Face Recognition Using FPGA Techniques

Diego González González

November 12, 2005

Abstract

In recent years there has been a great interest in face recognition techniques. While many of these techniques perform very well under controlled conditions it is very difficult to solve the more general problem in which pose, expression and lighting vary significantly. Whilst systems have been developed that address face detection, feature extraction and face recognition these systems operate at very low frame rates.

In this work a color-segmentation technique is used for face-detection, a method based on integral projections is used for face normalization and a modified "Eigenface" approach was selected for face recognition. Usually these systems have a low performance as require of big computers, one of the aims of the thesis is to place as much as possible into a Field Programmable Gate Array (FPGA), with the hope to be able to embedded the system in a chip and to improve its performance.

FPGA design imposes certain constrictions of its own to the problem that needs to be solved, these are based on the fact that FPGA have limited capacity and trade offs have to be done in order to improve speed or size, also there are parts of the original algorithms that will need to be changed in order to implement and keep them performant.

Contents

List of Figures

Chapter 1

Introduction

Automated face recognition systems have been a topic of great interest for a long time, interesting applications are robotics [14], automatic systems that act depending on the person that addresses them, human-machine interfaces [2], domotic systems [13] and security applications [12] which are experiencing a big growth.

A system for face recognition is composed by several subsystems, each being very important for the successful recognition of a known face.

- **Face detection**. Whose aim is finding the faces available in a picture or frame, different techniques can be applied depending on the type of input (a single image or a video). The output of these stage is a list of rectangular regions in which each face face lies.
- **Face normalization**. Before a face is fed to the recognition system, it should be normalized. Normalization implies one or more transformations, the final goal is to have the eyes at a certain distance for one another and in a given position, also the face needs to be of certain size. This step greatly improves the performance of the next stage, face recognition. To achieve the goal just exposed the face limits have to be found, and the eyes located, for a finer adjustment the center of the eyes also needs to be located.
- **Face recognition**. Once the face has been isolated and normalized it is given as an input to the face recognition system, whose only task it is to find a match between the input and the database of faces it has. The output of this step is a signal that encodes whether the face detected is known or not. The signal might be used as input to a system that can perform some action, e.g. opening a door depending on whether the face was recognized or not.

In the following sections each of these stages will be introduced briefly, a summary of the available techniques will be presented, next the method selected will be explained in depth and some parts of the algorithms developed in Matlab will be given. Afterwards, Field Programmable Gate Array (FPGA) devices are introduced, to continue with an introduction to the problems found when implementing the face recognition system and solutions taken to those.

Chapter 2

Face Detection

Face detection is the first step that any face recognition system should perform. Detecting faces has been a research topic for more than 20 years now, however the general problem remains unsolved due to the many uncontrolled variables that can change in a significant way the images taken: illumination, occlusions, face orientation and position (up-right, rotated), pose (profile, frontal) and facial expression. However the systems in which face detection is needed impose a set of constraints that make the problem easier to solve.

Face detection systems are given an input image in which they try to locate the faces and return the position and extents of each face found. The methods developed so far can be classified in different groups:

- **Knowledge-based methods**. These algorithms are based in a set of rules that encode human knowledge about what a face is. Whilst is easy to come up with rules it is a lot more difficult to translate these rules into a working system with precision. These systems usually start looking for simple features such as the eyes and then look for the face around them, if the background is complex they can easily found false-positives.
- **Feature invariant approaches**. These methods look for structural features of a face even when conditions change (pose, viewpoint, lighting, etc.). Many of these approaches are based on the fact that it is effortless for the brain to detect a face in an image, thus the first steps must imply very simple processing. Systems have been build that look for lines, circles, textures and colors. The most successful approach in this group is that based in color-skin segmentation (used when color images are available) or the Viola-Jones detector [7] (when gray level images are given).
- **Template matching methods**. This approach doesn't try to find individual components of a face, instead it considers the face as a whole. To achieve a detection a set of hand-build templates that encode facial appearance and pose are provided to the system, pattern classification techniques are used to compare the templates against the input images. Although quite easy to implement, these methods lack robustness.
- **Appearance-based methods**. The algorithms in this group try to build a template by themselves, this being the main difference with the template matching methods introduced before. Several ways have been tried to build the templates and classify the input image, from statistical-based algorithms (such as Principal Component Analysis -PCA- and Independent Component Analysis -ICA-) to biological based approaches (such as Neural Networks), other systems tried have been Bayesian classifiers, Hidden Markov models and Support Vector Machines (SVM).

From the methods presented those based on color-skin segmentation are the most popular, simplicity being the main reason behind this approach but also high performance and low resource consumption, given these advantages this method was selected for the face detection subsystem used in this thesis.

2.1 Color-Skin based face detection

Color skin segmentation provides useful information about what pixels in an image have skin color or are very close to it, as human skin-color, no matter which race, lies in a relatively tight cluster in chrominance space. This method is pose invariant, but produces a number of false-positives due to objects' color being close to that of the skin. The method proposed is composed of two steps that try to reduce the number of false positives: morphological operations are applied to the image to remove the noise, afterwards a connectivity analysis is performed and then the regions are analyzed to select which of them belong to a face. Figure 2.1 show the work-flow of the system.

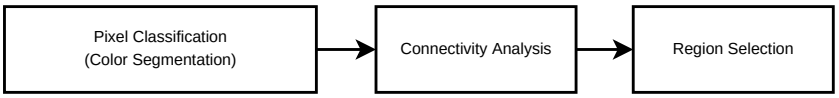
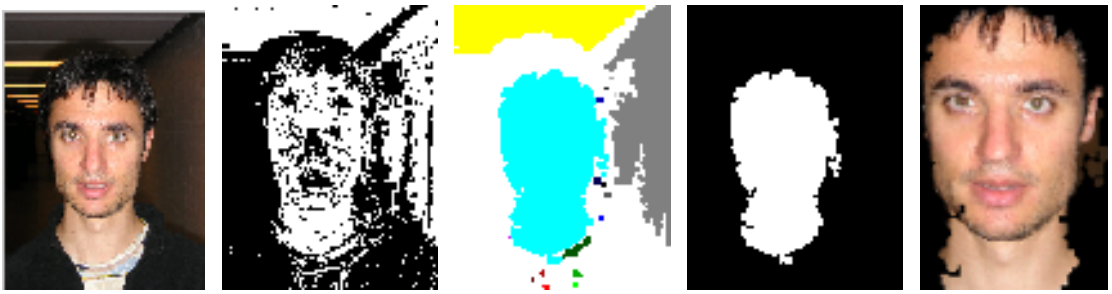


Figure 2.1: Face detection pipeline

Pixel Classification

Several color spaces can be used to perform the classification of the skin pixels, the most commonly used are: HSV and YCbCr or YUV. The last two are very similar and are frequently interchanged, and they match human color perception more closely than the standard RGB, which has its color cues tightly coupled.



(a) Input image (b) Mask before region analysis (c) Mask after region analysis (d) Final mask (e) Output image

Figure 2.2: Image in different steps of the face detection pipeline.

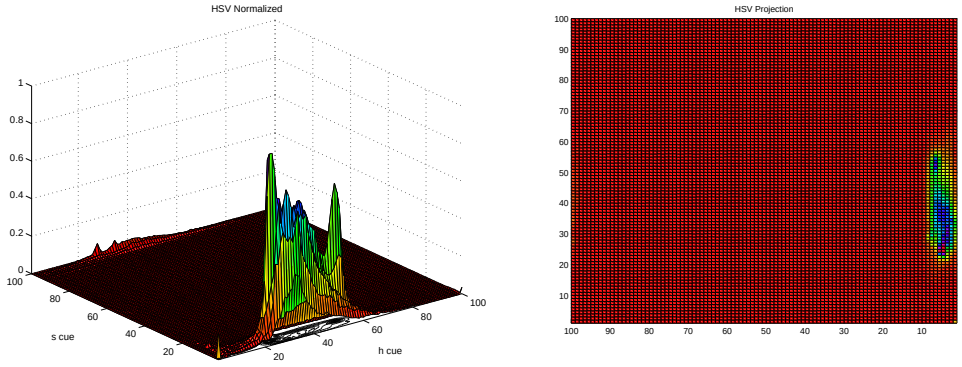
In HSV space the H cue stands for hue, which describes the color, the S cue or saturation describes how pure the color is, the last cue is value and describes how bright the color is, thus if this model is selected the V cue can be ignored and the resulting system will be relatively robust against changes in illumination. In YCrCb space the Y cue stands for luminance which describes the brightness of the color, Cb and Cr are Blue and Red Chrominance values respectively, the reason for this separation is that the eye is less sensitive to chrominance than to luminance, thus the chrominance values can be subsampled 2.1.

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2.1}$$

The YUV color space is very close to that described by YCrCb and sometimes is also used, in YUV space Y cue stands for luma or luminance, U and V cues provide color information and are "color difference" signals $U = B - Y$ and $V = R - Y$. Equation 2.2 describes the color transformation.

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2.2}$$

To classify the pixels a histogram is built, such histogram contains the frequencies of the color pixels in Cr-Cb or H-S space, after normalization each of the values lie in the 0-1 range, the histogram can be though of as begin the probability of a color in Cr-Cb or H-S space to be a skin color. Currently the training set is composed of about 618000 skin pixels, this is a very small quantity but given the fact that it is not the goal of this thesis to develop a new face detection algorithm it will be enough for demonstration purposes. Figure 2.3 shows the histogram in 2D and 3D views.



(a) (b)

Figure 2.3: 3D and 2D representation of RGB to YCrCb surface

Connectivity Analysis

Using skin classification one only knows which pixels are skin or are skin color-like, a higher level analysis is needed to know which pixels really belong to a face. We have to form meaningful groups of pixels, it makes sense in this analysis to remove regions that are too small and to merge regions that are very close to each other into a single one. For this purpose 8-connected neighborhood is required to classify pixels into regions.

There will be a high number of very small regions that can be considered as noise, to remove such regions a morphological open operation is performed (morphological open is an erosion followed by a dilation) using a disk shaped structuring element of radius 3 pixels. To improve the connectivity (it usually happens that a face might be divided into several regions that are very close to each other) a morphological dilation is performed and then the holes are filled.

Region Selection

After performing the last step only big connected regions should be left, for each region the area, bounding box and inertia moments are calculated, we are interested in regions that have an aspect ratio close to the golden ratio:

$$r = \frac{1+\sqrt{5}}{2} \pm tolerance \tag{2.3}$$

Also the area of the figure will be analyzed, regions taking less than 20% of the image are automatically rejected as face candidates. Using the minor axis of the ellipse defined by the inertia moments the image will be rotated, so that the minor axis is horizontal, this step makes it easier to look for eyes in the normalization step.

2.2 Algorithm

Figure 2.4 shows a high level view of the already explained algorithm.

Figure shows the image at each processing steps in the pipeline.

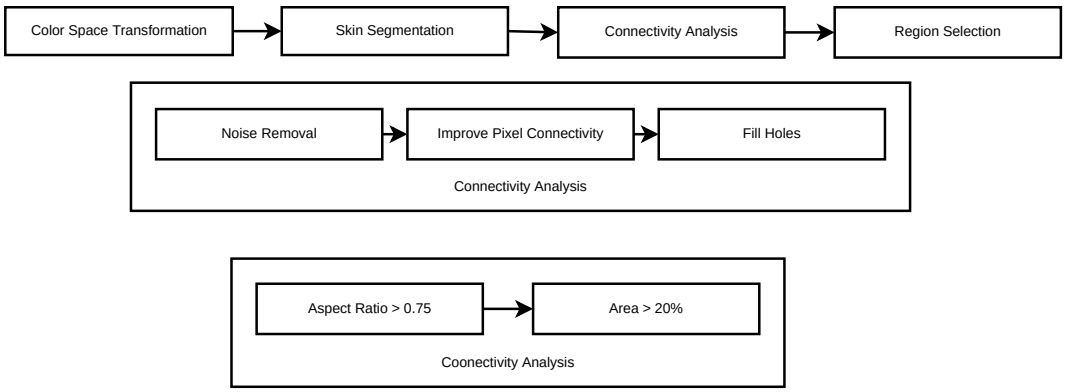


Figure 2.4: Face detection algorithm workflow

Chapter 3

Face Normalization

One of the problems with face recognition is that pictures of the face taken at different moments without any constraint can vary enormously in size, position, illumination and facial expression. Thus, to compare an input face image with those of the system some transformations have to be done to the input image. To solve the first set of problems a normalization stage is required in which several facial features are found and used to do size and position transformations on the original image.

Many exist for feature extraction, from using neural networks, algebraic and statistical methods (eigenfeatures), methods based on wavelets (mainly Gabor and Haar) and others build upon geometrical knowledge. The technique used in this work, integral projection, uses geometrical knowledge to locate the area where the eyes lay, afterwards to precisely locate the center of each eye a new method is used based on intensity projection. The distance between both eyes is used to normalize the position, rotation and size of the face. This method requires the rotation of the face to be quite small, which is achieved with an initial rotation using the inertia moments calculated in the previous stage.

3.1 Integral projection

Integral projection was a method first used by Takeo Kanade in his work on recognition of human faces, afterwards the method was improved using binary dominance maps (generated by performing edge detecting with algorithms sensible to horizontal and vertical segments). Vertical and horizontal integral projection functions are defined in equations 3.2 and 3.2, where $I(x, y)$, is the input image of dimension $M = r \times c$, r is the number of rows and c the number of columns.

$$V(x) = \sum_{y=1}^{c} I(x, y) \tag{3.1}$$

$$H(y) = \sum_{x=1}^{r} I(x, y) \tag{3.2}$$

Using these expressions combined with the binary dominance maps we can isolate with regions of the face that are interesting to us only by studying the projections.

Extraction of face region

The algorithm described here is based on the one proposed by Y.-S. Ryu and S.-Y. Oh in [11] with a some minor modifications to improve the boundary selection.

Calculating the binary vertical and horizontal dominance maps (trought the use of an edge detector operator that can discriminate between horizontal and vertical segments, like the sobel operator) and applying the integral projection method previously discussed the main features of the face can be extracted.

If the horizontal projection is calculated using the horizontal dominance map and the fact that in the hair and mouth regions the horizontal components dominate. Thus if we look in the upper face region for a spike in the projection we should find the hair. Also, looking for the highest value in the bottom half part of the face image should provide us with the vertical mouth position.

$$y_H = arg\max_j H_h(j), \qquad 1 \leq j \leq \frac{r}{3} \tag{3.3}$$

$$y_L = arg\max_j H_h(j) + \Delta y, \qquad \frac{r}{2} \leq j \leq r \tag{3.4}$$

To calculate the horizontal limits (i.e. right and left sides) of the face the vertical dominance map an its vertical projection are used, the faces is divided in two halfs, in the left we look for a maximum in the projection, this is the left side of the face, the same method is applied to the right side of the face.

$$x_R = arg\max_i H_v(i), \qquad 1 \leq i \leq \frac{c}{2} \tag{3.5}$$

$$x_R = arg\max_i H_v(i), \qquad \frac{c}{2} \leq i \leq c \tag{3.6}$$

With these for limits we can obtain the an extent that is the face region, in this region we will perform the search for the eyes. Note that from now on all the operations will be performed in this extent if not specified.

Extraction of eyes

In order to look for the eyes it is supposed that they will be found on the upper half of the extent. To properly find the eyes it is needed to separate them from the eyebrows, this is done using the information on the binary maps and taking into account that the eyebrows contain mostly horizontal components. This is, we will look for maximum in the vertical dominance map, in the region that goes from $\frac{1}{4}$ to $\frac{2}{3}$ of the extent's height.

$$H_hv(j) = \sum_{i=1}^{c} I_{VE}(i,j), \qquad \frac{1}{4}(y_L - y_H) \leq j \leq \frac{2}{3}(y_L - y_H) \tag{3.7}$$

$$E_c = arg\max_j H_{hv}(j). \tag{3.8}$$

The horizontal line defined by E_c is the row with the strongest vertical edge component within the region containing both eyes, the eyes are around this region, considering a tolerance up and down this line a extent can be defined where the eyes lay.

3.2 Fine location of eye center

3.3 Algorithm

[11]

Chapter 4

Face Recognition

Face recognition tries to match the face given as input with one of the faces that the system knows about, and output a signal that will encode whether the face is known or not.

Many systems have been developed that try to cope with some aspects of the variability of the input faces: expressions, occlusions, pose, position and so on, but up until now no system is able to deal with all the variations, it is a field where a lot of research is being done. To perform the matching several techniques have been developed [15] which can be classified into several groups:

- **Subspace based methods**. Faces often share a lot of characteristics this fact makes it possible to map a high-dimensional pixel array to a subspace of lower dimension. These methods use statistical and algebraic tools to extract and analyze the underlying subspace. Some algorithms that lay in this group are: Principal Component Analysis (PCA), Independent Component Analysis (ICA) and Linear Discriminant Analysis (LDA).
- **Feature based (structural) matching methods**. Local features are localized and extracted, its location and statistics are used for matching. Techniques such as neural networks, hidden Markov models and pure geometrical methods can be applied.
- **Hybrid methods**. The brain uses both models to analyze faces, the problem of such approach would be the computational power needed to perform them, but the performance of these methods combined should be better than the previous ones.

From these groups the first one has seen most of the research, quite a lot of approaches have been tried in this field, each one having its goodness and flaws. The better known method is that called Eigenfaces [1] which laid the foundations for all the existing methods, it is also a method that even today, 15 years after is original publication, exhibits great performances and requires an acceptable computing power.

The LDA[10] based method requires a lot more computing resources and to surpass the Eigenfaces method it requires a several images per subject, which is not a real life requirement[8]. Other methods have not seen such a warm acceptation, which might be due to its complexity, Independent Component Analysis has been one of them [9].

After analyzing the methods available the Eigenfaces was chosen due to its low complexity and real life requirements (it doesn't need a great amount of pictures per subject), and there is at least one published method on how to use only one picture per subject[16]).

4.1 Eigenfaces

The eigenface method tries to project the input image into a subspace whose orthogonal basis is composed by a set of vectors, also known as eigenfaces. This vectors contain the most significant features from a set of training faces. A face projected into the subspace is encoded by a set of weights, which are used to compare the image in the face space.

After doing this brief and rough introduction, the Eigenface method is going to be described in depth from a mathematical point of view.

The eigenface approach tries to extract the set of facial regions that are most different in the set of faces and to encode this variation, the regions just described can be considered features (note that the set of features that the algorithm may choose are not necessarily the ones that human would consider features - eyebrows, eyes, nose, mouth -). Mathematically this means that we are looking for the principal components of the distribution of faces, or the eigenvectors of the covariance matrix of the set of training images. These vectors will be ordered according to the amount of variation that they encode. If we represent each eigenvector as an image we can see a set of ghostly images, which are called eigenfaces. Figure 4.1 shows the steps that are needed to initialize the eigenfaces algorithm.

1. Get the set of training faces
2. Calculate eigenfaces from the training set
3. Select those eigenfaces whose eigenvalue is higher
4. Calculate the weight vector of each known individual

Table 4.1: Eigenface initialization algorithm

Figure 4.2 shows the steps performed when a new input face is fed into the face recognition system.

1. When a new face is presented project it into the face space
2. Calculate the distance from this new face to the others in the face space
3. A low value means that the face is know, otherwise it is not known to the system

Table 4.2: Eigenface matching procedure

Calculating Eigenfaces

We would like to find a subspace in which the basis is orthogonal and will maximize the variance in the training set. Principal Component Analysis, sometimes called Karhunen-Loeve transformation (KL), is used to find the linear representation of faces using only covariance of data and determines the set of orthogonal components (also known as eigenvectors, feature vectors or eigenfaces) which minimize the reconstruction error of a set of input vectors (training set).

Let $\Gamma = [\Gamma_1 \quad \Gamma_2 \quad ... \quad \Gamma_M]$ be the training set with M faces, let each face in the set be of size $m = w \times h$, being m the dimension of each face. To ensure that only significant features are captured in to the model the general face information should be removed, which is achieved calculating the average face vector and subtracting it from each of the faces in the training set, this results in a new set of face images Φ, see equations 4.2 and 4.2.

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i \tag{4.1}$$

$$\Phi_i = \Gamma_i - \Psi \tag{4.2}$$

To compute the basis for the images in set Φ we define a matrix of dimension $m \times M$ in which each column corresponds to each face in the training set Φ (each face is "stacked", one column will be placed after another creating a vector), next the covariance is calculated as shown in equation 4.4. The constant $\frac{1}{M}$ in equation 4.4 is introduced for convenience.

$$\widetilde{\Phi} = \frac{1}{\sqrt{M}}[\Phi_1 \quad \Phi_2 \quad ... \quad \Phi_M] \tag{4.3}$$

$$C = \sum_{i=1}^{M} \widetilde{\Phi}_i \widetilde{\Phi}_i^T \tag{4.4}$$

The only thing left to do is to calculate the eigenvectors u_i and eigenvalues λ_i of the covariance matrix as shown in equation 4.6.

$$Cu_i = \lambda_i u_i \tag{4.5}$$

$$C = \sum_{i=1}^{M} u_i u_i^T \tag{4.6}$$

However there exists a pitfall in the approach just exposed, it is a computationally intractable problem due to the size of covariance matrix (assume the we have a face of size 100×100 and M images in the training set, the covariance matrix would be of dimension 100000^2 which would take around 800 megs of ram and a very powerful processor), the key to a solution is to realize that the eigenvectors of $\widetilde{\Phi}_i \widetilde{\Phi}_i^T$ can (an $m \times m$ matrix) be found from the eigenvectors of $\widetilde{\Phi}_i^T \widetilde{\Phi}_i$ (an $M \times M$ matrix). The covariance matrix can be calculated as:

$$C = \sum_{i=1}^{M} \widetilde{\Phi}_i^T \widetilde{\Phi}_i \tag{4.7}$$

Also as the number of data points in the image space is less than the dimension of the space $M \ll m$, there will be only $M - 1$ meaningful eigenvectors, thus, we can calculate the covariance matrix in the new way exposed without loosing any generality, as the eigenvectors whose eigenvalues are bigger will be preserved. To recover the eigenfaces we can apply equation 4.8.

$$E = \frac{1}{\sqrt{|\lambda|}} \widetilde{\Phi} u_i \tag{4.8}$$

In this way we can recover the eigenfaces from the modified equation (which is reconstructed using its SVD). A method has been presented that permits the computation of eigenfaces from a reduced covariance matrix, after applying the modification the matrix is of size M^2, this it is dependant on the number of faces in the training set, if the training set is very big there exists some approaches to solve the problem [5].

Choosing the basis

For a face recognition system a good reconstruction is not required, it is only necessary that the features distinguish between those known and not known individuals [6], thus the basis can be trimmed down quite a bit. If the eigenfaces with the higher associated eigenvalues are removed then the reconstruction error will be very high, if we remove those eigenfaces with the lower eigenvalues the reconstruction error will be small. If fact, the reconstruction error when using the PCA method is a combination of the truncated eigenvalues, see equation 4.9, where s are the number of faces selected.

$$error = \sum_{i=s+1}^{M} \lambda_i \tag{4.9}$$

It has also be shown that if the first three eigenfaces are removed from the set of selected eigenfaces the accuracy of the recognition improves because the first eigenvectors seem to represent changes in illumination [17], which is a bit surprising because in this way the reconstruction error would be quite big.

During the years several other techniques have been published, this eigenvector selection techniques try to improve the recognition rate of the system under different conditions, e.g: considering only one image per subject [16], improving robustness under changes of facial expression [4].

Classifying a Face

To classify an incoming face it has to be projected into the subspace described by the eigenfaces, the result of this operation is a weight vector with as many components as the basis has. The resulting vector is used to compare this face against the set of known faces, the most easy comparison can done using a simple euclidean distance metric. The following figure contains the algorithm.

1. Project the new face into the face space
2. Calculate the distance of this face to each of the faces known by the system
3. If the distance is below a threshold then the face is known otherwise it is unknown

The following set of equations explain how the projection is done into the face space and how the closest face in the set is selected, where w_k is the set of weight vectors from each of the faces in the training set, x is the weight vector of the input face and w_0 is the face in the training set closest to the input face.

$$w_k = u_i^T(\Gamma - \Psi) \tag{4.10}$$
$$x = u_i^T(x - \Psi) \tag{4.11}$$
$$w_0 = arg \min_{1 \leq i \leq M} ||x - w_k|| \tag{4.12}$$

Thus the smaller the distance to an already known face the more likely the input face belongs to the subject whose distance is smaller. This method can even be used to know if an object is a face, if the input image is close enough to the face space, then it can be considered as a face.

Eigenfaces problems

The algorithm explained is the original one proposed by Turk and Pentland, this algorithm is quite good at obtaining the most expressive subset of eigenfaces to form the basis of the new subspace, however it is not the most discriminating subset, which is due to the way the way the covariance matrix is build, it contains all the faces that belong not only to the same subject but also to different ones. The best possible basis is that in which we have a very low intrapersonal variation while the interpersonal variation is as large as possible. It happens that the eigenfaces with the largest eigenvalues are the most expressive ones, but retain a large component of the intrapersonal variation and the eigenfaces with lower eigenvalues have no intrapersonal variation, however they contain almost no discriminating power.

If a basis could be built that retains the most interpersonal variation while keeping the intrapersonal variation as low as possible we could have only one sample image per subject. In the next section an eigenface basis selection method with such properties is investigated.

4.2 One Image per Subject

Chapter 5

On-Chip Face Recognition System

5.1 Introduction to FPGA devices

Today electronics are dominated by digital electronics integrated circuits ranging from general purpose microchips (such as those found in computers) to application specific integrated circuits (ASIC) which can be found on most electronic devices, this circuits are targeted to specific applications (e.g. MPEG decoders, controllers, signal processors, etc.). Most of these circuits are build upon *standard cells* (gates, flip-flops and memory blocks) instead of at the transistor level. Using standard cells layout is optimal for most of applications and reduces the production costs and time a lot, since most of the cost of producing the integrated circuits are eliminated (the cost of producing extremely expensive chip masks is removed).

Field Programmable Gate Arrays (FPGA) are standard chips that can be programmed in the field. These devices contain standard cells (logic gates, flip-flops and memory blocks), the developer has to specify how to wire those elements together. Powerful tools have been build around FPGAs that enable developers to dramatically cut the production time:

- **Hardware Description Languages (HDL)** such as Verilog (primarily used in the United States) and VHDL (VHSIC Hardware Description Language) have also been developed to ease the developer work.
- **Simulators** that enable the debugging of the circuits before actually using the FPGA.
- **Synthesis Tools** that translate the hardware description into a bitstream targeted at each device, in this step the different cells in the chip are wired as the developer specified.

VHDL is the main description language used today, it has several levels each tailored to specific needs, while the language considered as a whole can be close to C or other low-level programming languages, there is only a much restricted subset that is actually synthesizable (i.e. translatable to hardware) by most tools, this subset is composed only by the most basic constructions of the VHDL language. Advanced and expensive tools are also available which bring new languages closer C (like Handel-C) into the hardware design world.

As expected FPGAs and these development tools have changed the way in which ASICs are developed next, the most important steps followed to develop a new system are described and figure 5.1 shows the work-flow.

- **Arquitecture partitioning** decide which modules will be developed in software and which ones will be designed in hardware.
- **Hardware Description Language** design for hardware modules.
- **Simulation** of the hardware description.

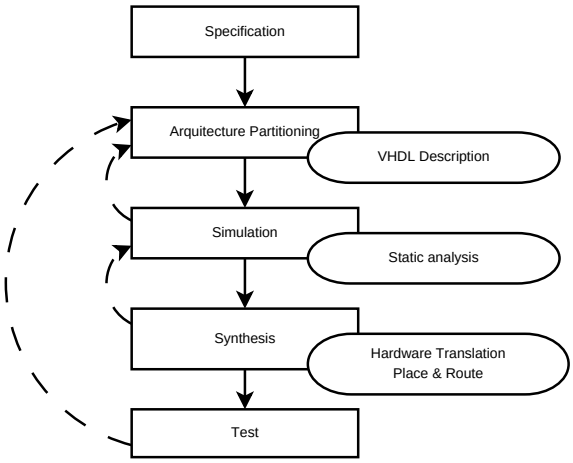


Figure 5.1: FPGA development workflow

- **Synthesis** the design is translated to a bitstream that the target hardware can understand, in this step the cells are wired (this is also called place & route).
- **Verification**, which involves the testing of the hardware.

During the last years FPGAs use in digital signal processing (DSP) has been growing due to their increasing power and the introduction of special cells geared towards DSP processing, currently FPGAs easily surpass general purpose DSPs and are starting to be used for application specific DSP. The main fields where FPGAs are used are: telecommunications industry (networking, telecom and DSP processing), consumer electronics (MPEG decoders and encoders), etc.

5.2 Face Detection

5.3 Face Normalization

5.4 Face Recognition

Bibliography

[1] M. Turk and A. Pentland. *Eigenfaces for Recognition.* In Journal of Cognitive Neuroscience 1991.

[2] Md. Al-Amin Bhuiyan, V. Ampornaramveth, S. Muto and H. Ueno. *Face Detection and Facial Feature Localization for Human-machine Interface.* NII Journal, (no.5), 2003.

[3] G. Shakhnarovich and B. Moghaddam. *Face Recognition in Subspaces.* Mitsubishi Electric Research Laboratories Technical Report TR2004-041. May 2004.

[4] S. Chen and B.C. Lovell. *Illumination and Expression Invariant Face Recognition with One Sample Image.* In proceedings of the International Conference on Pattern Recognition. August 2004.

[5] S. Chandrasekaran, B.S. Manjunath, Y.F. Wang, J. Winkeler and H. Zhang. *An eigenspace update algorithm for image analysis.* Graphical Models and Image Processing, vol.59, (no.5), Academic Press, pp.321-32, Sep 1997

[6] N. Muller, L. Magaia and B.M. Herbst. *Singular Value Decomposition, Eigenfaces, and 3D Reconstructions.* In SIAM Review, Vol. 46, No. 3. September 2004.

[7] P. Viola and M. Jones. *Robust Real-time Object Detection.* In International Journal of Computer Vision. 2001.

[8] A. Martínez and A.C. Kak. *PCA versus LDA.* In IEEE Transactions on Pattern Analysis and Machine Intelligence. February 2001.

[9] M.S. Bartlett, J.R. Movellan, T.J. Sejnowski. *Face recognition by independent component analysis.* In IEEE Transactions on Neural Networks. 2002.

[10] PN. Belhumeur, J.P. Hespanha and D.J. Kriegman. *Eigenfaces vs Fisherfaces: recognition using class specific linear projection.* IEEE Transactions on Pattern Analysis and Machine Intelligence. 1997.

[11] Y-S. Ryu and S-Y Oh. *Automatic extraction of eye and mouth fields from a face image using eigenfeatures and multilayer perceptrons.* Journal of Pattern Recognition. December 2001.

[12] J. Heo, B. Abidi, J. Paik and M. Abidi. *Face recognition: evaluation report for FaceIt Identification and Surveillance.* In Proceedings the International Conference on Quality Control by Artifical Intelligence, May 2003.

[13] F. Zuo and P.H.N de With. *Real-time Face Recognition for Smart Home Applications.* International Conference on Consumer Electronics. February 2005.

[14] G. Littlewort, M.S. Bartlett, I. Fasel, J. Chenu, T. Kanda, H.Ishiguro and J.R. Movellan.*Toweards social robots: Automatic evaluation of human-robot interaction by face detection and expression classification.* Advances in Neural Information Processing Systems. 2003.

[15] W. Zhao, R. Chellappa, P.J. Phillips and A. Rosenfeld. *Face Recognition: A Literature Survey.* In ACM Cumputing Surveys, December 2003.

[16] J. Wang, Y. Gu, K.N. Plataniotis and A.N. Venetsanopoulos. *Select Eigenfaces for face recognition with one training sample per subject*. In proceedings of the International Conference on Control, Automation, Robotics and Vision, December 2004.

[17] A. Pentland, T. Starner, N. Etcoff, N. Masoiu, O. Oliyide and M. Turk. *Experiments with eigenfaces*. Workshop of International Joint Conference on Artificial Intelligence, 1993.

Chapter 6

ToDO

- Incluir datos reales de pruebas
- Incluir imagenes de eigenfaces
- Incluir seccion en cada sitio que hable de las restricciones que impone la FPGA

Appendix A

Face Detection Matlab Algorithm

```
% Remove those pixels that are not on the skin
% histogram
for i=1:height
    for j=1:width

        h = h_cue (i,j) + 1;
        s = s_cue (i,j) + 1;

        if (histogram (h, s) > 0.10)
            im (i, j) = 1;
        else
            im (i, j) = 0;
        end

    end
end

% Remove small regions (noise)
se = strel ('disk', 3);
mask2 = imopen (mask, se);

% Dilate pixels to improve pixel connectivity
se = strel ('disk', 4);
mask2 = imdilate (mask2, se);

% Fill inside regions
mask2 = imfill (mask2, 'holes');

% Mark clusters of pixels
[mask3, numObjects] = bwlabel (mask2, 8);

grain = regionprops (mask3, 'all');


% Get the dimensions of each region found and test its
% size ratio and area
for i=1:numObjects
    bbox = grain(i).BoundingBox;
    area = grain (i).Area;
    ratio1 = bbox(4)/bbox(3);
```

```
    % Filter image regions whose aspect ratio is below
    % 0.75 or above 2.0
    if (ratio1 < 0.75 || ratio1 > 2.0)
        mask3 = face_filter_out (mask3, i);
    else
        % Only consider those cluster that take 19% or
        % more of the picture
        if ((100 * area / imsize) < 15.0 )
            mask3 = face_filter_out (mask3, i);
            if (numObjects == 1)
                disp ('Warn: filtering object, it is not big enough');
            end
        else
            face_box = bbox;
            orientation = grain(i).Orientation;
            g = grain(i);
        end
    end
end

% Close any holes in the mask
se = strel ('disk', 3);
mask3 = imfill (mask3, 'holes');
```

Appendix B

Feature Extraction Matlab Algorithm

```
% Load the image containing the face
base_dir = fullfile ('..', '.');
face = fullfile ('..', 'face.jpg');
x = imread (face);

xgray = double (rgb2gray (x));
[height, width] = size (xgray);

xbw = im2bw(xgray, 0.01);

%--------------------------------------
% Horizontal Projection
h_edge_map = edge (xgray, 'sobel', 'horizontal');

hproj = zeros(height,1);
for i=1:height
    for j=1:width
        hproj(i, 1) = hproj (i, 1) + h_edge_map(i,j);
    end
end

% Get the start of the face with more precision
%  *) Look for the hair in the region 0 - Height/4
%  *) Look for the mouth in the region Height/2 - Height
%
% [1] Y-S. Ryu and S-Y. Oh. Automatic extraction of eye and mouth fields from
% from a face image using eigenfeatures and multilayer perceptrons. Pattern
% Recognition 34.
t = round (height/6);
temp = hproj (1:t, 1);
[v, y_max] = max (temp); % y_max is the hair line

temp = hproj (height/2:height, 1);
[v, y_min] = max (temp) % y_min is the chin line
y_min = y_min+height/2;

%--------------------------------------
%% Vertical Projection
```

```
v_edge_map = edge (xgray, 'sobel', 'vertical');
vproj=zeros (width, 1);

for i=1:width
    for j=y_max:y_min
        vproj(i,1) = vproj(i,1) + v_edge_map(j,i);
    end
end

% Get the vertical limits of the face
%  *) Look for the left side in the region enclosed by 0 - width/2
%  *) Look for the right side in the region enclosed by width/2 - width
t = round (width/2);
temp = vproj(1:t, 1);
[v, x_left] = max (temp);

temp = vproj(t:width, 1);
[v, x_right] = max (temp);
x_right = x_right + t;

%---------- Extraction of eyes ----------
t1 = round(2/8*(y_min-y_max))
t2 = round(2/3*(y_min-y_max))

proj = zeros(height,1);

for i=t1:t2
    for j=x_left:x_right
        proj(i,1) = proj(i,1) + v_edge_map(i,j);
    end
end

[v, eye_line] = max(proj)

% We take 20 as the initial eye threshold
[v, eyeThreshold] = min(proj(eye_line-20:eye_line,1));
eyeThreshold = 20-eyeThreshold;

eye_max = eye_line - eyeThreshold;
eye_min = eye_line + eyeThreshold;

proj = zeros(width,1);

for i=x_left:x_right
    for j=eye_max:eye_min
        proj(i,1) = proj(i,1) + h_edge_map(j,i);
    end
end

t=(x_right-x_left)/2+x_left;

[v, indx_right]=max(proj(t:x_right,1));
indx_right=indx_right+t;
[v, indx_left ]=max(proj(x_left:t,1));
indx_left=indx_left+x_left;
```

Appendix C

Matlab Functions Used

B
bwlabel

E
edge

G
graythresh

I
im2bw
imcrop
imdilate
imfill
imopen
imread
imshow
imwrite

R
regionprops
rgb2gray
rgb2hsv

S
strel