independent to those variables.

### 2.4.2 Algorithms

#### 2.4.2.1 Eigenfaces. Principal Components Analysis

The eigenfaces method is probably the simplest and efficient method used for human face recognition. Contrary to others algorithms based on the identification of gestures and classification of distances between them, this method evaluates the image as a whole. Thanks to this consideration, is feasible to use not only in prepared environments with pre-defined light conditions but also outside, in the real world. Basically, this method is supported on a simple concept: reduce useless information as much as possible. When a picture is studied, even if it is small, there is a lot of information expressed inside. Therefore, depending on the final purpose, the focus on some parameters will be highlighted over the rest.

When a general image is being analysed, is expected that most of the area of the image will not represent a face. That is why is necessary a method to detect and extract human faces from the rest of the background as a fast and accurate procedure. The way to do that is creating a base of faces and trying to represent any analysed image as a combination of them. It is easy to compare this technique to the colours representation. The base of colours is formed by primary red, green and blue. Thus, it is possible to represent any colour as a partial addition of red, green and blue. For instance, orange will be formed as a combination of the maximum value of red, half value of green and nothing of blue.
Referring to eigenfaces, the problem becomes to find the precise collection of base faces that represents the best a specific face. Taking advantage of the previous comparison, the issue is how much quantity of red, blue and green paint does the painter need to reproduce the colour he is already watching. It is important to notice that the precise selection of the base faces will provide better results in the recognition step. The creation and development of the mentioned face base is explained in detail in several publications [26], [27], [28], taking into account that some information reduction has to be applied in order to diminish the complexity of the original problem up to 1/1000.

Each face we want to introduce in the base to be classified can be projected into a face space and then analysed as a vector. After that, any existent distance method like Euclidean distance, k-nearest-neighbour, Mahalanobis distance or neural network can be used to classify the input face. The problem is solved after this step, inducing next points to be effortless.

The eigenfaces method can be divided into three differentiated sections:

- Generation of eigenfaces.
- Projection of trained data into face-space to be used afterwards with a classification method.
- Evaluation of a projected test image and compare it with training data.

2.4.2.2 Linear Discriminant Analysis

Linear Discriminant Analysis is a statistical approach for classifying samples of unknown classes based on training samples with known classes. The purpose of this technique is to maximize between-class (e.g., across users) variance and minimize within-class (e.g., within user) variance. If this is moved to high dimensional face data, this method faces the small sample size problem that arises where there are a small number of available training samples compared to the dimensionality of the sample space [29].

2.4.2.3 Elastic Bunch Graph Matching

Elastic Bunch Graph Matching is based on the idea that human face images have many non-linear characteristics that are not supported by the linear analysis methods such as
variations in pose (standing straight versus leaning over), face expression (smile versus sad) or illumination (outdoor lighting versus indoor fluorescents).

A Gabor wavelet transform (see [30] for more details) generates a dynamic link architecture that projects the face onto an elastic mesh. The Gabor jet is a node on the elastic grid (represented by circles) which determines the image behaviour around a given pixel. It represents the result of a convolution of the face image with a Gabor filter, which is applied to detect shapes and to extract relevant features by means of image processing. The recognition process relies on the similarity of the Gabor filter response at each Gabor node. This biologically-based methodology that processes Gabor filters is a procedure executed in the visual cortex of higher mammals. The main disadvantage of this method lies on the necessity of use an accurate landmark localization, that often can be achieved mixing PCA and LDA methods.
Chapter 3

Development of the application

3.1 Language, libraries and environment

3.1.1 Programming Language: C++

For the complete development of this project, C++ language has been chosen. One of the main requirements was to increase speed as much as possible, and C++ is ideal for high performance applications because it works not only with high level functions but also with low-level machine functions. C++ is a middle level language object oriented (OO) with some features like classes management, class heritage and overloaded functions. In the Figure ?? a comparison between C and C++ is performed showing that for small-medium applications C++ is recommended, being better C for big applications or high resources necessities [31]. For this project gcc v.4.3.2 i486-linux-gnu has been used.

![Figure 3.1: Comparison between C and C++ performance.](image-url)
Chapter 3. Development of the application

3.1.2 Operative System

Other of the advantages of C/C++ is the portability that it offers. Because this programming language is multi-platform, is possible to compile it in any Unix environment (Linux, BSD, Solaris, Mac, etc...) or Windows environment. For this study, Ubuntu 8.10 - the Intrepid Ibex (released in October 2008) has been used.

3.1.3 Graphic User Interface Libraries: Qt4

To make the usability easier towards the final user, a graphic interface has been designed in Qt4. The Qt4 libraries were created originally by TrollTech. In June 2008 Nokia acquired Trolltech to enable the acceleration of their cross platform software strategy cross-platform for mobile devices and desktop applications, and to develop its Internet services business. On September 2008 Nokia renamed Trolltech to Qt Software. In fact, Qt libraries are a very good solution to develop standard interfaces. Qt is a cross-platform application framework. Using Qt, it is possible to develop applications and user platform interfaces once, and deploy them across many desktop and embedded operating systems without rewriting the source code. Some of the systems that base part of the code in Qt are consuming electronics (mobile handsets, GPS navigators), manufacturing (part of the Siemens manufacturing assembly line), medical (digital radiology), entertainment (film production) or transportation (mobility systems). The original idea of this project was to implement the algorithms in a standard laptop/PC but it is interesting to mention that could be a feasible option to re-pack the application and install it in embedded systems, PDAs or hand-holds to make the idea portable.

Qt platform is governed by the GNU General Public License version 2 and version 3 so can be used freely as long as it is not used with commercial purposes. There are two possible licenses for Qt4, the open source and commercial. For the target of this project, open source version has been selected.

3.1.4 Computer Vision Libraries: OpenCV

After the study of the Computer Vision basic algorithms and digital image processing fundamentals, one of the first issues to deal with is the complexity of the functions to be used to create a useful application. Because Computer Vision is on every robotic engineer’s lips, several libraries of public distribution have been released amongst which

\footnote{More info: \url{http://www.qtsoftware.com}}
two of them stand out. One of the objectives of the project was to investigate which of them are available and select the most adequate. As it is mentioned before, libraries must be written in C++ in order to be included in the main program.

3.1.4.1 CImg $^2$: A Templated Image Processing Library in C++

The CImg Library is a free C++ tool-kit providing simple classes and functions to load, save process and display images in C++ code. It consists only of a single header file CImg.h that must be included in the program source. It contains useful image processing algorithms for loading/saving, resizing/rotating, filtering, object drawing (text, lines, faces, ellipses, ...). Images are instanced by a class able to represent images up to 4-dimension wide (from 1-D scalar signals to 3-D volumes of vector-valued pixels), with template pixel types. One of the big advantages of this libraries is that it depends on a minimal number of libraries: it is easy to compile it with only standard C libraries. There is no need for exotic libraries and complex dependencies.

3.1.4.2 OpenCV $^3$: Open Source Computer Vision Library

The most popular free library mainly aimed at real time Computer Vision. It includes a complete list of resources and manuals to interact fully with the libraries from the main application, enhancing the possibilities and the expectations of the project. This point and its easy merger with the graphical libraries Qt4 mentioned before made OpenCV the best choice to implement the application for this project (the extension of this library exceeds the limits of this project). Some example areas covered by OpenCV would be Human – Computer Interaction (HCI), object identification, segmentation and recognition, face recognition, gesture recognition, motion tracking, motion understanding, Structure From Motion (SFM) and mobile robotics.

3.2 Schema of the application

Because the entire software has been developed in C++, is straightforward to describe the classes involved in the application and their distribution but the relationships and connections are in some cases complex to be represented.

$^2$More info at: http://cimg.sourceforge.net/
$^3$more info at: http://sourceforge.net/projects/opencvlibrary/
3.2.1 Schema of the application

To describe it later on, a list with the classes with a briefly explanation about the purpose of each class is presented in Table 3.2.1

<table>
<thead>
<tr>
<th>Name of the Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AutomaticWidget.cpp</td>
<td>Interface for the automatic average generator window.</td>
</tr>
<tr>
<td>CameraCapture.cpp</td>
<td>Capture frames and look for faces inside it.</td>
</tr>
<tr>
<td>Detector.cpp</td>
<td>HaarCascades engine.</td>
</tr>
<tr>
<td>EigenFaces.cpp</td>
<td>EigenFaces engine.</td>
</tr>
<tr>
<td>GraphWidget.cpp</td>
<td>Manual average engine. Nodes, edges and warped images manager.</td>
</tr>
<tr>
<td>Main.cpp</td>
<td>Program launcher.</td>
</tr>
<tr>
<td>MainWindow.cpp</td>
<td>Main application interface, tool menus and status information handler.</td>
</tr>
<tr>
<td>Mixer.cpp</td>
<td>Manual average merger engine.</td>
</tr>
<tr>
<td>MixerAut.cpp</td>
<td>Automatic average merger engine.</td>
</tr>
<tr>
<td>Node.cpp</td>
<td>Representation of each node included in GraphWidget designed for manual mode.</td>
</tr>
<tr>
<td>NodeAut.cpp</td>
<td>Representation of each node included in GraphWidget designed for automatic mode.</td>
</tr>
<tr>
<td>SearchWidget.cpp</td>
<td>Interface for the search face window.</td>
</tr>
<tr>
<td>StoneFaceWidget.cpp</td>
<td>Automatic average engine. Manager of all the automatic features.</td>
</tr>
<tr>
<td>WelcomeWidget.cpp</td>
<td>Interface for the initial menu.</td>
</tr>
</tbody>
</table>

Table 3.1: Description of the classes programmed for the application

In Figure 3.2 there is a representation a global view of the life cycle of the application. In short, there upper-level classes are responsible for the user interface (creation of menus, buttons, list-views, user-dialogues or display alerts) while the bottom-level classes manage the mathematical functions (open images, read them, warp them, compute meshes, search for faces, merge files, ...). A complete schema with the methods and classes connections can be found in Appendix C

3.2.2 Interface objects and custom objects

The connection between the interface objects and the internal classes is done through slots and signals. The function that relate slots and signals is called connect and it is defined as:

```cpp
#include <QObject>

bool QObject::connect ( const QObject * sender, const char * signal, const QObject * receiver, const char * method, Qt::ConnectionType type = Qt::AutoConnection )
```

```cpp
QObject::connect ( this, &MainWindow::search_clicked, &SearchWidget::search_window )
```
Figure 3.2: Schema with the classes and the links between them.

An example extracted from [32] connects a label with a scrollbar. In this case, when the user interacts with the scrollBar object, it will emit a signal called valueChanged(int) that has an argument int that contains the actual value of the scrollBar. The label will display whatever is set in the only argument int of its function setNum(int). Because this function is linked to the valueChanged(int), the label will represent directly the value of the scrollBar.

QLabel *label = new QLabel;
QScrollBar *scrollBar = new QScrollBar;
QObject::connect( scrollBar, SIGNAL(valueChanged(int)),
                   label, SLOT(setNum(int)));

With this method, it is possible to capture all the events or actions the user perform
in the application (clicks, drag drop, keys pressed, etcetera...) and treat them independently.

3.3 Description of the application

The usage of the application is very simple. Once the user execute it, a menu will appear offering three options in form of big icons. Depending on the selection, three different windows could appear.

3.3.1 Manual average

3.3.1.1 Interface

The first option open a full screen window with menu bars, toolbars and a status bar ready to open a batch of images and let the user to decide the node’s position in order to generate a face triangular mesh and afterwards warp the image.

---

*Figure 3.3: Screenshot. Manual average face creator.*

---

\(^4\) All the resources of the application have been downloaded from http://www.crystalxp.net (last visited April 10, 2009) and are free to use and distribute by the GPL
Figure 3.4: Screenshot. Highlight of the different sections of the manual form: toolbars, menus and working area.

It is possible to customize the location and size of the menus in order to make a friendly user environment. Furthermore, is possible to hide the right head with the help nodes to save space and also to show/hide the mesh to make the image more clear (for full information see User’s manual in Appendix D). The left toolbar is designed to let the user to move the nodes grouped by face features (eyebrows, eyes, mouth, nose, etcetera...) or individually. Furthermore, the mesh has been coloured in order to distinguish better the different groups of nodes.

3.3.1.2 Manual mesh

The mesh is made up of 85 nodes distributed along the whole face forming a triangular mesh. It was designed in order to enhance some important face features like eyes or mouth\(^5\) (for full information see Appendix E). That is the reason why there are more

\(^5\)Originally designed by Dr. Rob Jeckins. Psychology Department. University of Glasgow (2009)
nodes around this areas. The mesh can be modified easily creating new results highlighting other points/features. The level of detail of the deformation is directly linked with the number of nodes of the mesh to be warped. As long as we increase the number of nodes, the deformation will be more accurate and therefore more complex to perform for the machine. In the following Figure 3.5 the mesh is displayed. The results obtained of applying this mesh can be read later on Chapter 4.

![Triangular mesh designed for the creation of a manual warped image of a face enhancing key features like eyes and mouth.](image)

**Figure 3.5**: Triangular mesh designed for the creation of a manual warped image of a face enhancing key features like eyes and mouth.

### 3.3.2 Automatic average

#### 3.3.2.1 Interface

The second option open a window with three main parts (see Figure D.6). The top one ask the user to introduce the batch of images to analyse. Afterwards, some option about the output image can be defined like the colour space (colour or grey scale) or the final size. Finally pressing *Create automatic average...* button, the application will start to open each one of the images and will use Haar Cascades to detect the required face features. It is important to notice that if any of the 5 things searched for any face (face, left eye, right eye, mouth, nose) is not found, this face will be automatically rejected. The reason for that is because the application cannot create a mesh if it unknowns the
position of any of the nodes involved (consequences of this decision will be discussed later on Chapter 4).

Figure 3.6: Screenshot. Automatic average face creator.

3.3.2.2 Automatic mesh

Contrary to the previous mesh, the automatic mesh was designed to be less complex. There were two different versions of this mesh. The first one was developed with 34 nodes and the second one was just 12 nodes. After some experiments it was decided that the results did not vary so much between both meshes due to points like eyebrows or chin were poorly estimated in the 34-nodes mesh. As a consequence, the 12-nodes mesh has been implemented for the experiments. Both meshes can be seen in Figure 3.7

Figure 3.7: Triangular meshes designed for the automatic average face generator. Left: 34-nodes mesh. Right: 12-nodes mesh.
3.4 Warp algorithm

One of the top features of the project is the warp algorithm implemented. After the study of the most popular ways to deform images in Chapter 2, two of them were implemented. Afterwards, they were compared to decide who was the optimal decision.

3.4.1 Implementation of transformation with multiple pairs of lines

The algorithm was implemented following the steps described in [16] also explained previously. As discussed in [17], the main advantage of this method is that gives the user a high level of control over the process but it is necessary to select corresponding feature lines, being more oriented to morphing rather than warping. Everything that is specified is moved exactly as the animator wants it to move, and everything else is blended smoothly based on those positions. There are two main problems to highlight: [17].

- **Speed** All segments have to be referenced for every pixel. The number of operations is proportional to the number of lines times the number of pixels, making the procedure consume lots of computer resources.

- **Quality** Not always the result is as expected. The apparition of ghosts [15] or undesirable artefacts can be produced.

The results of the implementation of this algorithm are presented on the next Figure 3.8. It is easy to notice that the deformations are inaccurate in some cases and the textures are poorly warped as a consequence of the previous problems (deformations are not completely smooth because the definition of the lines is complex).

3.4.2 Implementation of triangular mesh warping

After the implementation of the previous method, a different solution was needed and that is why this algorithm was written. The computation cost is close to mesh warping and the complexity is lower. It is not necessary to use any mesh interpolation like Cat-Mull Algorithm (full code in Appendix A) saving large computational cost.

The steps followed to warp an image using a triangular mesh are very simple:

1. Obtain which is the correspondent triangle for each pixel in the warped mesh.
Figure 3.8: Results after the implementation of the algorithm transformation with multiple pairs of lines.

```cpp
for (int j = 0; j < imf1->height(); j++)
{
    for (int i = 0; i < imf1->width(); i++)
    {
        // For each triangle in the image...
        for (int k = 0; k < nTri; k++)
        {
            inside = insideTriangle(
                aNodes2[ aTri[k][0]][0], aNodes2[ aTri[k][0]][1],
                aNodes2[ aTri[k][1]][0], aNodes2[ aTri[k][1]][1],
                aNodes2[ aTri[k][2]][0], aNodes2[ aTri[k][2]][1], i, j);

            if (inside == true)
            {
                mTriangles[i][j] = k;
                flag = true;
            }
        }
    }
}
```
if (flag == false) {
    printf("Point out of matrix?? X(%d,%d)\n",i, j);
    return(-1);
}

flag = false;
}

2. Creation of the matrix displayed in the equation 2.8 for each pixel.

for (int j = 0; j < imf1->height()-1; j++) {
    for (int i = 0; i < imf1->width()-1; i++) {

        //Point A of the triangle in which the point (i,j) is located
        A[0][0] = aNodes2[aTri[mTriangles[i][j]][0]][0];
        A[1][0] = aNodes2[aTri[mTriangles[i][j]][0]][1];
        A[2][0] = 1;

        //Point B of the triangle in which the point (i,j) is located
        A[0][1] = aNodes2[aTri[mTriangles[i][j]][1]][0];
        A[1][1] = aNodes2[aTri[mTriangles[i][j]][1]][1];
        A[2][1] = 1;

        //Point C of the triangle in which the point (i,j) is located
        A[0][2] = aNodes2[aTri[mTriangles[i][j]][2]][0];
        A[1][2] = aNodes2[aTri[mTriangles[i][j]][2]][1];
        A[2][2] = 1;

    }

    3. Compute the inverse to obtain the values of α, β, γ

    if (detInv == 0.0 ) {
        printf("Determinant null! %d %d %f\n",i,j, A[1][1]);
    }
break;
}

detInv = (double)(1.0 / detInv);


4. Calculate $X = \alpha \cdot A + \beta \cdot B + \gamma \cdot C$ for each pixel.

alpha = in[0][0]*i + in[0][1]*j + in[0][2];
beta = in[1][0]*i + in[1][1]*j + in[1][2];
gamma = in[2][0]*i + in[2][1]*j + in[2][2];

5. Interpolate the pixel to know where to locate the pixel in the final mesh.

//Point Ap of the triangle (original triangle)
Apx = aNodes [aTri[ mTriangles[i][j] ][0]][0];
Apy = aNodes [aTri[ mTriangles[i][j] ][0]][1];

//Point Bp of the triangle (original triangle)
Bpx = aNodes [aTri[ mTriangles[i][j] ][1]][0];
Bpy = aNodes [aTri[ mTriangles[i][j] ][1]][1];

//Point Cp of the triangle (original triangle)
Cpx = aNodes [aTri[ mTriangles[i][j] ][2]][0];
Cpy = aNodes [aTri[ mTriangles[i][j] ][2]][1];

Xp = alpha * Apx + beta * Bpx + gamma * Cpx;
Yp = alpha * Apy + beta * Bpy + gamma * Cpy;
// Interpolation of the pixel

// by Nearest Neighbour Method .... or ...
// valueOrig = imf1->pixel(Xp,Yp);

//... my own interpolation :)

dxLeft = Xp - int(Xp);
dxRight = 1 -dxLeft;
dyUp = Yp - int(Yp);
dyDown = 1 - dyUp;

// Get the interpolated colours:

RedInterpolated =
 qRed(imf1->pixel((int)Xp ,(int)Yp )) * (dxRight * dyDown) +
 qRed(imf1->pixel((int)Xp ,(int)Yp+1)) * (dxRight * dyUp ) +
 qRed(imf1->pixel((int)Xp+1,(int)Yp )) * (dxLeft * dyDown ) +
 qRed(imf1->pixel((int)Xp+1,(int)Yp+1)) * (dxLeft * dyUp  );

GreenInterpolated =
 qGreen(imf1->pixel((int)Xp ,(int)Yp )) * (dxRight * dyDown) +
 qGreen(imf1->pixel((int)Xp ,(int)Yp+1)) * (dxRight * dyUp ) +
 qGreen(imf1->pixel((int)Xp+1,(int)Yp )) * (dxLeft * dyDown ) +
 qGreen(imf1->pixel((int)Xp+1,(int)Yp+1)) * (dxLeft * dyUp  );

BlueInterpolated =
 qBlue(imf1->pixel((int)Xp ,(int)Yp )) * (dxRight * dyDown) +
 qBlue(imf1->pixel((int)Xp ,(int)Yp+1)) * (dxRight * dyUp ) +
 qBlue(imf1->pixel((int)Xp+1,(int)Yp )) * (dxLeft * dyDown ) +
 qBlue(imf1->pixel((int)Xp+1,(int)Yp+1)) * (dxLeft * dyUp  );

// Create a color with the values obtained and fill the pixel
QColor color(RedInterpolated, GreenInterpolated, BlueInterpolated);

    // Set the new value in destination
    imf2->setPixel(i,j,color.rgb());

where the interpolation implemented is very simple based on the four areas created when a destination value does not match with any of the values of the final mesh. A simple weigh is done between these areas instead of using the nearest neighbour interpolation. The Figure 3.9 shows the names of the variables involved to facilitate the understanding of the method applied.

![Figure 3.9: Interpolation of a pixel over a new mesh.](image)

Some examples of the results given by this algorithm can be shown in Figure 3.10 where is the deformations are smoother and clear. The quality of the deformation for this method depends directly on the level of detail and the size of the triangles. As long as the quantity of triangles is increased, the deformation of each triangle is lower and therefore the total transformation is less "sharp". Other of the advantages for this method is that the mesh is straightforward to deal with (it is much easier to imagine a mesh than a set of lines representing outlines of the face ) allowing the user to change it comfortably, intuitively and just modifying one file with the coordinates. Therefore, for the final version of the project, the triangular mesh warping method was selected.

### 3.5 Looking for face features

#### 3.5.1 Cascade files management

Exactly as it was discussed previously, the pattern recognition was performed using OpenCV libraries adapted to the project environment. The classes in charge of this
task are GraphWidget (manual performance), Detector (Haar cascades management) and StoneFaceWidget (automatic performance). To do that, it is essential to build a cascade’s file with a collection of positive and negative samples of frontal faces, left eyes, right eyes, mouths and noses. Because it takes a lot of time to prepare this files, they have been downloaded from a specialized open source website \(^6\) were the files are already prepared and checked to be the best updated approach[34]. A list of the files included in the project can be checked in the next Table 3.5.1

3.5.2 Implementation of the Haar Cascades algorithm

The complete code written to deal with Haar Cascades (extracted partially from [35], [36]) is very extensive to copy it directly here but the main points to follow are:

1. Create an image using the `cvCreateImage` function.

\(^6\)Website: \url{http://alereimondo.no-ip.org/OpenCV/34} last visited April 10, 2009
2. Empty the image.

3. Create a Detector object passing as parameter the image created already.

4. Create the cascade taking into account the feature to be recognized (mouth, nose, eyes ...) using the expression `cascade = (CvHaarClassifierCascade*)cvLoad(cascadePathName, 0, 0, 0);`

5. Create a sequence of recognized images using the expression:
   
   `CvSeq* faces = cvHaarDetectObjects(image, cascade, storage,1.1, 5, CV_HAAR_DO_CANNY_PRUNING);`

   where the function `cvHaarDetectObjects` looks for rectangular regions in the created image that are likely to contain objects the cascade has been trained for and returns those regions as a sequence of rectangles. The function scans the image several times at different scales\(^7\).

6. Create a new rectangle to pick up the coordinates using the expression `CvRect* r = (CvRect*)cvGetSeqElem(faces, i);` for each region found in the image.

7. Extract the image embedded in this rectangle using the previous data.

8. Set the mesh points applying the coordinates given.

Some examples of the Haar Cascades results are displayed in the next figures. In Figure 3.11 an example of a clear frontal face is detected including all the features: eyes, nose and mouth. In Figure 3.12 an example of a frontal face is detected. This time the subject wears glasses but both eyes have been found. In Figure 3.13 an example of multiple people face detection is displayed.

### 3.6 Face recognition

#### 3.6.1 Eigenfaces using OpenCV

The last part of the project consist on the development of a simple face recognition module to demonstrate if the results are or not valid. The OpenCV libraries include a module dedicated to PCA, that means, it is ready for eigenfaces applications. The code used for this part was extracted based on a previous project\(^8\) with several modifications in order to be compatible with the actual project application.

---

\(^7\) More info can be found in [http://www710.univ-lyon1.fr/~bouakaz/OpenCV-0.9.5/docs/ref/OpenCVRef_Experimental.htm](http://www710.univ-lyon1.fr/~bouakaz/OpenCV-0.9.5/docs/ref/OpenCVRef_Experimental.htm)

\(^8\) Check [http://www.codeproject.com/KB/cpp/TrackEye.aspx](http://www.codeproject.com/KB/cpp/TrackEye.aspx)
Description | Author(s) | Vers. | Filename
--- | --- | --- | ---
Frontal Face stump 24x24 | Rainer Lienhart | 1.0 | frontalface_alt_tree.xml
Right Eye 18x12 | Modesto C. Santana | 1.0 | ojoD.xml REye18x12.zip
Left Eye 18x12 | Modesto C. Santana | 1.0 | ojoL.xml LEye18x12.zip
Frontal eyes 35x16 | Yusuf Bediz | 1.0 | frontalEyes35x16XML.zip
Mouth 25x15 | Modesto C. Santana | 1.0 | Mouth.xml
Nose 25x15 | Modesto C. Santana | 1.0 | Nose.xml Nose25x15.zip

Table 3.2: Haar-cascades files used to implement the application

Figure 3.11: Haar Cascade examples. Face location, face features location without occlusions.

Figure 3.12: Haar Cascade examples. Face location, face features location with glasses.
Chapter 3. Development of the application

Figure 3.13: Haar Cascade examples. Face location inside a multiple faces in the laboratory.

1. Creation of a Database with all the eigenfaces.
2. Application of PCA to the Database to reduce the information
3. Load training information of the Database created.
4. Project the test image into the PCA subspace
5. Find the closest distance between the subspace and the projected face
Chapter 4

Results and Discussion

In this chapter, some studies and experiments will be discussed to understand if the average of a face increase the accuracy measurement of a face recognition software. Furthermore, several comparisons have been run to determine if an automatic average face can reach the quality and recognition rate of a manual average face.

4.1 Database

To apply the algorithms implemented during the previous episodes, a batch of 500 male celebrity images has been selected. All of them have been picked up under several light conditions, different periods of their lives and variant face gestures. All the images are 190x190 pixels in JPEG format grey-scale and contain busts. They have been extracted from public websites and were provided to this project by Dr. Rob Jeckins. Some of this samples can be displayed in Figure 4.1

4.2 Experiments realized

Several experiments have been done to demonstrate that the average face of a person is better than any random single face by itself. Beginning experiments compare the recognition rates of manual average faces versus automatic average faces to understand the level of similarity between them. Afterwards, some experiments compare the recognition rates between automatic average faces and random single faces. The generated faces are attached in Appendix ?? and some of them are also displayed in Figure 4.2.

1Psychology Department. University of Glasgow (2009)
Figure 4.1: Example of some single faces introduced in the face database to do the experiments.
It is possible to notice the differences between automatic and manual output images and, in conjunction with the following studies, the results achieved with this project. Finally time comparisons and the evolution of average faces will be done. Before start comparing the warped images, a simple comparison is done between a *warped average face* and a *non-warped average face*, to demonstrate that the deformation of the images to a frontal position before making the average operation is one of the main points of this investigation.

Figure 4.2: Average images generated by the application. Left images are manual average faces and right images are automatic average faces.
4.2.1 The importance of warping the images

For the first experiment, a group of non-warped faces has been created and a face recognition algorithm has been applied to them (see Figure 4.3 for some samples and Appendix ?? for the complete list of faces). To make the experiment complete, the number of faces used to generate the non-warped average is increased to understand the behaviour of the algorithm.

<table>
<thead>
<tr>
<th>Faces used to create the average</th>
<th>Faces recognised</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 faces</td>
<td>9/25</td>
<td>36.0%</td>
</tr>
<tr>
<td>10 faces</td>
<td>7/25</td>
<td>28.0%</td>
</tr>
<tr>
<td>15 faces</td>
<td>6/25</td>
<td>24.0%</td>
</tr>
<tr>
<td>20 faces</td>
<td>5/25</td>
<td>20.0%</td>
</tr>
</tbody>
</table>

Table 4.1: Results of recognition rate in non-warped faces. The number of images used to generate the average changes

Obviously, as long as the number of faces is increased, the output generated is more fuzzy and less similar to a face and therefore the recognition rate is smaller. This is the reason why the deformation followed in this project enhance the chances for a face to be recognized.

![Figure 4.3: Example of some non warped faces.](image)

The results in Table 4.1 present that the deformation of the images before the creation of an average is essential for the correct subsequent recognition.

4.2.2 Manual and automatic average faces

Comparison between automatic and manual average faces in different experiments to see how far they are one to the other.
4.2.2.1 First experiment: Average with included images

The first experiment includes 500 faces (20 images x 25 people) in the database and consist on the next steps:

1. Generate an automatic average faces for each celebrity with the photos included in the database.
2. Generate a manual average faces for each celebrity with the photos included in the database.
3. Run the eigenfaces algorithm for all the average images generated.
4. Compare results between manual and automatic average faces.

The results of the experiment are displayed in Table 4.2 and surprisingly return that the percentage of accuracy of the automatic average face is greater than the manual average. The high accuracy of the results in both cases demonstrates that the average face can be a very good representation for artificial face recognition. On the other hand, the reason why the automatic average seems to be better than the manual one is that some of the images generated automatically include just two or three images of each person because the rest have been rejected due to lack of information (shadows that hide eyes, moustaches, etcetera...) while in the case of manual faces, all of the generated faces have been created from 20 images of each celebrity.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Faces recognised</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual Face</td>
<td>403/500</td>
<td>80.6%</td>
</tr>
<tr>
<td>Automatic Face</td>
<td>481/500</td>
<td>96.2%</td>
</tr>
</tbody>
</table>

Table 4.2: Manual average face and automatic average face recognition rates for the first experiment

4.2.2.2 Second experiment: Average with not included images

The second experiment includes 375 faces (15 images x 25 people) in the database and consist on the next steps:

1. Generate an automatic average faces for each celebrity with the remaining 5 photos not included in the database.
2. Generate a manual average faces for each celebrity with the remaining 5 photos not included in the database.
3. Run the eigenfaces algorithm for all the average images generated.

4. Compare results between manual and automatic average faces.

Once again, the automatic faces generated are closer to the generated database than the manual faces (Table 4.2.2.2). The recognition rate has decreased because the faces to be recognised are not included in the database and therefore the generated face is not a combination of database faces.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Faces recognised</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual Face</td>
<td>123/375</td>
<td>32.8%</td>
</tr>
<tr>
<td>Automatic Face</td>
<td>152/375</td>
<td>40.53%</td>
</tr>
</tbody>
</table>

Table 4.3: Manual average face and automatic average face recognition rates

4.2.2.3 Third experiment: Database formed with averages. Single faces included

The third and forth experiments correspond to a change in the point of view of the problem. If instead of trying to recognise average faces in a single-face database, the average faces are the ones that create the database, results may change. In this case, the average faces create a subspace of images, reducing substantially the number of images inside the database (×20). The difference between both experiments is that in one the single images are part of the images used to generate the average and in the other one are not included.

The steps followed for the third experiment are:

1. Introduce all the manual averages of the 20 images in the database
2. Try to recognise single images (5) included in the database.
3. Introduce all the automatic averages of the 20 images in the database
4. Try to recognise single images (5) included in the new database.

The results obtained developing this method decrease the recognition rate significantly smaller as shown in Table 4.2.2.3.
4.2.2.4 Fourth experiment: Database formed with averages. Single faces not included

The steps followed for the fourth experiment are:

1. Introduce all the manual averages of the 20 images in the database
2. Try to recognise single images (5) not included in the database.
3. Introduce all the automatic averages of the 20 images in the database
4. Try to recognise single images (5) not included in the new database.

The results obtained developing this method decrease the recognition rate drastically as shown in Table 4.2.2.4.

4.2.2.5 Fifth experiment: Comparison between manual and automatic average faces

Other interesting aspect to compare both kind of averages (manual and automatic) is studying the correlation between both. That is, see how many automatic faces are recognised in a database of manual faces and vice versa.

The results in Table ?? show that there is not very different to create the database in one way or in the other because results are very close.

4.2.3 Automatic single faces and average faces

In this second part, a different aspect will be examined: the difference between search a single face or an average face inside a database. For this section, instead of the PCA method used before to recognise the faces, an external application is going to be used. As in [37], the website www.myheritage.com will be used. This website has an on-line application for face recognition as mentioned in Chapter 1 that detects faces inside photos and displayed the closest celebrities with a percentage of similarity.
4.2.3.1 First experiment: Study comparing single faces with automatic averages including non-matched faces

The first experiment compares the submission of a random single face and the submission of an automatic average face. For this, all the non-rejected images used to generate the automatic faces have been submitted and search in www.myheritage.com. In Figure 4.4 the results are displayed including all the faces that the website failed to recognise. Examining the mentioned Figure, there are 5 average face that where not recognised (George Clooney, Jack Nicholson, Will Smith, Sean Connery and Tom Cruise) while for the rest of the faces the average is more probable to be recognised (12/19) than the average single image (7/19) and just in one submission.
### Table 4.4: Database created with average faces using single included images to be recognised

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Faces recognised</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database of manual faces</td>
<td>63/125</td>
<td>50.4%</td>
</tr>
<tr>
<td>Database of automatic Face</td>
<td>50/125</td>
<td>40.0%</td>
</tr>
</tbody>
</table>

### Table 4.5: Database created with average faces using single included images to be recognised

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Faces recognised</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database of manual faces</td>
<td>10/125</td>
<td>8.0%</td>
</tr>
<tr>
<td>Database of automatic Face</td>
<td>13/125</td>
<td>10.4%</td>
</tr>
</tbody>
</table>

### Table 4.6: Correlation between both: manual and automatic faces

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Faces recognised</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database of manual faces</td>
<td>19/25</td>
<td>76.0%</td>
</tr>
<tr>
<td>Database of automatic Face</td>
<td>17/25</td>
<td>68.0%</td>
</tr>
</tbody>
</table>
Figure 4.4: Results comparing the recognition rate between a single face and the automatic average face (1).
4.2.3.2 Second experiment: Study comparing single faces with automatic averages without non-matched faces

Now if the failed hits are removed, is possible to compare the probability of a face to be recognised being an automatic average or just a random single face.

![Figure 4.5: Results comparing the recognition rate between a single face and the automatic average face (2).](image)

Examining now Figure 4.5, it is concluded that when the probability of a single image to be zero is reached, the probability of the average face is still 53.68%. That makes the average strong and more persistent against changes than a single image (as it is expected).

4.2.3.3 Third experiment: Study comparing single faces with automatic averages without non-matched faces

Now the failed hits are removed to understand how relevant they are for the results.

Finally, looking at Figure 4.6, where the failed hits have been removed for both variables, the average probability is increased slightly because failed hits were more detrimental to the average face than for the individual faces.

4.2.3.4 Forth experiment: Single non-frontal faces Vs. automatic average face

To demonstrate that the average face can be a better choice in order to recognise faces, non-frontal faces of each celebrity were introduced in the website to be recognised. The
Figure 4.6: Results comparing the recognition rate between a single face and the automatic average face (3).
results show that statistically 11 of 24 non-frontal images are not recognized while for average faces the numbers decreases until 4.

The results obtained developing this method decrease the single face recognition rate drastically as shown in Table 4.2.4.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Faces recognised</th>
<th>Recognition rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single non-frontal face</td>
<td>13/24</td>
<td>39.29%</td>
</tr>
<tr>
<td>Automatic average face</td>
<td>21/24</td>
<td>56.26% height</td>
</tr>
</tbody>
</table>

Table 4.7: Single non-frontal faces Vs. automatic average faces

### 4.2.4 Time comparisons

The main advantage of the automation of the warping algorithm is to save great amount of time. Because the nodes’s positioning is hard work and extremely slow, the automatic average face becomes more interesting than the manual average face. To understand the quantities of time expended for both cases, a simple experiment have been done. Time has been measured for both using the `cputime()` function in C/C++ which returns the actual time of the computer when the function is called. The statistical results of this experiment are displayed in Table ?? for 20 faces.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Time expended (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual average face creation</td>
<td>2340.46</td>
</tr>
<tr>
<td>Automatic average face creation</td>
<td>8.83</td>
</tr>
</tbody>
</table>

Table 4.8: Single non-frontal faces Vs. automatic average faces

It is straightforward to realise that the implementation of the automatic system decreases abruptly the time to generate an average face **265 times**.

### 4.2.5 Evolution of average faces

In this section, the behaviour of the recognition rate will be tested increasing the number of images used to create the average image as in [39]. In Figure 4.7 the evolution of the average can be seen. As long as the number of images that compound the average is increased, the recognition rate increases from 63.5% to 96.2% as in Table 4.2.
Figure 4.7: Results comparing the recognition rate between a single face and the automatic average face (3).
4.2.6 Rejection of faces

The most important problem of using automatic face averaging is the number of images rejected in the process. As mentioned before, in the case that in an image both eyes, nose and mouth are not recognised, this image is removed from the average image. This assumption makes manual images contain more information than the automatic faces. For this project, the average of images used for each face was $7.37 / 20$ (36.9%).
Conclusions and future works

5.1 Conclusions

In the present work, a new average facial generator has been designed combining triangle-mesh warping and Haar Cascades. A basic face recognition system has been developed using PCA and eigenfaces to determine the quality of the results obtained. To complete the investigation, an external face recognition tool provided by http://www.myheritage.com has been used in order to contrast the results.

With this study some important aspects are concluded:

1. The deformation of the faces before generating the average face increases rapidly the recognition rate because of the control of the position of the face features: eyes, mouth and nose.

2. The number of images used to create the average affects radically to the recognition rate increasing it when the adding new images of the subject to create the average.

3. When a face image is non-frontal, has badly light conditions or low resolution the average improves the recognition rate.

4. The time spent for the generation of a manual average face is 265 times the time necessary by the solution presented. Because 40 minutes to generate a manual average face is too long, this innovation makes the idea of automatic average faces to be feasible and useful.

5. The correlation between manual and automatic faces is fairly close, providing both similar results and being automatic average even better in some situations.
5.2 Future works

After the development of the application and the complete solution to the original planted problem, some future works can be done. Because the application has been developed in C++, all the posterior ideas should be implemented in that way, avoiding complex parsing functions or new bugs. To improve the application, some new ideas can be added and implemented.

- Increase the number of features of the face for the automatic average face. To do that, new Haar Cascades are required, including eyebrows, chin, etcetera...
- Implement the mesh warping algorithm with the Cat-Mull interpolation attached in Appendix A to check if the recognition ratio increases.
- Add new features to the graphic interface such as parameter options configuration, image format management, configuration registry to save the user preferences, improved image management based on thumbnails, etcetera...
- Optimize the source code to gain speed in hard functions like warping or Haar-like feature’s search.
- Implement a web-based application that generate average faces based on captured images via webcam.
- Development of a complete real-time application that controls and generates average faces of each person detected in the camera.
Cat-Mull Algorithm. Mesh interpolation

Cat-Mull algorithm in C

```c
/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
* catmullRom:
* 
* Compute a Catmull-Rom spline passing through the len1 points in arrays
* x1, y1, where y1 = f(x1)
* len2 positions on the spline are to be computed. Their positions are
* given in x2. The spline values are stored in y2.
*/
void
catmullRom(x1, y1, len1, x2, y2, len2)
float *x1, *y1, *x2, *y2;
int len1, len2;
{
    int i, j, dir, j1, j2;
    double x, dx1, dx2;
    double dx, dy, yd1, yd2, p1, p2, p3;
    double a0y, a1y, a2y, a3y;

    /* find direction of monotonic x1; skip ends */
    if(x1[0] < x1[1]) {/* increasing */
        if(x2[0]<x1[0] || x2[len2-1]>x1[len1-1]) dir=0;
```
else dir = 1;
} else {  /* decreasing */
    if(x2[0]>x1[0] || x2[len2-1]<x1[len1-1]) dir=0;
    else dir = -1;
}
if(dir == 0) {  /* error */
    printf("catmullRom: Output x-coord out of range of input\n");
    return;
}
/* p1 is first endpoint of interval
 * p2 is resampling position
 * p3 is second endpoint of interval
 * j is input index for current interval */

/* force coefficient initialization */
if(dir==1) p3 = x2[0] - 1;
else  p3 = x2[0] + 1;

for(i=0; i<len2; i++) {
    /* check if in new interval */
    p2 = x2[i];
    if((dir==1 && p2>p3) || (dir== -1 && p2<p3)) {
        /* find the interval which contains p2 */
        if(dir) {
            for(j=0; j<len1 && p2>x1[j]; j++);
            if(p2 < x1[j]) j--;
        } else {
            for(j=0; j<len1 && p2<x1[j]; j++);
            if(p2 > x1[j]) j--;
        }
    }
    p1 = x1[j]; /* update 1st endpt */
    p3 = x1[j+1]; /* update 2nd endpt */

    /* clamp indices for endpoint interpolation */
    j1 = MAX(j-1, 0);
    j2 = MIN(j+2, len1-1);
*/ compute spline coefficients */
dx = 1.0 / (p3 - p1);
dx1 = 1.0 / (p3 - x1[j1]);
dx2 = 1.0 / (x1[j2] - p1);
dy = (y1[j+1] - y1[j]) * dx;
yd1 = (y1[j+1] - y1[j1]) * dx1;
yd2 = (y1[j2] - y1[j]) * dx2;
a0y = y1[j];
a1y = yd1;
a2y = dx * (3*dy - 2*yd1 - yd2);
a3y = dx*dx*(-2*dy + yd1 + yd2);
}
/* use Horner’s rule to calculate cubic polynomial */
x = p2 - p1;
y2[i] = ((a3y*x + a2y)*x + a1y)*x + a0y;
}
Appendix B

HSI-to-RGB Conversion

Depending on the sector of the color circle, the formula changes.

For \(0^\circ \leq H < 120^\circ\),

\[
R = \frac{I}{\sqrt{3}} \cdot \left[1 + \frac{S \cdot \cos(H)}{\cos(60^\circ - H)}\right]
\]

\[
B = I \left(1 - S\right) \sqrt{3}
\]

\[
G = \sqrt{3} \cdot I - R - B
\]

while for \(120^\circ \leq H < 240^\circ\),

\[
G = \frac{I}{\sqrt{3}} \cdot \frac{1 + S \cdot \cos(H - 120^\circ)}{\cos(180^\circ - H)}
\]

\[
R = I \sqrt{3 (1 - S)}
\]

\[
B = \sqrt{3} \cdot I - R - G
\]

\[\text{(B.1)}\]

and for \(240^\circ \leq H < 360^\circ\),

\[
B = \frac{I}{\sqrt{3}} \cdot \frac{1 + S \cdot \cos(H - 240^\circ)}{\cos(300^\circ - H)}
\]
\[ G = I \frac{1}{\sqrt{3} \cdot (1 - S)} \]

\[ B = \sqrt{3} \cdot I - G - B \]

(B.2)
Appendix C

Schema of the Application

[Diagram showing the schema of the application with various widgets and functions]
D.1 Introduction to the application

StoneFace will allow you to create manual and automatic average faces and also to recognise them in a database. To do that, you just need to select what you want to do in the main menu and start working! The main menu is divided in three independent parts: two for face creation and a last one to detect a face as you can see in the next figure.

D.1.1 Average face creation

1. **Manual** This method will offer you the possibility of designing node by node the positions of the triangular mesh edges. You will be able to modify one by one the images and decide which are the most important features you want to highlight (such as the eyes, nose or mouth). The main disadvantage of this method is the time expended. You will need approximately (depends on the level of detail and the patience of the person who is producing the images) about two minutes to create a warped image.
2. **Automatic** This alternative method offers the possibility of generating automatically an average image introducing a large batch of images. The advantage is the speed: more than 20 images in less than a second. The problem is that because it is an automated procedure, only those pictures where both eyes, mouth and nose are detected will be accepted. Otherwise, the face image will be rejected. That means that an important number of faces would be rejected due to light conditions or extreme non-frontal position of the face.

### D.1.2 Database search

The last of the three options of the main menu. This form will allow you to recognise a face using eigenfaces and a face database created previously.

### D.2 Manual face creation

This first part of the application has been designed to involve in the process the user as much as possible. For this reason, all the menus and options have been decided to be clear and accurate. As soon as we select this option in the main menu, a complete new environment will be displayed.

This form is defined by 5 parts:

![Figure D.2: Layout of the manual face creator](image)

**Figure D.2:** Layout of the manual face creator

### D.2.1 Text menu
Contains all the options that the software offers. It is divided in 6 parts. Each one focused on a different area.

- **File** Open a batch of images, save the actual one, close the actual image and also close the application.
- **View** Show the mesh lines inside the picture and display the help face on the right area of 4
- **Mesh options** Move the nodes individually, in a group etc...
- **Operations** When the user wants to create an average or see how the actual mesh has been deformed yet.
- **Images** Control of the batch of images loaded. Move to the next face or return to the previous one.
- **Help** Information about the author and the libraries used.

### D.2.2 Main options toolbar

This toolbar contains the top used commands in order to save time navigating by the menus of the previous section. The options are displayed in Table D.2.2

### D.2.3 Secondary options toolbar

The purpose of this secondary toolbar is to play with the nodes when an image is loaded. To facilitate the management of the nodes, this toolbar has been divided in three different modules:

#### D.2.3.1 Single node administration

With the buttons provided, the user can move the selected node up, down, right and left 5 pixels each time. Pressing directly the keyboard keys ”up” ”down” ”left” or ”right” the same effect will be done.
<table>
<thead>
<tr>
<th>Icon</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Open" /></td>
<td><strong>Open</strong> - Open a batch of images to be treated individually by the user.</td>
</tr>
<tr>
<td><img src="image" alt="Save" /></td>
<td><strong>Save</strong> - Save the actual image with the actual mesh.</td>
</tr>
<tr>
<td><img src="image" alt="Warp" /></td>
<td><strong>Warp</strong> - In case the user wants to obtain an image of the deformation created before saving the image this option will generate it and shows it in an independent window.</td>
</tr>
<tr>
<td><img src="image" alt="Average" /></td>
<td><strong>Average</strong> - If the user wants to create an average of a batch of warped images this option will ask for an input group and a place to save the average image generated.</td>
</tr>
<tr>
<td><img src="image" alt="Webcam" /></td>
<td><strong>Webcam</strong> - Do you need to acquire an image in real time? This option lets the user capture an image from a connected webcam and treat it in the main window directly.</td>
</tr>
<tr>
<td><img src="image" alt="Previous image" /></td>
<td><strong>Previous image</strong> - Deformation of the actual image with the mesh designed and turn back to the previous face in the list.</td>
</tr>
<tr>
<td><img src="image" alt="Next image" /></td>
<td><strong>Next image</strong> - Deformation of the actual image with the mesh designed and advance to the next face in the list.</td>
</tr>
<tr>
<td><img src="image" alt="Show edges" /></td>
<td><strong>Show edges</strong> - Display or not the mesh lines in order to provide the best distribution of the nodes. This option can help sometimes to understand the deformation of some areas of the image.</td>
</tr>
<tr>
<td><img src="image" alt="Help view" /></td>
<td><strong>Help view</strong> - Display or hide the right face that highlights the node selected. This option is quite useful to understand which node is being selected.</td>
</tr>
</tbody>
</table>

Table D.1: Description of the toolbar icons in the Manual Form

### D.2.3.2 Group of nodes administration

With the first four buttons, the user can move the selected group of nodes up, down, right and left 5 pixels each time. With the next two buttons, the group will turn clock or anticlockwise. With the last two buttons, the distance between nodes in the group can be increased or decreased.

Groups are created by colour and they are provided in the table 1. Pressing directly the keyboard keys "a" for left movement, "d" for right movement, "w" for up movement or "s" for down movement the same effect would be done.
Table D.2: Landmarks/nodes referenced by colour in groups

<table>
<thead>
<tr>
<th>Colour</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue</td>
<td>Outline (jaw, ears, pinna, temples, scalps and crown)</td>
</tr>
<tr>
<td>White</td>
<td>Chin</td>
</tr>
<tr>
<td>Orange</td>
<td>Left eyebrow</td>
</tr>
<tr>
<td>Turquoise</td>
<td>Right eyebrow</td>
</tr>
<tr>
<td>Pink</td>
<td>Right eye</td>
</tr>
<tr>
<td>Yellow</td>
<td>Left eye</td>
</tr>
<tr>
<td>Red</td>
<td>Mouth</td>
</tr>
<tr>
<td>Green</td>
<td>Nose</td>
</tr>
<tr>
<td>Violet</td>
<td>Other landmarks like cheeks or forehead</td>
</tr>
</tbody>
</table>

D.2.3.3 All nodes administration

With these four buttons the user can move all the nodes at the same time. This option is very useful when the face is displaced inside the image a certain distance. Pressing directly the buttons "f" for left movement, "h" for right movement, "t" to move all nodes up and "g" to move them down.

D.2.4 Working Area

This zone corresponds to the area where the user interacts with the application during most of the time. Here the face image is plotted on the left and (optionally) a help face is represented on the right to understand the nodes selected. Because this area is a graphic area, some extra features can be done.

1. If the user launched the software from a terminal, the nodes position can be displayed in the terminal pressing key "m".

2. The key "l" displays / hide the actual mesh

3. Scrolling the mouse over a picture, a zoom in/out will be done

All nodes can be moved using the toolbar, keyboard or dragging and dropping them!

D.2.4.1 Status bar

This bar indicates the tips and descriptions of each one of the objects inside the window and other messages as warnings or progress.
Figure D.3: Example of a real face processing. The left area contains the desired warping image while the right one contains the help face with a red node advising us the last node selected.

These messages will appear just during a period of two or three seconds to alert the user about a new situation. If an important error happen or an question is generated, the software will create a message dialog to interact with the user. This messages are described in the next point.

During the life of the application several messages can appear on the screen. Each one of them has a different purpose. Most relevant are enumerated in the following list with a short explanation.
### D.3 Automatic face creation

The second module of the application consists of a simple form where first the user is asked for a batch of images as an input. Then some information is displayed about the images selected and afterwards some output parameters can be selected.
Appendix D. User’s Manual

D.3.1 Face batch selection

This is the first step. When the button Search for face files... is pressed an external window will appear waiting for a selection of the images to load. In the next figure the dialog is displayed. It is important to notice that it is possible to select more than one image at the same time. That means, it is multi-selection.

D.3.2 Output configuration

The program offers as an option to modify the standard conditions of the output image generated.

- **Size** By default the images will be generated at 180x180 if the text boxes are blank, otherwise the introduced values will be taken into account.
• **Grey scale** It is possible to convert the output image into a grey scale image.

> The grey scale algorithm used is just as the average of R + G + B for each pixel.

• **Save path** A button that launches a dialog window to select the name of the image that is going to be created.

### D.3.3 Current information

During the scanning of images and the automated process, some information of each face is displayed such as the position of the image in the list, the number of colours and the size of the original image. In the figure 6 an example of real time process is demonstrated.

### D.3.4 Input intermediate image before warping

This image contains the original image before it is warped. Just resizing is done.

### D.3.5 Output intermediate image after warping

This image contains the final individual image after it is warped automatically. It is easy to compare them and find the differences for each face.

### D.3.6 Progress

The progress bar represents the number of faces processed over the total number of faces in the list. This object can display easily the real state of the process and internal subroutines.

### D.4 Face recognition. Search a face.

The last module of the application is designed to complete the investigation. Here the user can provide a face database and the software will find by itself the closest face inside the database introducing a given face. The database must be introduced following a valid eigenfaces list format (xml file with standard tags).
Appendix D. User’s Manual

Figure D.6: Automatic face averaging

Figure D.7: Layout of the search face form

D.4.1 Database selection

Introduce the path of the database file. Some information about the eigenfaces stored inside the database is presented, also the number of images and the date of creation of the database.

D.4.2 Face to recognise selection

Introduce the path of the face to recognise after pressing the button Select face. Some information about the image is presented (number of colours, size, number of bytes, etc...)
D.4.3 Matched face report

Once the image is selected and the corresponding button is pressed (Recognise this face) and the application has found the closest face, some information about the individual will be displayed (age, profession, real name or link to find more information)\(^1\).

D.4.4 Matched face

This image contains the resulting image. This image will match with the closest person of the database.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{example_face_recognition}
\caption{Example of face recognition in real time}
\end{figure}

\(^1\)All this information has been extracted from www.wikipedia.org
Manual face mesh

Coordinates of the nodes and description for location
<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Chin centre</td>
<td>186</td>
<td>522</td>
</tr>
<tr>
<td>2</td>
<td>Chin left</td>
<td>148</td>
<td>505</td>
</tr>
<tr>
<td>3</td>
<td>Chin right</td>
<td>219</td>
<td>506</td>
</tr>
<tr>
<td>4</td>
<td>Jaw left</td>
<td>85</td>
<td>472</td>
</tr>
<tr>
<td>5</td>
<td>Jaw right</td>
<td>287</td>
<td>472</td>
</tr>
<tr>
<td>6</td>
<td>Ear left</td>
<td>42</td>
<td>369</td>
</tr>
<tr>
<td>7</td>
<td>Ear right</td>
<td>330</td>
<td>366</td>
</tr>
<tr>
<td>8</td>
<td>Pinna left</td>
<td>37</td>
<td>301</td>
</tr>
<tr>
<td>9</td>
<td>Pinna right</td>
<td>337</td>
<td>310</td>
</tr>
<tr>
<td>10</td>
<td>Temple left</td>
<td>41</td>
<td>258</td>
</tr>
<tr>
<td>11</td>
<td>Temple right</td>
<td>333</td>
<td>257</td>
</tr>
<tr>
<td>12</td>
<td>Scalp left</td>
<td>51</td>
<td>100</td>
</tr>
<tr>
<td>13</td>
<td>Scalp right</td>
<td>309</td>
<td>97</td>
</tr>
<tr>
<td>14</td>
<td>Crown</td>
<td>190</td>
<td>59</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Blip_B_C</td>
<td>185</td>
<td>438</td>
</tr>
<tr>
<td>16</td>
<td>Blip_B_L</td>
<td>156</td>
<td>434</td>
</tr>
<tr>
<td>17</td>
<td>Blip_B_R</td>
<td>214</td>
<td>435</td>
</tr>
<tr>
<td>18</td>
<td>Blip_T_C</td>
<td>186</td>
<td>422</td>
</tr>
<tr>
<td>19</td>
<td>Blip_T_L</td>
<td>152</td>
<td>419</td>
</tr>
<tr>
<td>20</td>
<td>Blip_T_R</td>
<td>215</td>
<td>419</td>
</tr>
<tr>
<td>21</td>
<td>Tip_B_C</td>
<td>187</td>
<td>415</td>
</tr>
<tr>
<td>22</td>
<td>Tip_B_L</td>
<td>169</td>
<td>413</td>
</tr>
<tr>
<td>23</td>
<td>Tip_B_R</td>
<td>203</td>
<td>413</td>
</tr>
<tr>
<td>24</td>
<td>Tip_T_C</td>
<td>187</td>
<td>406</td>
</tr>
<tr>
<td>25</td>
<td>Tip_T_L</td>
<td>169</td>
<td>404</td>
</tr>
<tr>
<td>26</td>
<td>Tip_T_R</td>
<td>203</td>
<td>404</td>
</tr>
<tr>
<td>27</td>
<td>Lips_left</td>
<td>136</td>
<td>417</td>
</tr>
<tr>
<td>28</td>
<td>Lips_right</td>
<td>241</td>
<td>419</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>Septum</td>
<td>187</td>
<td>369</td>
</tr>
<tr>
<td>30</td>
<td>Left nostril</td>
<td>164</td>
<td>360</td>
</tr>
<tr>
<td>31</td>
<td>Right nostril</td>
<td>210</td>
<td>363</td>
</tr>
<tr>
<td>32</td>
<td>Left flange</td>
<td>145</td>
<td>353</td>
</tr>
<tr>
<td>33</td>
<td>Right flange</td>
<td>228</td>
<td>353</td>
</tr>
<tr>
<td>34</td>
<td>Tip</td>
<td>186</td>
<td>345</td>
</tr>
<tr>
<td>35</td>
<td>Left valley</td>
<td>160</td>
<td>334</td>
</tr>
<tr>
<td>36</td>
<td>Right valley</td>
<td>214</td>
<td>332</td>
</tr>
<tr>
<td>37</td>
<td>Bridge</td>
<td>188</td>
<td>271</td>
</tr>
</tbody>
</table>
## Appendix E. Details of the manual face mesh

### REGION 4: Eyes

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>Lpupil</td>
<td>118</td>
<td>268</td>
</tr>
<tr>
<td>39</td>
<td>Liris_TO</td>
<td>105</td>
<td>266</td>
</tr>
<tr>
<td>40</td>
<td>Liris_TI</td>
<td>133</td>
<td>265</td>
</tr>
<tr>
<td>41</td>
<td>Llid_LO</td>
<td>105</td>
<td>280</td>
</tr>
<tr>
<td>42</td>
<td>Llid_LI</td>
<td>133</td>
<td>279</td>
</tr>
<tr>
<td>43</td>
<td>Llid_TO</td>
<td>94</td>
<td>261</td>
</tr>
<tr>
<td>44</td>
<td>Llip_TI</td>
<td>118</td>
<td>254</td>
</tr>
<tr>
<td>45</td>
<td>Louter fold</td>
<td>78</td>
<td>275</td>
</tr>
<tr>
<td>46</td>
<td>Ltear duct</td>
<td>153</td>
<td>276</td>
</tr>
<tr>
<td>47</td>
<td>Lbag outer</td>
<td>118</td>
<td>296</td>
</tr>
<tr>
<td>48</td>
<td>Lbag inner</td>
<td>148</td>
<td>285</td>
</tr>
<tr>
<td>49</td>
<td>Rpupil</td>
<td>260</td>
<td>268</td>
</tr>
<tr>
<td>50</td>
<td>Riris_TO</td>
<td>276</td>
<td>265</td>
</tr>
<tr>
<td>51</td>
<td>Riris_TI</td>
<td>245</td>
<td>265</td>
</tr>
<tr>
<td>52</td>
<td>Rlid_LO</td>
<td>276</td>
<td>280</td>
</tr>
<tr>
<td>53</td>
<td>Rlid_LI</td>
<td>245</td>
<td>278</td>
</tr>
<tr>
<td>54</td>
<td>Rlid_TO</td>
<td>286</td>
<td>262</td>
</tr>
<tr>
<td>55</td>
<td>Rlip_TI</td>
<td>260</td>
<td>254</td>
</tr>
<tr>
<td>56</td>
<td>Router fold</td>
<td>300</td>
<td>276</td>
</tr>
<tr>
<td>57</td>
<td>Rtear duct</td>
<td>227</td>
<td>278</td>
</tr>
<tr>
<td>58</td>
<td>Rbag outer</td>
<td>261</td>
<td>297</td>
</tr>
<tr>
<td>59</td>
<td>Rbag inner</td>
<td>234</td>
<td>289</td>
</tr>
</tbody>
</table>

### REGION 5: Eyebrows

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>Lbrow inner</td>
<td>157</td>
<td>247</td>
</tr>
<tr>
<td>61</td>
<td>Lbrow TI</td>
<td>148</td>
<td>237</td>
</tr>
<tr>
<td>62</td>
<td>Lbrow TM</td>
<td>118</td>
<td>229</td>
</tr>
<tr>
<td>63</td>
<td>Lbrow LI</td>
<td>145</td>
<td>249</td>
</tr>
<tr>
<td>64</td>
<td>Lbrow LM</td>
<td>118</td>
<td>239</td>
</tr>
<tr>
<td>65</td>
<td>Lbrow outer</td>
<td>74</td>
<td>249</td>
</tr>
<tr>
<td>66</td>
<td>Rbrow inner</td>
<td>222</td>
<td>246</td>
</tr>
<tr>
<td>67</td>
<td>Rbrow TI</td>
<td>229</td>
<td>235</td>
</tr>
<tr>
<td>68</td>
<td>Rbrow TM</td>
<td>261</td>
<td>228</td>
</tr>
<tr>
<td>69</td>
<td>Rbrow LI</td>
<td>235</td>
<td>247</td>
</tr>
<tr>
<td>70</td>
<td>Rbrow LM</td>
<td>260</td>
<td>238</td>
</tr>
<tr>
<td>71</td>
<td>Rbrow outer</td>
<td>309</td>
<td>248</td>
</tr>
</tbody>
</table>

### REGION 6: Landmarks

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>Chin tip</td>
<td>186</td>
<td>487</td>
</tr>
<tr>
<td>73</td>
<td>Chin top</td>
<td>186</td>
<td>462</td>
</tr>
<tr>
<td>74</td>
<td>Chin left</td>
<td>147</td>
<td>472</td>
</tr>
<tr>
<td>75</td>
<td>Chin right</td>
<td>224</td>
<td>472</td>
</tr>
<tr>
<td>76</td>
<td>Line left</td>
<td>113</td>
<td>385</td>
</tr>
<tr>
<td>77</td>
<td>Line right</td>
<td>255</td>
<td>385</td>
</tr>
<tr>
<td>78</td>
<td>Cheek left</td>
<td>84</td>
<td>334</td>
</tr>
<tr>
<td>79</td>
<td>Cheek right</td>
<td>285</td>
<td>337</td>
</tr>
<tr>
<td>80</td>
<td>Forehead centre</td>
<td>190</td>
<td>199</td>
</tr>
<tr>
<td>81</td>
<td>Forehead left</td>
<td>105</td>
<td>185</td>
</tr>
<tr>
<td>82</td>
<td>Forehead right</td>
<td>270</td>
<td>185</td>
</tr>
</tbody>
</table>
Complete list of average faces
Appendix F. Complete list of average faces

85
Appendix F. Complete list of average faces
Bibliography


http://davis.wpi.edu/~matt/courses/morph/2d.htm


