DESIGN AND IMPLEMENTATION ON A FPGA OF A FACIAL RECOGNITION SYSTEM USING "EIGEN FACES"

Sergio Barrios del Villar

Universidad de Guadalajara, Departamento de Electrónica, CUCEI

sergio.barrios@alumnos.udg.mx

Juan José Raygoza Panduro

Universidad de Guadalajara, Departamento de Electrónica, CUCEI

juan.raygoza@cucei.udg.mx

Alejandra Salamanca Chavarín

Universidad de Guadalajara, Departamento de Electrónica, CUCEI

salecita_ale@hotmail.com

Edwin Christian Becerra Álvarez

Universidad de Guadalajara, Departamento de Electrónica, CUCEI

edwincbecerra@gmail.com

Susana Ortega Cisneros

Centro de Investigación y de Estudios Avanzados del I.P.N., CINVESTAV, Unidad Guadalajara

susana.ortega@gdl.cinvestav.mx

Abstract

Automated Facial recognition is a very complex problem due to the many factors that affect the way an image of a person's face looks. Most of these have no relation to the actual identity of the person. The algorithms used to solve this issue can take advantage of a high level of parallelism and the applications require real time processing. For these reasons, an implementation on hardware is very convenient. In this article, such implementation is presented using a Xilinx Virtex 6 FPGA using one of the most common algorithms, called Eigen Faces.

Index Terms: Eigen Faces, facial recognition, FPGA, PCA.

1. Introduction

The problem with facial recognition consists in the following: Given the image of a face and a database of known individuals, the system must determine the identity of the person. It has been seen [1] that even babies from 1 to 3 days of age are able to distinguish between known faces; however, the exact method in which the human brain performs this task is unknown. Automated facial recognition consists of extracting significant characteristics from the image of a face, changing it to a useful way of representation, and classifying it as belonging to a person. This procedure is very useful, since other authentication systems depend on the individual performing a task such as remembering a password, carrying an object (key, token), or in the case of biometric systems placing a finger on or an eye in front of a special purpose sensor. On the other hand, facial recognition can be done passively with cameras, which have a low cost in comparison with other sensors, and without the individual being aware of the process.

Successful facial recognition is quite complex, since the faces of several people can be similar with only subtle differences. In addition, a human face is a flexible object; it changes with age, expression, and the use of objects such as glasses and cosmetics. The way an image of a face is perceived changes due to the position of the observer and the illumination of the environment. Most facial recognition systems work properly under ideal circumstances, the system presented in [2] achieved better results than humans. But their performance falls significantly in an environment where the factors that affect the variability of faces are not controlled.

2. Classification of facial recognition methods

Jafri and Arabnia [3] categorize facial recognition methods in the following manner.

Based on Geometric Characteristics

These methods consist of first processing the input image in order to identify and measure distinctive facial characteristics such as the eyes, mouth, nose, and

other features. Then, they calculate geometric relations between these points (for example the distance between the eyes), reducing the image to a vector of geometric characteristics. Finally, they use standard pattern detection techniques to identify the face in the database that shares those features. Since the points are extracted before analysis, these kinds of methodologies are robust with respect to position changes. The greatest disadvantage of these methods is the difficulty of automatically extracting the facial characteristics. Leung, Chang and Yu Li [4] implemented a design which performs the extraction step on a FPGA and communicates the results to a Personal Computer for the recognition process.

Statistical

Consider that an image of a face is represented as a two dimensional matrix with light intensity values. Then, a simple facial recognition method is to perform the correlation operation between the input image and the images in the database. This approach is computationally expensive and sensitive to variations in position, illumination, noise, and a cluttered background. These problems arise due to performing classification on a highly dimensional space. A better performance can be achieved by using statistical techniques to represent images in a space with a smaller dimensionality.

AI (Artificial Intelligence)

AI methodologies use tools such as neural networks and Machine Learning techniques to perform the classification. These are often combined with Statistical methods to reduce dimensionality before performing the actual recognition process due to their computational complexity. The solution proposed by Mahale in [5] used this approach.

3. Statistical tools: PCA

Principal Component Analysis (PCA) is one of the most widely used statistical methods for efficiently representing images of faces. It takes advantage of the fact that any face can be represented economically in the coordinate space of the

Eigen Pictures and that any face can be reconstructed partially by using a small collection of Eigen pictures and the corresponding projections, or coefficients, along each one. Using this methodology, Eigen faces can be constructed that correspond to the Eigen Vectors associated with the Eigen values of the covariance matrix of the known faces. It is possible to recognize a face by comparing the projections along the Eigen faces with the projections of the images in the database. The Eigen faces define a characteristic space, drastically reducing the dimensionality of the original space, and the identification of the face is performed in this new space. This system is robust with respect to illumination changes, but its performance decays with changes in size. This technique is used in [6] to detect facial expressions.

The multiple observer technique consists of having a database not only with an image of a known face, but also with several images of it from different angles. PCA performs properly with a single image per person but fails with more, since the change of an image due to position and illumination is bigger than the change caused by the identity of the person. This causes PCA to consider unwanted variations. This problem can be solved by using Fisher's Linear Discriminant Analysis (LDA), which maximizes the variation between classes and between images of the same class. This method, called Fisher Faces, is better at handling simultaneous changes in light and expression. With small collections of training images, PCA can outperform LDA, according to [7].

Another approach uses difference images. These are defined as the subtraction between the intensity values of corresponding pixels from two images. Two classes of difference images are proposed: Intrapersonal, the difference between images of the same person, and extra personal, the difference between images of different persons. It is assumed that difference images originate from discrete Gaussian distributions from the space of all the difference images. The probability that a difference image belongs to the interpersonal class can be calculated using the Bayes' theorem. If the probability of a difference image being of the interpersonal class is higher than the probability of it being of the extra personal class, then it is considered that the two images from which the difference image was calculated

belong to the same person. Thus, the problem of facial recognition is reduced from a classification of grade m (m being the number of people in the database) to a binary classification problem.

4. Eigen faces algorithm

Facial recognition consists of two stages: preparing a set of images of known individuals and, given an input face, recognizing if it belongs to one of the known individuals. The advantages of this algorithm are explored in [8], [9] and [10] by implementing it on MATLAB. The steps using the Eigen Faces algorithm to prepare the images are the following:

A database S is gathered using *M* images with the faces of known persons (1). An average image is obtained by calculating the average of the intensity values of each pixel in the set of images (2).

$$S = \{I_1, I_2, I_3, \dots, I_M\} \tag{1}$$

$$P = \frac{1}{M}\sum_{n=1}^{M} I_n \tag{2}$$

A new set of images (4) is gathered, using new images with the distinctive characteristics of each, by subtracting the average image (3).

$$J_n = I_n - P \tag{3}$$

$$A = \{J_1, J_2, J_3, \dots, J_M\} \tag{4}$$

A covariance matrix is obtained (5) using the set of images and is transposed. The resulting matrix is symmetrical, which is useful for calculating the Eigen vectors u_k and Eigen values λ_k, defined as complying with the relationship in (6).

$$C = AA^T \tag{5}$$

$$\lambda_k = \frac{1}{M}\sum_{n=1}^{M} (u_k^T J_n)^2 \tag{6}$$

Finally, the weights v_{lk} are calculated using (7). These represent the projections of the images over the Eigen Faces space.

$$u_l = \sum_{k=1}^{M} v_{lk} J_k \qquad l = 1, \dots, M \tag{7}$$

Now that the faces are represented in the Eigen vector space, it is possible to classify an input image. Eigen vectors are sometimes called Eigen faces in this application, because they resemble faces when interpreted as images. First, the weights of the input image over the new space are obtained similarly to how the weights of the images in the database were processed. First, the mean image is subtracted and the result is multiplied by the Eigen faces (8). The vector with the weights is called Ω (9).

$$w_k = u_k^T (I - P) \tag{8}$$

$$\Omega^T = [w_1, w_2, \dots, w_M] \tag{9}$$

The vector Ω, which represents the input image over the Eigen faces space, can be compared to the vectors with the projections of the images in the database. One way of doing this is by subtracting the vector of the input from each vector of the database. The result will have a small magnitude (or Euclidean distance) if the images come from the same face. The Euclidean distance is defined as the square root of the sum of the square power of each weight (10), but since it is only important to compare how big it is and not the exact magnitude, then it is possible to use the square power of the Euclidean distance, thus avoiding the need to calculate the square root; instead, the calculation becomes a dot product between the vector and itself (11).

$$\|p\| = \sqrt{p_1^2 + p_2^2 + \cdots + p_n^2} \tag{10}$$

$$\|p\|^2 = p.p \tag{11}$$

5. Implementation on FPGA

The FPGA used was the Xilinx Virtex-6 model VC6VCX240T. Values calculated on MATLAB are declared as constants in VHDL, so they synthetize as connections to VCC or ground. The image is stored in a special distributed Random Access Memory (RAM) made for matrices, which receives two addresses, one for the

position of the pixel in the x axis and one for the y axis. This memory is also different because it outputs all the contents of the memory in parallel, so it can be processed in parallel as well, figure 1.

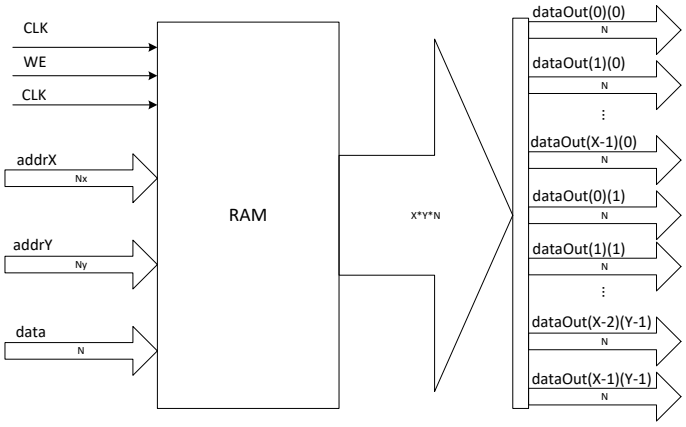

Figure 1 Distributed RAM Memory.

The first step in facial recognition is to subtract the mean image from the input image; this is done via subtraction modules. These are instantiated once per pixel, as seen in figure 2.

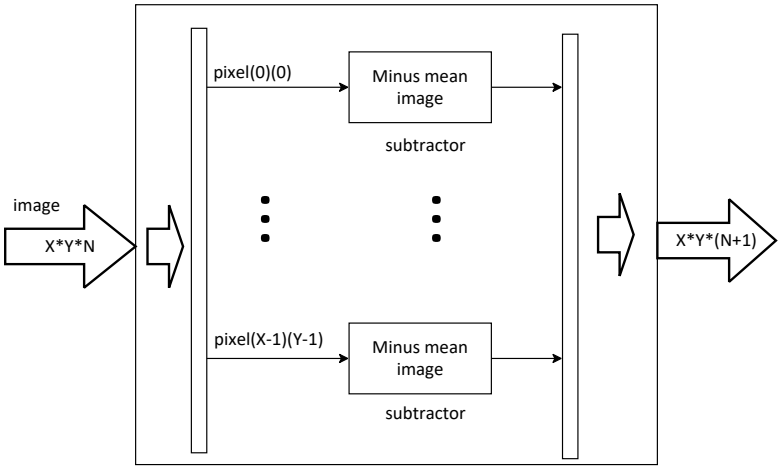

Figure 2 Mean image subtraction module.

The second step is to perform a dot product of the resulting image by each of the Eigen Faces. The dot product block consists of a pixel by pixel parallel multiplication followed by the sequential sum of the results, as seen in figure 3.

One dot product module is used for each Eigen Face (figure 4). For a very small database, there is one for each image in the database. After the dot product, the result is a representation of the image in the new space.

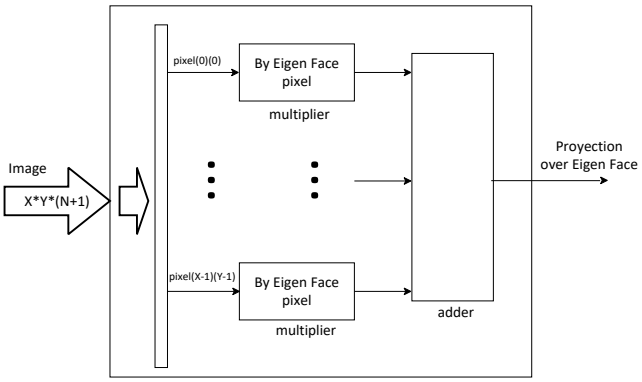

Figure 3. Dot product module.

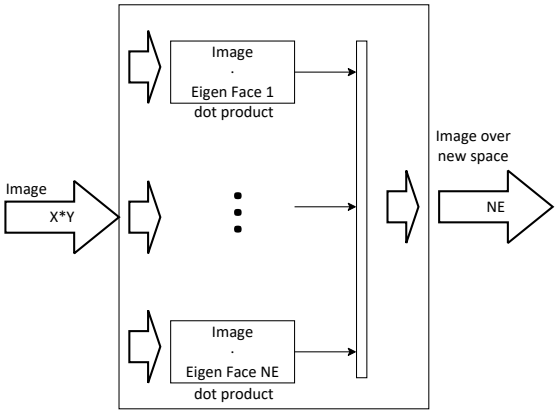

Figure 4 Module to obtain the new representation.

The last phase is to compare it with the representation of the images of the database. These modules are implemented once for every image in the database. The first step of the comparison is to subtract the weights of the image in the database from the weights of the input image, as seen in figure 5.

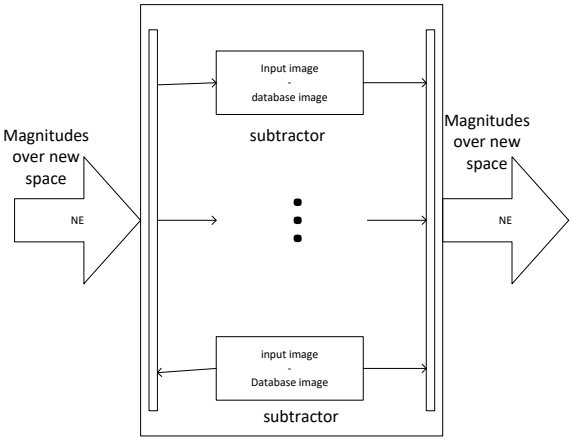

Figure 5 Module to subtract the weights.

The result indicates how different the input face is from that in the database. A value proportional to the magnitude of this vector is obtained using a dot product module (figure 3) with a bigger word length but a much smaller number of elements—one per eigen face instead of one per pixel. As mentioned in the previous section, the dot product operation is performed between the vector and itself. It is important to avoid overflow errors by expanding the word length at each step.

6. Results

Two experiments were used to verify that the system works correctly and to test if the amount of elements needed to synthetize the system are feasible. The first experiment is a trivial case. It has images of 2 by 2 pixels and a word length of 8 bits. The database has only two images. This experiment allows the results to be easily verified, since synthetizing or calculating the result does not require much time. An image with only 4 pixels cannot represent a face, but the method works well with other patterns. The images in the database can be seen in figure 6, the mean image appears in figure 7, and figure 8 shows an image outside the database to verify that the system recognizes it as unknown.

Figure 6 Image database for experiment one.

Figure 7 Mean image for experiment one.

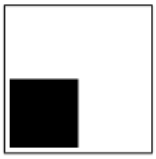

Figure 8 Image external to the database in experiment one.

The Eigen vectors can also be interpreted as images and hold the distinctive characteristics of the images in the database. In this case, the Eigen pictures resemble the images in the database minus the mean image, since all the images in the database are completely different. Figure 9 shows these images. The utilization of Look-Up Tables (LUT) and Input Output Blocks (IOB) in the FPGA (Virtex 6) for this experiment is shown in table 1.

Figure 9 Eigen pictures of experiment one.

Table 1 FPGA utilization for experiment one.

Slice Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	277	150,720	1%
Number of bonded IOBs	15	400	3%

The second experiment is more useful, since it has images of 32 by 32 pixels and a word length of 8 bits. Images of this size, although small, are capable of representing faces, and since face recognition systems are mainly about reducing the dimensionality, it is better to start with images with lower resolution (and therefore lower dimensionality) to begin with. A word length of 8 bits is also a common standard for grayscale images, which are preferred, since color has little information about the identity of a person.

The images used for this experiment are drawings of faces, guaranteeing that all of them will have the same position, expression, and illumination. These were taken from [11]. Since these are the ideal conditions, the system is able to correctly classify the known faces shown in figure 10.

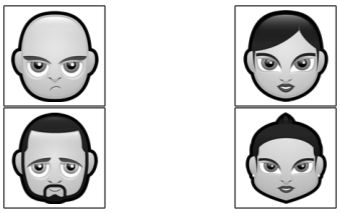

Figure 10 Image database for experiment two.

The mean image appears in figure 11. As with the previous experiment, another image outside the database was used to verify that it is not recognized as any of the known faces. This is shown in figure 12.

Figure 11 Mean image for experiment two.

Figure 12 Image external to the database in experiment two.

This time, the Eigen faces represent the principal components of the faces in the databases; these can be seen in figure 13.

Figure 13 Eigen faces of experiment two.

The amount of components needed to synthetize the system increases greatly with the size of the image. The 32 by 32 images used result in a system small enough to fit in the proposed Virtex 6, and their size is adequate for practical face recognition uses. However, a larger image will require a bigger FPGA. The exact utilization percentages can be seen in table 2.

Table 2. FPGA utilization for experiment two.

Slice Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	104,093	150,720	69%
Number of bonded IOBs	25	400	6%

7. Conclusion

An FPGA implementation is very useful for facial recognition, especially if real time processing is a requirement, because many computations can be performed in parallel. In order to take advantage of the parallelism, it is necessary for all the values of the pixels in the input image to be available at the same time. This can be a problem, since an image of 32 by 32 pixels with 8 bit intensity levels has 8192 bit values, which far exceeds the amount of input output blocks available. However, the implemented distributed RAM solves this issue and allows for larger sizes without requiring a large amount of IOBs, since the pixel values are loaded one at a time, while the reading is done all at once, so the processing begins while the image is still loading.

The system produced the desired result in the ideal conditions tested in the aforementioned experiments. In order to use this in a real life project, it is necessary to load images into the system in gray scale and a fixed size, so a program or an additional hardware module would be required to read images from a camera or another input device and convert it to the required format with the face centered. Such systems are presented by Gorde (2016) [12] and Kim (2015) [13]. The proposed work uses only 72% of the resources needed by Kim's work but it includes the preprocessing stage.

8. Bibliography and References

[1] C. Turati, V. Macchi, F.S. Cassia, I. Leo, "Newborns face recognition: Role of inner and outer facial features". Child Development. Vol. 77. No. 2. 2006. Pp. 297–311.

[2] C. Lu, X. Tang, "Surpassing Human-Level Face Verification Performance on LFW with GaussianFace". In arXiv, Cornell University. 16 Jun 2014.

[3] R. Jafri, H. Arabnia, “A Survey of Face Recognition Techniques”. Journal of Information Processing Systems. Vol. 5. No. 2. June 2009.

[4] H. Leung, L. Cheng, X. Yu Li, "A FPGA implementation of facial feature extraction". Journal of Real-Time Image Processing. Vol. 10. 2015. Pp. 135-149.

[5] G. Mahale, H. Mahale, A. Goel, S. K. Nandy, S. Bhattacharya, R. Narayan, "Hardware Solution for Real-Time Face Recognition". 28th International Conference on VLSI Design, Bangalore. 2015. Pp. 81-86.

[6] S. Agrawal, P. Khatri, "Facial Expression Detection Techniques: Based on Viola and Jones Algorithm and Principal Component Analysis". Fifth International Conference on Advanced Computing & Communication Technologies, Haryana. 2015. Pp. 108-112.

[7] A. Martinez, A. Kak, “PCA versus LDA”. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 23. No.2. 2011. Pp. 228-233.

[8] P. Singh, A. Sharma, “Face Recognition Using Principal Component Analysis in MATLAB”. International Journal of Scientific Research in Computer Science and Engineering. Vol. 3. Issue 1. 28 February 2015.

[9] M. J. Pemeena, K. Murugesan, S. R. Inabathini, G. K. Rajini, Dinesh, A. Kumar, “Recognizing the 2D Face Using Eigen Faces By PCA”. Journal of Theorical and Applied Information Technology. Vol. 82. No. 1. 10 December 2015.

[10] R. S. Negi, R. Garg, “Face Recognition Using Hausdroff Distance as a Matching Algorithm”. International Journal of Scientific Research. Vol. 4. Issue 9. September 2015.

[11] Hopstarter, “Face Avatars Icons”. August 2011. via IconArchive, Creative Commons Attribution.

[12] H. Kim, M. Lee, Y. Baek, S. Kim, B. Koo, J. Lee, “Fully Hardwired Illumination Invariant Face Recognition Engine and FPGA Implementation”. 2015 IEEE International Symposium on Consumer Electronics. January 2015.

[13] S. H. Gorde, M. Kumar. P. Balramudu, “An FPGA based Face Recognition System using Gabor and Local Binary Pattern”. International Journal of Advanced Research in Electronics and Communication Engineering. Vol. 5. Issue 1. January 2016.