Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Industriales

Departamento de Automática, Ingeniería Electrónica e Informática Industrial

Master on Industrial Electronics

CPU/GPGPU/HW comparison of an Eigenfaces face recognition system

Author: Julio Camarero Mateo

Advisor: Eduardo de la Torre Arnanz

March 2014

Master Thesis

Quiero agradecer a todos aquellos que me habéis acompañado durante este tiempo, porque este trabajo ha sido posible por vosotros y es tan vuestro como mío.

Este trabajo está dedicado a mi madre, que me enseña cada día a luchar por lo que uno quiere en la vida. Sigue haciéndolo y sigue tan valiente como hasta ahora.

A mi padre, mi hermana Elena, Laura, Rubén y mis tíos, por el buen equipo que formamos y porque sin vosotros no podría haber terminado este último paso.

A Edu, Juan, Andrés y Mora, por haberme guiado sabiamente durante todo este tiempo.

A Laura de nuevo, porque no sólo me renuevas la alegría cada día, sino que haces más pequeñas mis cargas.

A los compañeros del máster Ángel, Alfonso, Filip, Dave y los profesores del CEI, que habéis hecho que este año sea divertido a la vez que aprendía con vosotros.

A los compañeros de comedor Aledo, Giuliano, Yann, Mavi, Gabriel, y Fermín, que siempre enriquecéis la comida con inquietantes divagaciones intelectuales.

A los adictos al café, Portilla, Víctor, Chema, Dani Martel y los ya citados, porque pocos cafés me tomaré tan interesantes, a la par que arriesgados, como los del almacén.

A Blanca, Fede, Santiago y todos a los que en algún momento me habéis sufrido, tenéis un gran futuro por delante, ánimo.

A Álvaro, Mónica, David, Peña, Jorge, Airán y todos los demás compañeros del CEI, ha sido un honor el tiempo pasado con vosotros.

A todos mis amigos de siempre, que me encanta que sean como son y que son los mejores amigos del mundo, no les cambiaría por nada.

A todos aquellos que me dejo en el tintero… ¡MUCHAS GRACIAS!

Un nuevo horizonte se abre lleno de posibilidades.

Table of Contents

Chapter 1. INTRODUCTION

During the last fifteen years, many electronic devices have been provided with powerful processing units, while at the same time they included power consumption reduction techniques as complex as having variable clock frequency or the ability to enable and disable different blocks of resources depending on the demanded processing load. This fact opens a new outlook in the range of applications one device is able to manage, due to the improvement in efficiency while handling more data concurrently. In order to illustrate this assertion, it is sufficient to think about portable devices such as notebooks [1], with quad cores and GPGPUs inside, or even novel smartphones that can have up to eight cores and a GPU in a single chip [2], [3].

Taking into account these new characteristics, it could be safe to say that it has emerged thereby the possibility of accomplishing very demanding tasks that were previously unfeasible. Due to this evolution, it is possible to find multimedia applications embedded in surveillance systems, notebooks or even smartphones. One example of this is the face detection applications in the software of digital cameras.

In this Master Thesis one face recognition system is studied in another type of smart devices: a Wireless Sensor Networks (WSN) platform. It will be implemented in hardware in an FPGA, in software in an embedded microprocessor, and compared with the results of two C++ and CUDA (Compute Unified Device Architecture) programs in a laptop and in a desktop computer. The advantages and disadvantages in terms of performance and power consumption of every solution will be seen and compared.

I. Security and Image Processing

Nowadays, security is a major concern not only in public spaces, such as airports or shopping centres, but also in different scopes of applicability like surveillance of private properties or pass-permission for an enclosure. On top of that is the trend for growing countries towards adopting the latest models of development in cities [4]. The generalization of the Smart City goal, together with the Internet of Things, comes up with the utilization of distributed systems in almost every place where humans can carry out any economic or social activity. Common traffic lights are

endowed with new functionalities, temperature sensors, cameras and even communication capabilities in order to send the information for management and future use. This research work is focused in the area of security using video cameras, and more specifically in the analysis and comparison of alternatives for processing images with different platforms.

With the purpose of supporting the network, some authors [5] have proposed to take advantage of cooperative users and use portable devices as anonymous routers or even distributed processors of the Smart City. While this can be helpful in terms of saving resources, it could also become quite risky because of the data exposure endured during the process [6]. Nevertheless, Wireless Sensor Nodes are good candidates at this scenario and security is a well-known issue studied by many authors since the popularization of these devices [7], [8]. Precisely, one of the main contributions achieved in this Master Thesis is developed on the WSN designed at Centre of Industrial Electronics (CEI) called HiReCookie, which is described in Chapter 2.IV.

In the field of subjects' identification, biometric modalities such as fingerprint acquisition, iris analysis or ID card verification can be used. However, the face recognition technique is not only nonintrusive and natural for humans, but also can be captured at a distance and in an undercover manner. According to Heitmeyer et al. [9], facial features achieved the best results when identifying passengers in an airport or station. In Figure 1 an example of this scenario is shown.

Figure 1 Cooperative identification system in an airport or company.

But image processing is usually considered as a very time and resource consuming task due to the large amount of managed data. In order to cope with this task, different approaches have been used, from commercial software products sold by companies, thought to be running in personal computers in communication with

IP cameras [10], to ASIC solutions. More information related with this approach will be given in Chapter 2.II.

It is worth stressing that, on the one hand, image processing is a huge field that has been broadly exploited with usual microprocessors in software, DSPs and even in the last years using novel techniques of processing images with FPGAs [11] and parallel computing with GPUs. On the other hand, Visual Wireless Sensor Networks have acquired enough computing power to deal with multimedia applications, and there are, for instance, studies related with the optimization of power consumption [12] or data encoding [13].

Nevertheless, what is actually an original contribution in this Master Thesis is the design of an image processing block in hardware for an FPGA-based WSN node. Due to the high performance characteristics of this WSN platform, a brief overview of possible applications is shown in the next section.

II. WSN in High Performance Applications

WSNs have been regarded as smart sensors based on low power nodes, with microprocessors of limited data processing capabilities. Moreover, they are designed in order to cope with scenarios where human intervention is not possible or so frequent, as well as limited maintenance and low profile in the amount of measurements and communications. In the end, power consumption is crucial because the operability depends on a battery, which obviously provides a limited amount of energy.

However, the tendency is changing towards having more powerful processing units, mainly due to new requirements in terms of security, speed or volume of data demanded. Fortunately, it has been demonstrated [14] that it is feasible to save energy with FPGA-based WSNs, by computing information as fast as possible and then switching off, or sending all the expendable modules in the platform to deep power saving mode.

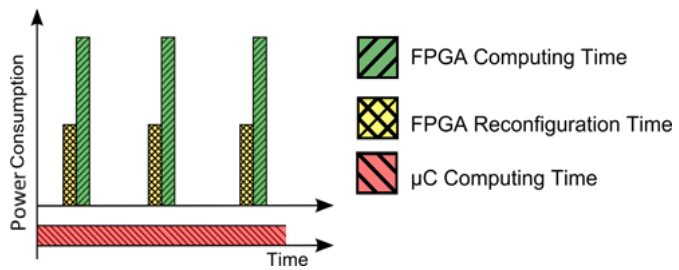


Figure 2 Power consumption profile comparison between a microprocessor and an FPGA.

These techniques have twofold benefits; they not only make possible to cover a wide variety of new applications, but also allow energy to be managed at the same time in a more efficient manner. Additionally, it could be interesting to take advantage of already proposed schemes of low-power image compression in WSNs [15]. Nevertheless as it will be seen, in the RUNNER project any image exchange is avoided, returning minimal information after a request.

One example of feasibility study for detecting humans with a WSN, in a robust and efficient manner was carried out by H. Fu et al. [16]. Although the research work discusses different algorithms for detecting humans, it seems to be proven only in a personal computer, while the purpose of this Master Thesis would be to comprise a real implementation in a WSN also tested in a real environment.

In [17], authors deployed a complete surveillance system based on MICA2 nodes, which have as high performance microprocessor an Atmega 128L. In this work, motion and magnet sensors are used instead of cameras, and the main contribution is made in the communication process.

Environmental care and even industrial monitoring could make the most of high performance solutions with hybrid platforms like the ones proposed by Portilla [18] or Latha [19]. When certain sensor, such as light, proximity or pressure detect a risky situation, it is possible to have extra-processing capacity by activating a coprocessor. Some possible tasks can start being encrypted and become more secure during communication, start computing multimedia data or achieve the results faster or in a more complex manner.

Accordingly with this framework, in Centre of Industrial Electronics was designed a novel WSN node architecture called HiReCookie. It will be the hardware platform for this Master Thesis and more details will be given in Chapter 2.IV.

As it has been shown, in this research work it is possible to take advantage of the high processing capabilities and energy saving strategies of FPGAs, in order to achieve the best efficiency for a WSN in the process of computing images. In the next section the motivation and the framework of the project will be exposed.

III. *Motivation and Aim of the Project*

The motivation of this research work is dual: on the one hand, it pursuits the realization of a fully functional face recognition application in a high performance WSN platform; on the other hand, it encompasses the study of different alternatives for achieving that and its features and performance.

The main goals of this Master Thesis are listed below:

- Develop a first approach of a face identification algorithm and look into some other possibilities and extensions.
- Apply the acquired knowledge on behalf of designing the first software version of the algorithm in the HiReCookie platform.
- Create a hardware reconfigurable peripheral; this block will be used in the RUNNER project.
- Verify with automatic tools the results in every solution, comparing them with a Golden Reference.
- Explore how the FastCUDA approach, which facilitates the design of such HW accelerators, would make easier to achieve future solutions for similar cases.

Due to the direct implication of this Master Thesis regarding RUNNER and the proximity to FastCUDA, both projects are introduced in the following section.

IV. *Framework of the Project*

The results of this research work have supposed a great progress for two European projects, RUNNER and FastCUDA. The participation of the CEI in RUNNER started with the final degree project of the same author of this Master Thesis [20], and it was the initial motivation of the present work. Furthermore, due to the collaboration with colleagues on the second project, it became possible to enrich the range covered by the study, by taking into account the FastCUDA strategy. Hence, brief introductions for each one of the mentioned projects are shown below.

IV.1 The RUNNER Project

RUNNER aims to provide an innovative infrastructure, to be exploited in the creation of highly autonomous robots. It utilises high-end reconfigurable devices, in order to allow extremely high performance and power-efficient processing, when implementing 3D sensing/matching schemes.

Going into more detail, in the general definition of a robot, Sense-Plan-Act, the sense-part requires a highly efficient and detailed perception. The most important challenges for autonomous and effective robots in automatic environment are the capacity of seeing, understanding and interacting with a three-dimensional real world. Since it is not a coincidence that the visual cortex of the human brain is the largest sensory part of the brain, in order to obtain a high performance 3D-perception system for a robot, a significant amount of parallel computations are required. Given the availability of image sensors with very high performance at a

low cost, the challenge is now to create a low cost, high performance Artificial Visual Cortex. This vision system will perform all the required calculus of the environment, keeping the central processor free of any visual computation, with some simple messages for the indispensable interchange of information.

The problem domain of autonomous robotics comprises two major tasks with different key aspects. The first one covers exploration issues while fast and accurate matching algorithms are taking part; the second task includes the navigation in known and unknown terrains, where the real time response takes precedence over the quality of the numbers.

Detecting and recognizing an object from a video input turns out to be an issue that requires great precision. The problem stems from the fact that a single object can be viewed from an infinite number of ways. By rotating, obscuring, or scaling a single object, one can create multiple representations of an object - which makes the problem of matching the object to a database of objects very difficult. Depending on the organization of the database, the problem expands either linearly or exponentially whenever there are numerous objects that should be identified simultaneously.

The Real-time 3D computation of the scene in the moving direction of a robot is required to ensure obstacle avoidance, whereas the precision is secondary. One possible method of navigation is to use stereo-vision algorithms in order to know how far or close objects are located in the environment. Such a challenge could only be achieved by the utilization of something else than one camera; it could be one camera and a radar or sonar system, a 3D vision camera, or as it is proposed in RUNNER, two cameras, in which the distance and the focus angle are controllable.

RUNNER aims to create a framework in which highly autonomous Robots with much better perception than the existing solutions will be based. This innovative infrastructure will utilize state-of-the-art reconfigurable devices (FPGAs), which are well known for allowing extremely higher performance and power-efficient processing when implementing data manipulation methods such as 3D sensing/matching schemes as well as template and feature-based object recognition algorithms. They will be partially reconfigured on run-time either remotely or autonomously. In order to accomplish that, the robotics management scheme will provide the mechanisms to alter the configuration of the robots' devices to support different vision and/or object recognition schemes and/or different functionality. In the general case, the robot itself will be able to monitor the environment and change its configuration providing an extremely flexible environment. By altering the functions executed on the reconfigurable device in real-time, given the running operational conditions, the features versus performance trade-off will be very close to optimum at any point in time.

In general, the system developed within RUNNER takes advantage of the much higher processing power offered by the reconfigurable devices, when compared with the general purpose CPUs, and the partial real-time reconfiguration feature of the state-of-the-art reconfigurable devices, that will allow it to alter the processing tasks according to the environment of the robot. Moreover, the reconfiguration feature of the FPGAs can also be used for decreasing the overall power consumption, as well as for increasing the fault tolerance. Dynamic reconfiguration in the former case can be done while the remaining processes of the chip continue to operate. It would be possible to change the algorithm, which is executed on the chip, among several candidates, without having the need for a large part to simultaneously run all different algorithms. What is more, even in the presence of permanent errors, a design can be placed on a different part of the FPGA while the rest continues to operate, achieving the desired fault tolerance as a result.

More precisely, the role of the Centre of Industrial Electronics in this work will be to design and implement modules supporting very high rates by taking full advantage of the high processing power provided by the high-end FPGAs. The first one is the face recognition algorithm called Eigenfaces algorithm, described in this Master Thesis, while the second one is the Stereo Matching Algorithm implemented by Federico Pérez in his final degree project. This work will be tested in the autonomous robot Nao – shown in Figure 3.

Figure 3 Robot Nao, in charge of testing RUNNER results.

The ultimate objective of RUNNER is to deliver a reconfigurable prototype with excessive cross-domain applicability. In RUNNER, we believe that in a few years there would be millions of robots in various application areas that will all navigate in an autonomous manner based on 3D video capture; such robots can be efficiently

and inexpensively built based on the provided innovative highly flexible infrastructure.

IV.2 The FastCUDA Tool Flow

The increased needs of today's market are causing an exponential growth of the efforts required by hardware designers and software developers. Europe is at the leading edge in wireless, multimedia, telecom and automotive electronic systems. However, as global market competition is increasing and as new generations of intelligent and high performance digital products are continuously required, the industrial landscape is changing dramatically. The new generations of digital products feature ever increasingly complex functionalities and design constraints, while their market window becomes shorter.

Due to increasing demands of the end users, companies need to accelerate their time-to-market and improve the profitability of new products. Ensuring profitability and sustaining competitiveness can only be achieved through high quality, high performance, fast design and implementation and cheap products. Nevertheless, the designers encounter significant difficulties in delivering competitive products to the market, since traditional design methods cannot satisfy, at the same time, issues such as short time-to-market, low cost, low power designs and a complex, reliable, high performance.

ASICs and FPGAs are usually very convenient for real time systems, due to their higher computation capabilities, and generally turn out to be more efficient, but a product developed on them requires more design time and it is usually needed to invest more resources and money on it; whereas CPUs and GPGPUs have easier implementations but consume more power.

Ideally, it would be desirable to have a new methodology with the best characteristics of each one but without any drawback of any of them. The purpose of FastCUDA is nearly that ambitious: to create an innovative embedded system design flow that will take advantage of the flexibility, low cost, the ease development and testing of software solutions, while having the performance, power characteristics, efficiency and acceleration of hardware implementations. Even more, it is desirable to avoid Intellectual Properties (IP) and private resources in order to achieve that, choosing when possible open-source ongoing efforts, as well as enabling an easier transition from research results to industrial exploitation [21].

The results of this project will be available online in an open-source repository for public utilization [22], and the input language for the tool will be CUDA.

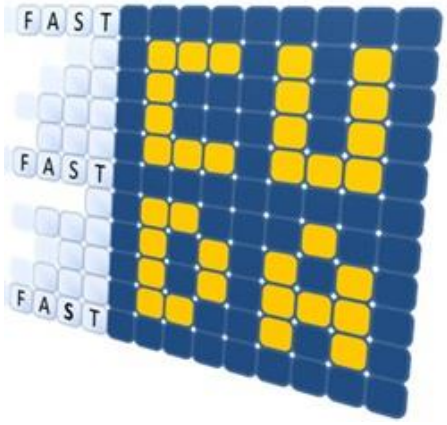


Figure 4 FastCUDA logotype.

CUDA is a parallel computing architecture developed by NVIDIA in order to execute general purpose computation algorithms on GPGPUs. The primary parallel construct in CUDA is a data-parallel, Single Process, Multiple Data (SPMD) kernel function. A kernel function invocation explicitly creates many CUDA threads. The threads are organized into multidimensional arrays that can synchronize and quickly share data, called thread blocks. The CUDA programming model allows the programmer to take advantage of the massive parallel computing power of a graphics processor in order to perform general purpose computation. CUDA benefits from the hundreds of ALUs inside a graphics processor and use massive parallel interfaces in order to connect with its memory. Applying the CUDA programming model on the FPGA technology seems to be very promising since the FPGA fabric can both support hundreds of ALUs and provide massive parallel internal memory bandwidth. In parallel, FPGAs can offer better performance and power characteristics than GPUs.

Additionally, the platform will be relatively easy to use through a graphical user interface (GUI) developed in order to gain wide acceptability by the embedded design community. Especially, as the tool targets are the group of designers programming in a high-level and, in the meantime, it is critical to speed-up their design time, the factor of having that specific tool operating in a user's friendly environment is of major importance. In fact, this will play an important role to the wide adoption of the tool.

The FastCUDA project opens new market channels for the SME participants and will allow them to expand their market and sell their products and services worldwide; this wide exploitation of the end-product will be heavily facilitated by the fact that it will be distributed on an open-source manner and such products, as it

has been extensively proved in the past, immediately gain significant attention by the embedded systems community.

V. Document Structure

This Master Thesis follows the subsequent organization: Chapter 2 describes the main terminology and procedures in face recognition algorithms, explains which are the current solutions and its features, as well as the theoretical knowledge necessary in the development of the Eigenfaces algorithm, and lastly, a brief description of the wireless sensor node employed in the hardware implementation of this technique.

Chapter 3 defines the main requirements every solution should have in order to have a fair comparison with common specifications between different versions. The flow followed for obtaining each one is included, as well as particular characteristics for every implementation.

In Chapter 4, verification methodologies are detailed. This work was tested at different levels; not only the usual simulation and embedded oscilloscope verification are applied, but also partial numerical results and images could be easily verified thanks to the TCP/IP socket established and the scripts developed.

The summary of the research work and achievements can be found in Chapter 5, where the results obtained are commented and discussed. A further work is shown, including the capabilities of the final hardware system, additional functionalities and the progression towards the new design flow with FastCUDA.

Finally, Chapter 6 presents some conclusions and the future lines of this work, in addition to presentations and some other projects this Master Thesis has a close cooperation with.

Chapter 2. GENERAL CONCEPTS AND STATE OF THE ART

The generalization of software solutions for face recognition systems is a fact nowadays. Since April 2000 [23] it is possible to use open-source libraries with functions that can make easier the task of facial detection and recognition [24]. In contrast, in this Master Thesis are studied and compared hardware, CPU and GPGPU solutions. Although some of the main concepts could appear to be well-known, it is necessary to clearly define the concepts involved in the process because normally they are misunderstood or even mixed up. Consequently, in this section the principal terms are explained, a brief overview of the current solutions is introduced, the specific algorithm is detailed and, in the end, the starting point for the implementation of this research work is presented.

I. Recognition, Detection and Identification

Face recognition seems to be very intuitive and part of the natural daily life for humans. However, if we are dealing with computer science, it was not until relatively recent years that it became a matter of fact, while previously it was more a topic for science-fiction movies or theoretical studies than realities. First of all, it is necessary to clarify the terminology used in order not to get confused by wrong concepts, so this section would explain the differences between core terms and definitions.

Face recognition is one type of visual pattern recognition problem, where the system commonly is classified in four independent modules, as can be seen in the processing flow depicted in Figure 5. The first step is to seek and localise one or more faces in the image, subtracting the background by selecting only the face area for further analysis. This first process is known as "*detection*". It is worth noting that detection does not imply any personal identification. For instance, if detection was being executed by a mobile phone application, it would be sufficient to put a square enclosing the face of each possible individual.

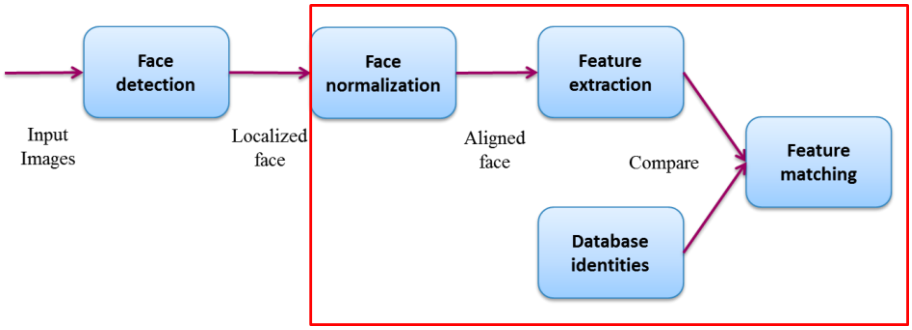


Figure 5 Processing flow of face recognition.

However, in order to make a fully recognition from a picture to an identity, detection step usually also implies to make some calculus or transformations for the next stage of the process, such as resize the input image or take into account its scale compared with the reference, place landmarks (nose, eyes, ears and chin) or give some measurements such as the contrast or brightness.

With those data, face "*normalization*" could be performed in order to bring robustness. This second phase allows recognizing faces even with varying pose and illumination (Figure 6). The geometrical normalization process transforms the face into a standard frame by face cropping, warping or morphing. The light normalization could also be applied in order to improve the accuracy of identification. The automation of this phase is not trivial, there are several researches [25] in this field and a further study is out of the boundaries of the purpose in this Master Thesis.

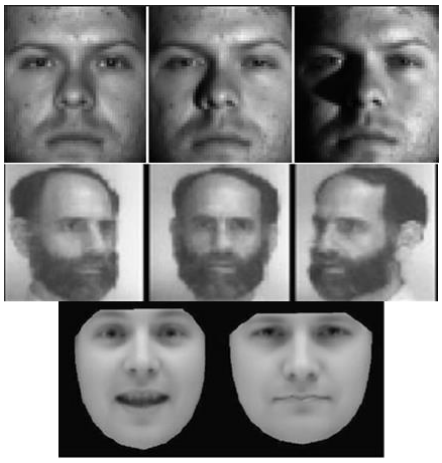


Figure 6 Illumination, pose and geometrical variations in faces.

Once a face is conditioned to a certain restrictions, feature "*extraction*" can be executed. There are different methodologies and procedures, and the main solutions are discussed in Chapter 2.II. These features are used for comparing with a reference database of identities. The more subjects and the more variety in lighting and pose conditions of every person are introduced in the database, the more accuracy will be obtained in the results. Again, there are several groups currently studying how introducing certain features, such classes with common characteristics [26] - elder people, smiling, angry, etc. - can make possible to predict the age of individuals [27], the gender or even the mood with only a picture [28].

There are, at the same time, two possibilities when comparing with a reference: authentication and identification. When a query image is contrasted against an identity in order to verify a candidate, the process is called "*authentication*", while if the unknown image is compared to the whole list of candidates in order to match it with the more probable person from the sample, an "*identification*" process would be taking place [29].

As it will be seen in Chapter 3.I, in this research work it is required to authenticate individuals, although an identification extension is immediate and it will be also provided. In turn, it is supposed to have cooperative subjects in a constrained environment, as could be a pass control with face identification in the entry of a security building. On the contrary, there are applications for surveillance systems, where non-cooperative individuals - sometimes intruders - are attempting to conceal his features (Figure 7) and the aim would be to monitor and register the features of a suspect. Besides, regarding the first assumption, the first step (detection) of the full face recognition process could be avoided, and even the normalization part would be considerably simplified.

Figure 7 Non-cooperative subjects make harder detect faces.

In the same way, when researchers address face recognition term, they usually refer the three last steps: normalization, feature extraction and identification [30]; while detection by itself is another different field of study. Next section shows some

working devices, proposed platforms, existing algorithms, libraries of functions and databases developed until today, in order to illustrate the level of complexity and detail that could be achieved.

II. State of Solutions

Pattern recognition techniques have been studied since the early fifties [31], becoming popular with the spread of computers in the nineties. Most of the current face recognition algorithms were discussed and evaluated by authors along those years. Therefore, the effort made at the present time is towards accuracy and robustness of the results, changing the perspective toward new methodologies or bringing more effectiveness to already existing algorithms. Even so, it is still a hot topic in many fields such as electrocardiography and the study of heart diseases [32], or 3D human face model and its faithful representation [33], [34].

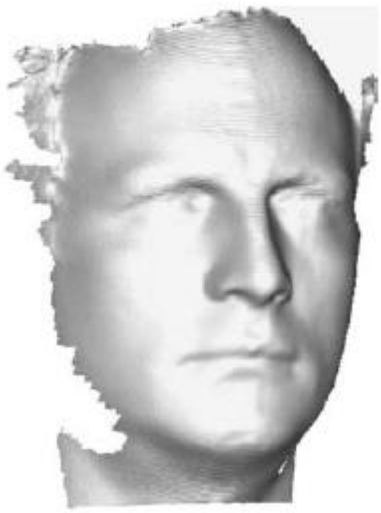


Figure 8 3D face acquisition from the study of Bronstein et al.

The first step [35] in order to create a recognition system is to obtain a large number of face images. This is called the dataset of faces, and it is used to train the recognition system for obtaining a database of identities. These faces should have the same size and should be centred. Some databases used several pictures of the same person in order to be robust under gesture variations and light conditions, but in the simplest case it is desirable to have at least one face of every subject, a snapshot which should be taken under similar conditions.

The most successful datasets employed in recognition are usually created by universities or big companies. For instance, Yale University launched in 2001 a free collection of 5760 single light source images of 10 subjects each seen under 576 viewing conditions (9 poses x 64 illumination conditions) [36]. The Massachusetts Institute of Technology (MIT), in turn, provided two training sets, one with high resolution pictures and another with 3240 synthetic subjects, though for building 3D head models [37]. And just to mention another example of these kind of contributions, AT&T Corporation (The ORL Database of Faces [38]) and the National Institute of Standards and Technology (Colour FERET Database [39]) are two institutions that have also worked with face recognition systems and have two datasets of faces available for free. If the reader had more interest in this field, many more examples can be found collected in web-pages, such as www.face-rec.org [40].

Chosen one database, there are several algorithms developed nowadays. One possible classification into groups [41] is shown in Table 1, and several comparatives between multiple algorithm have been found searching in the literature [42], [26], [43], although further information may exceed this project.

Technique	Characteristic	Algorithm example
Principal Component Analysis (PCA)	Karhunen-Loeve's transformation	Eigenfaces
Independent Component Analysis (ICA)	From Bartlett et al	ICA face recognition
Linear Discriminant Analysis (LDA)	Discriminates classes	Fisherfaces
Evolutionary Pursuit (EP)	Evolutionary algorithm	Egenfaces + Evolution
Elastic Bunch Graph Matching (EBGM)	Recognize objects under transformations	Trace Transform
Kernel Methods	Not linear methods	Kernel Eigenfaces. Kernel Fisherfaces
Trace transform	Allows rotation, translation and scaling	Trace Transform
Active Appearance Model (AAM)	Statistical model	Statistical Models of Appearance
3-D Morphable Model	Encodes shape and texture	3D Morphable Model
Bayesian	Probabilistic	Bayesian Face Recognition
Support Vector Machine (SVM)	PCA + LDA	Vector Machines,
Hidden Markov Models (HMM)	Statistical model	Markov Models

Table 1 Classification of face recognition algorithms.

During the last decade, several companies have launched commercial solutions of face recognition systems. There have been found lists of more than 50 different APIs

and libraries for software developers [44]. Most of those applications are open-source [45], while others only include an executable demonstrator. On the other hand, an example of an ASIC implementation can be found in Prasanna's work [46]. Authors of this paper, implemented hardware with a Neuronal Network strategy in order to make a Principal Components Analysis (PCA) for face recognition, and they got manufactured it in a 1.5 x 1.5 mm. chip.

There is also a commercial solution for FPGAs offered by Xilinx, called Image Characterization, but it has some problems detailed as follows. The main inconvenient features of this IP core are two: the large number of FPGA resources estimated (minimum 35 DSP48A1 [47]) and that, because it is an Intellectual Property, requires additional charges and it is encrypted. The first statement requires the Image Characterization to be allocated on the static part of the device more than on the dynamic reconfigurable area; while the second forces to purchase a full utilization license for the IP, in addition to be non-readable.

The study of those alternatives underlined the necessity of designing from scratch one single and simple algorithm that could be easily implemented in all the platforms - hardware, CPU and GPGPU - in consideration, mostly because none of them could be exported to other platforms. Those examples have been used more for figuring out a certain number of characteristics that the final solution should include in global, than as guidelines or references. In the end, in order to have a fair comparison, the best solution shall be implemented sharing the same requirements in every platform. The Eigenfaces algorithm is the most disseminated and extensively documented method, and since the online part of the algorithm is not so complex but rather computational expensive, it becomes the ideal technique to implement in this research work. In the following section the main principles of Eigenfaces are presented.

III. Eigenfaces Algorithm Basics

Kirby and Sirovich [48] developed a low-dimensional procedure for the characterization of human faces based on the Karhunen-Loeve's transformation. This work was utilised by Turk and Pentland, who reinvigorated the research field of face recognition with the publication of the popular Eigenfaces method [49]. According to them, there are two parts of the system, one calculated offline and another in the final device. For having operative the recognition system device, it is only necessary to have available the results of the first computations as input data. First, every step of the offline procedure will be explained.

Step 1. Human faces can be seen as an infinite linear combination of orthogonal functions of stochastic features. Among all the possible images in a picture

(the space of possible pictures), only a tiny fraction of them are faces. They are known as the subspace of human faces, and in order to study different individuals in this subspace, it is required, in the first place, to acquire a set of representative real faces. These training faces (Figure 9) are matrices of pixels $\{I_1, I_2, \dots I_M\} \in R^{RxC}$ (R=rows, C=columns) and constitute the database of faces once centred and adjusted.

Figure 9 Training faces.

Step 2. The usual procedure is to take each row of pixels from those images and concatenate them with the next row, until the image is a single vector column $\{\Gamma_1, \Gamma_2, \dots \Gamma_M\} \in R^D$ (being $D = RxC$) that represents each original face.

Step 3. Those vectors can be added up and divided by the number of samples (M) in order to obtain the arithmetic mean vector $\Psi = \frac{1}{M}\sum_{i=1}^{M} \Gamma_i$, which represents the average face shown in Figure 10.

Figure 10 Average face from people of CEI.

Step 4. Build the matrix Φ with the results of subtracting the average face from previous vectors $\Phi_i = \Gamma_i - \Psi$.

Step 5. The covariance matrix is usually computed as $C = \frac{1}{M}\sum_{i=1}^{M} \Phi_i \Phi_i^t = AA^t \in R^{DxD}$ where A is a DxM matrix. This matrix C is the input data of a Principal Component Analysis (PCA), last step of the offline part of the algorithm.

Step 6. Compute the eigenvectors u_i of the covariance matrix. These eigenvectors are known as Eigenfaces because of their appearance when printed as RxC

matrices. The method followed is a Principal Component Analysis (PCA), and further information is given below.

The sixth step is too computational intensive even for recent computers. Fortunately, there is a solution for this issue: instead of computing the eigenvectors of the matrix (AA^t), it is possible to compute the ones of the matrix (A^tA). For instance, if each picture was 144 rows by 176 columns, (AA^t) would have 25344 rows by 25344 columns, so there would be 25344 eigenvectors of 25344 elements each one. Nevertheless, for 128 samples of faces, (A^tA) would have only 128 rows by 128 columns, and so, only 128 eigenvectors of 128 elements each one could be obtained.

Following this technique, it will result:

Step 1. Computing the eigenvectors of (A^tA): $(A^tA)\omega = \lambda\omega$ being ω the reduced eigenvectors of the matrix (A^tA).

Step 2. By pre-multiplying the previous equation by A:

$A(A^tA)\omega = \lambda A\omega$ Since $C = AA^t$ then:

$CA\omega = \lambda A\omega$ By calling $\mu = A\omega$:

$C\mu = \lambda\mu$ So μ are the eigenvectors of the matrix C.

It has been obtained the transformation from the reduced eigenvectors ω of the matrix (A^tA) to the M most significant eigenvectors μ of the large matrix (AA^t). Those eigenvectors are the features of the human face. Each one represents an orthogonal axis of the subspace of faces, and three of them can be seen in Figure 11. The eigenvalues of A^tA are the same to the most significant in AA^t.

Figure 11 Eigenfaces.

Step 3. The dimension of the subspace can be reduced even more by keeping only the K eigenvectors (μ_i) corresponding to the K largest eigenvalues (λ_i) of the Eigenfaces. It is possible because most of the eigenvalues are zeroes or negligible in contrast to the most significant eigenvalues.

Step 4. Those K Eigenfaces (μ_i) should be normalised for further utilisation, so in the end, there will be K normalised Eigenfaces (u_i).

Reached this point, it has been calculated the average face, centre of the subspace of the human faces, and the Eigenfaces, axes of the subspace of the human faces. With this information it is possible to represent every single face as a linear combination of a number of weights on those axes, for instance, 127 coordinates. Accordingly with the numbers in all the examples exposed, the storage of every identity in a system with this information only requires 127 bytes, which in comparison with the original 25344 bytes of the original picture is a huge reduction.

It is worth to highlight that this algorithm is supposed to be universal: beyond being applicable to the training faces, it can also be utilised with faces from the outside of the dataset.

Having processed the offline part of the recognition system, now it is presented the method for obtaining identities. This part of the algorithm could be calculated offline by simply adding subjects to a database of identities. However, in the mandatory online part of recognition system will always be necessary to have the authentication part, so this fuctionality should be included. The steps are the following:

Step 1. Subtract the average face to the normalised picture of the unknown person (j) to recognise. This step is the same as above, the one that was previously done with the training images $\Phi_j = \Gamma_j - \Psi$. The arithmetic mean of the faces (Ψ) was previously calculated and provided as input data for the recognition task. The image resulting can be seen in Figure 12, and it represents the face in the origin of the subspace of faces.

Figure 12 Query face, average face and subtracted face for one candidate.

Step 2. Projection onto the human face subspace. The previously calculated and normalised eigenvectors u_i are the axes of the subspace of faces; therefore, each subtracted image Φ_j can be expressed in the face subspace. Such operation is made by projecting every candidate onto the Eigenfaces, obtaining thus the weights ($w_{ij} = u_i^t \Phi_j$) or coordinates for every axis:

$\widehat{\Phi_j} = \sum_{i=1}^{K} w_{ij} u_i$. If the purpose is only to store some identities in the system, it would be enough to save these coordinates and the name of the person studied. If the system already has some identities and the aim is to authenticate one unknown person, two more steps are required.

Step 3. Each subject has one unique representation with those coordinates and Eigenfaces, and it can be seen as a single point in the subspace of face. The transformation to the real picture returns faces rebuilt in a synthetic manner. Figure 13 is one example of a synthetic rebuilt face. For instance, the optimal number of principal components to keep could be studied by measuring the accuracy in representation with certain number of Eigenfaces.

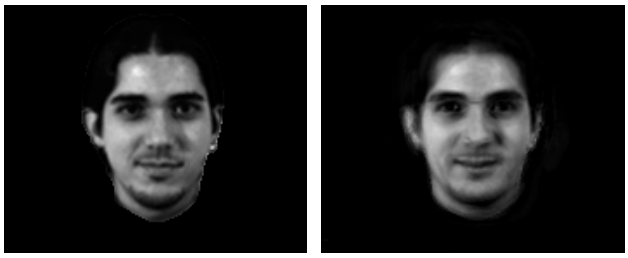


Figure 13 Synthetically rebuilt images of the candidate 9 with 127 and 50 Eigenfaces.

Step 4. The last part is to compute the Euclidean Distance. Given the coordinates of the projected query face (j), and the coordinates in the same subspace of one candidate previously known (i), it is possible to distinguish if they are the same person or not by calculating the Euclidean Distance between those points $\delta_2 = \left\|\hat{U}^T(\Phi_i - \Phi_j)\right\|$. It is defined a threshold T_i or level of acceptance below which it can be safely said that both faces are the same identity $\delta_2 \leq T_i$. If the distance is greater than this value, the candidate does not match with the query face and the authentication is rejected. As it was said before, an extension of this application can be easily made by going along all the database of identities and assuming that the minimum distance corresponds to a correct identification, provided this distance is also less than a threshold.

Although it is not the most common technique, Eigenfaces could be used also as a detection algorithm [50], by computing the distance between the possible face image and its projection onto the face space. $\delta_1 = \left\|(I - \hat{U}\hat{U}^T)(\Gamma_i - \Psi)\right\| = \left\|\Phi_j - \widehat{\Phi_j}\right\|$ In the end, it is the same as verifying if a picture is well rebuilt by projecting it onto Eigenfaces. If the distance is greater than a certain threshold, it means the image is not a face and because of that, it will have a poor rebuilt result, with many errors when comparing with the original picture. As it has been seen in Figure 14, a zebra is

not well defined with human Eigenfaces, although the representation tries to imitate some features.

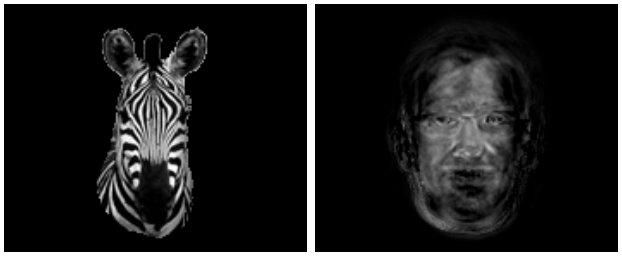


Figure 14 Attempt to rebuild a non-human face image with Eigenfaces.

Once the theory and main considerations have been shown, there is nothing left to present but the platform where the hardware and some CPU parts of this work will be implemented.

IV. Starting Point

High Performance WSNs and Autonomous Robots share a certain number of characteristics, such as a limited amount of energy to spent, or the sensing/acting abilities while processing real time data. What is more, WSNs have become an appropriate solution for certain functionalities of robots, and this became evident after the appearance of commercial wireless platforms with robotic related application [51] [52]. At the same time, a WSNs platform for High Performance application was designed and implemented in CEI, which was presented in the Final Degree Project of the author of this Master Thesis [20] as the first stage of RUNNER.

As illustrated in Figure 15, the platform has four different PCBs. The red one is the power supply layer, called HiRePower. It could be powered by a lithium battery or through a mini USB connector, and it is able to charge the battery while supplying five different voltages on five different rails. It also has an automatic control that follows the demanded current (up to 1.6A) with up to 97% of efficiency.

The blue layer is the brain of the platform, it is called HiReCookie, and its main characteristics are: a smart management of power islands, executed by a tiny microcontroller but directed by the decision made in the FPGA with the power consumption measured; the capability of having, simultaneously, an embedded microcontroller in the FPGA for running software programs together with hardware accelerators embedded to speed up some functions; and finally, the ability of being dynamic and partially reconfigured, even remotely.

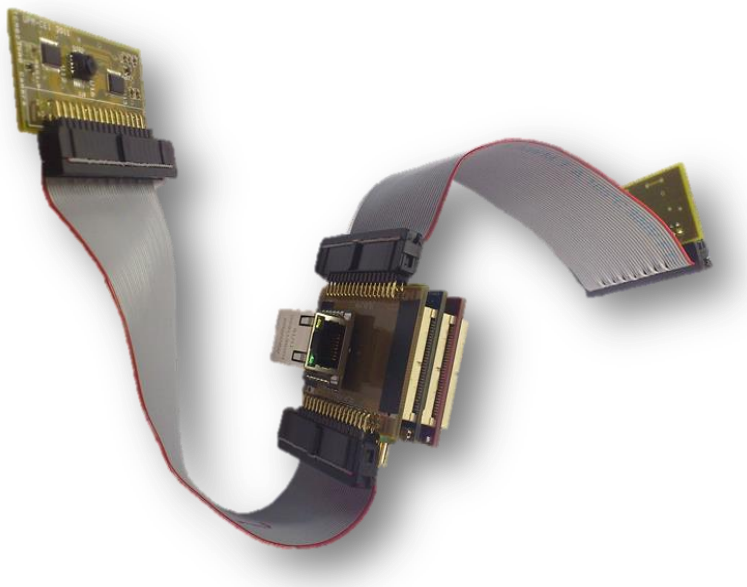


Figure 15 High Performance WSN Platform designed in CEI.

The Ethernet layer has an RJ45 connector and a W5200 microcontroller that has a SPI communication with the FPGA, for interchanging with Internet the desired information, from the FPGA to other devices and vice versa, such as Nao robot, other HiReCookie or simpler nodes, computers or smartphones. At the same time, it is the link with the PCBs of the cameras.

The last two PCBs are replicas of the same board, which has a digital sensor camera with VGA resolution (640 by 480 pixels) and 30 fps. The communication with the FPGA is an I2C interface, which makes possible an easy control of the operation and modes of the camera.

The final integration of this novel platform was performed by implementing camera and Ethernet controllers for the FPGA, as well as an application that sent images from the cameras to a client in a personal computer.

In Figure 16 is depicted the diagram of the system that was made before this Master Thesis and the communication between blocks. Once the communication is established between the client and the server (Ethernet module), the SPI interface translates the message to MicroBlaze, so in the end petitions launched by external devices will be listened by the FPGA, and some information will be returned after the task has been executed.

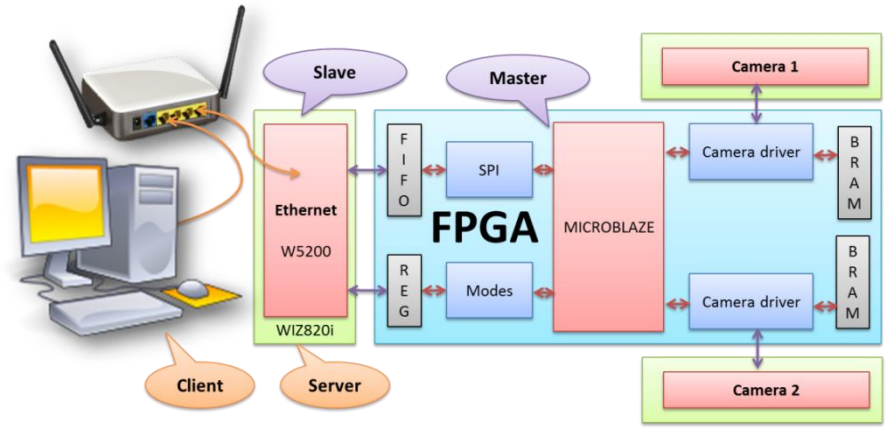


Figure 16 Diagram of the initial system and interconnections.

Although this system is able to take pictures, for the second stage of RUNNER it is necessary to design two new blocks: Face Recognition and Stereo Matching. As it was mentioned above, the first one will recognise people in a controlled environment based on the Eigenfaces algorithm, while the second will be used as a simple navigation system, by avoiding the direction where the closest objects are and following the direction of the furthest point in the view. In the development of the recognition system, this new modules imposed certain requirements that will be explained in Chapter 3.I.

Chapter 3. REQUIREMENTS AND IMPLEMENTATIONS OF RECOGNITION SYSTEM

Once framework and general concepts of this work have been studied, the requirements for designing different solutions of the selected recognition system should be presented. Several versions will be designed pursuing the same pattern, in order to make a fair comparison in the end. The process and methodologies of design will be detailed. Following a rising level of complexity, it is described from the development of the first functional system and first applications, up to the higher-end implementation of the final hardware recognition system.

I. Requirements for the System Design

In order to implement systems with common characteristics, it is necessary to study particular limitations of final devices. They will become specifications for all the designs, for the sake of an impartial performance comparison.

As well as the aim of RUNNER, the principles of FastCUDA shall be studied. The author of this research work will be taking part in the team in charge of the synthesis of the CUDA Kernels, as well as being responsible of certain deliverables of both projects. It not only shall be perform the design in the WSN node, but it also should be supervised the development of CUDA code, and make sure all solutions fulfil the same specifications in order to achieve a fair comparison with the whole list of versions.

The nature of constraints this work must bear with can be divided in three different types, detailed as follows:

I.1 Related with RUNNER Project:

- The protocol of messages between the robot and the platform has to be defined. The robot will launch petitions and the visual cortex shall answer a **simple response** with the results as soon as possible. The robot will take fast decisions with those data, acting in consequence.

- In RUNNER it is desired to have two interchangeable blocks: the Stereo Matching and the Face Recognition hardware. Both shall have **reconfiguration capabilities** since the idea is to use one or the other when the robot will decide, but not both at the same time, in order to save resources. Both peripherals shall be implemented in the novel platform HiReCookie and tested online. Following the starting system presented in Chapter 2.IV, a representation of those two modules can be seen in Figure 17:

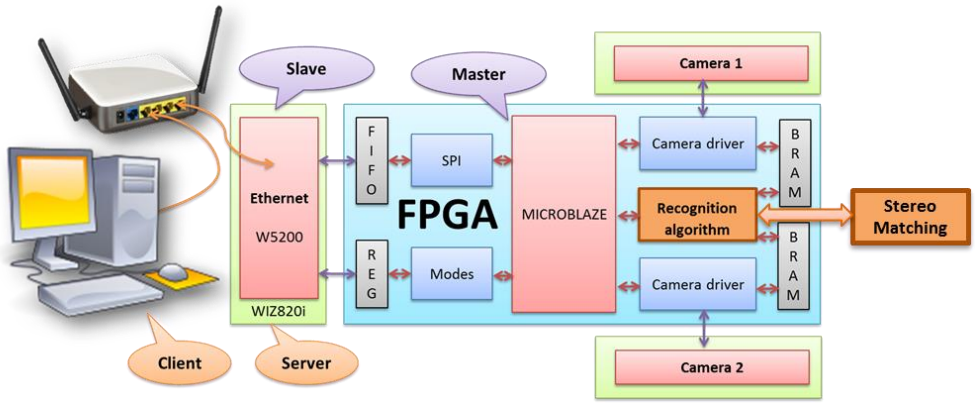


Figure 17 Reconfigurable hardware for RUNNER project.

- The **maximum** resolution available by the cameras in the project RUNNER specification is **640x480 pixels**, at a ratio of **30 fps**. Smaller face pictures should be taken into consideration, due to the fact that one face will not occupy the whole area of a picture.
- It is sufficient to process the luminance Y of the YUV encoding of the camera, so **greyscale** images will be a simpler and more practical option in this case. The extension of the solution to colour images is as simple as replicate the block for each one of the three channels and have a voting system, but this exceeds the present goals.

I.2 Related with FastCUDA Project:

- As consequence of the SystemC synthesis tool limitation, it is **not supported any floating point data type** in FastCUDA. Neither float nor double variables are allowed in the CUDA source code, so it will be avoided any effort in the C source code as well as in the VHDL description. Besides, images used will have either integer or unsigned precision.

I.3 Additional Purposes:

- This Master Thesis is focused on the **identification** part of a recognition system. It is desired to contrast face images against a database of identities. In order to achieve this purpose, it will be supported the minimal required functions and skipped non-essential ones.
- It is required to build a database of identities with people working in the laboratory of the Centre of Industrial Electronics, in order to prove the results in a real scenario. These individuals are supposed to be **cooperative**; if they endeavour to be authenticated, they will be awarded with the interaction of a robot, or with the permission to access to certain area.
- It is requested to have a fair comparison between different parts of devices that are being taken into consideration, so if **only the processing unit** for energy consumption was taken into account, no other parts should be included in the comparison. Another possibility could be to consider the whole device as one indivisible unit.

The process followed during the design should be progressive due to the complexity of the system. It is necessary to start by the simplest solution, test the functionality and then step up to the next level in difficulty until a complete system is obtained.

Due to the multiple solutions that will be implemented, it would be necessary to develop automatic tools in order to make easier the debugging task of successive versions. Besides, in order to take measurements of the accuracy in the obtained systems, it is possible to use double data type precision.

Once having exposed the requirements, and highlighted the specifications, it is the turn to present the practical work and detail how the task was accomplished.

II. MATLAB as Proof of Concept

The first step after having studied the existing solutions and the requirements of the recognition system is to start by a simple application and understand every single step that is taking place in the process of face recognition. MATLAB was chosen as the environment to develop this demonstrator application because it allows easy image processing, in addition to matrix manipulation and inclusion of complex mathematical functions. Thus, a simple application for understanding some concepts will be developed.

Later, it will be verified that this amount of pictures is enough to have a universal base of faces, by rebuilding faces from outside of this set. However, before contrasting query faces against identities, it is necessary to register the face of those subjects. This process is detailed as follows:

II.1 From a Set of Images to the Database of Identities

First of all, it is needed to have a dataset with certain number of images with different faces. These images should have almost all the features required to identify any face, such as wide or narrow cheekbones, different coloured iris, age and sex. On the other hand, non-permanent characteristics like hairstyle, and the background of the picture should be minimised. All the set of pictures must be under the same light condition, at the same scale and positioned with eyes and nose references.

As it was mentioned in Chapter 2.II, several dataset of faces were studied. They are included as a future line in order to enrich the study, making it more robust against changes in luminosity, geometry or pose. Nevertheless, it is necessary to know the features of the working people at CEI, and since there is a set of 128 face pictures, it was the chosen one for training the system.

Moreover, the availability of BRAM memory in the FPGA was studied, and it turned out to be possible to have thirteen blocks of 32 Kbytes. The appropriate scale for managing full images is the format QCIF of the camera, which needs 25344 bytes distributed on an image of 176 of width by 144 of height pixels. The camera controllers use two 32Kbytes of this memory, the processor two more 64 KB for cache memory, so with this size it is possible to have up to seven 32 KB modules – more than seven images - stored in BRAMs in this Spartan-6.

The 128 pictures of faces were coloured, they had different sizes and, on top of that, they had heterogeneous file formats, such as JPEG file compression, BMP bitmap or PNG extension. In order to align the pictures and have a simple readable format of the same size, it was employed the following BASH script, running on Windows operating system by naming the text as "*jpg2pgm.bat*" for instance:

```
convert.exe *.jpg -resize 176x144 -gravity center -background black -extent
176x144 -scene 1 image%%03d.pgm
```

Table 2 BASH script used in order to adequate the images.

As it can be easily inferred by understanding the code, the script will use the application convert.exe, which will take as an input every file in the same folder with the extension jpg in this case. The size of the found images will be accommodate to 176 width by 144 height pixels, put in the centre of the area an filled the rest of the image with black colour. The output image has the same 176x144 size, and the script

will write PGM files with the name "*imageXXX.pgm*", being XXX a number from 0 to 127. An example of the input and output can be seen in Figure 18.

Figure 18 Original and output of the BASH script.

The chosen PGM (Portable Graymap Format) is one simple greyscale format of images. It is related with PBM and PPM, being used PBM for black or white pixels and PPM for coloured ones. The structure of a PGM file is as follows:

- First, one or more lines as the header of the file. It can contain the string "*P2*" or "*P5*", depending on whether it is an ASCII file, with numbers encoding the luminosity of every pixel, or a raw one, having encoded this value with binary characters. The next two numbers are the width and height, in this case "*176 144*", and the last one is the depth of the colour, between 0 and 255 for 8bits and from 0 to 65535 for 16 bits depth.
- Secondly, the body of the image. It was chosen the binary encoding, because it requires less storage space. For instance, for writing down a file with 8 bits of depth and 5 pixels with values 74, 117, 108, 105 and 111 in a raw file it is written 5 bytes: "*Julio*", while in the ASCII encoding it would require 18 bytes counting spaces: "*74 117 108 105 111*". This example can be seen in Table 3.

# Example BINARY	# Example ASCII	# Same image in both # cases
P5	P2	
5 1	5 1	
255	255	
Julio	74 117 108 105 111	

Table 3 Example of a readable PGM file.

Although the automatic transformation returned scaled images, in a greyscale easy format, there was the necessity of applying four more transformations manually, with GNU Image Manipulation Program (GIMP). It was created a template for all the faces, which put landmarks in the area where forehead, eyes,

nose, mouth and chin is supposed to be, in addition to the extension and localisation of the face. In Figure 19 it is shown the input, template and output of the transformations described below.

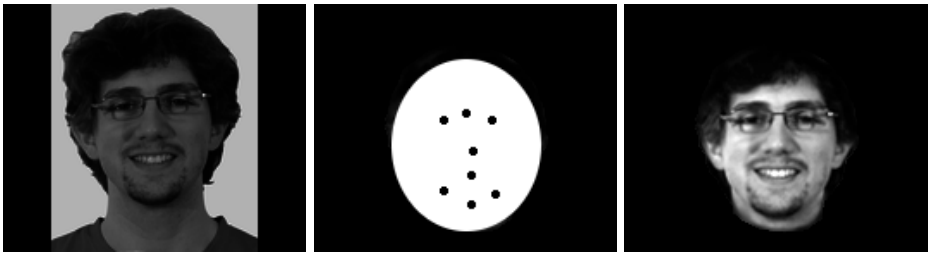


Figure 19 Result of applying 4 manual transformations.

- Resize and move every head in order to be place inside the white oval, filling up the whole area.
- Rotate and turn over the faces in order to fit with the landmarks. Sometimes the faces are not straight or looking to one side.
- Fade the limits of the shape, in order to have progressive disappearance of the features. Hair is a mutable feature and it is not desired to take it into account.
- Improve the contrast of the image and proportionate a homogeneous luminosity for all the pictures. Some pictures are too dark, or show some differences between dark zones.

Once the dataset of pictures was ready, it was written a "*base.m*" script for MATLAB. Basically, it follows the steps described in Chapter 2.III. The program is outlined below:

- Read all the PGM images of a certain folder. Each row of pixels is concatenated with the next, so that at the end each image is one column of data.
- With all the pictures, a matrix of faces is generated.
- The arithmetic mean of each row, pixel by pixel will be the average face.
- Calculate a second matrix M by subtracting this average face to every column of the previous one.
- As it was already mentioned, the calculus of the eigenvector of the covariance matrix S=M*M'/(N-1) is impractical, so they are calculated for Sr=M'*M/(N-1) and the result is pre-multiplied by M.
- A configurable number of the greatest vectors are kept, once ordered and normalised.

- These Eigenfaces are used for projecting different faces in that base prj=U'*M, so each face is represented by a maximum of 127 coordinates.

This is the so-called database of identities, and in Annex 1. Table of Identities a summary with the number of each subject and the corresponding name is included. The code of this script is included only in the CD because of its extension, as well as is the dataset.

It is worth noting that this script writes down files with some results, as the Eigenfaces and the result of projecting the set of faces onto the subspace. These files will be used in the future with the FPGA as reference, and besides with future verification scripts as a golden reference with real precision instead of integer one.

In order to verify how far subjects are from others, it was also developed some code for measure the Euclidean Distance. There are three individuals who have two different pictures included in the database; one example can be seen in Figure 20. The matrix of distances between identities, one against another, was computed, and as it was expected, the three minimum distances are those three pairs of similar faces that belong to the same individual.

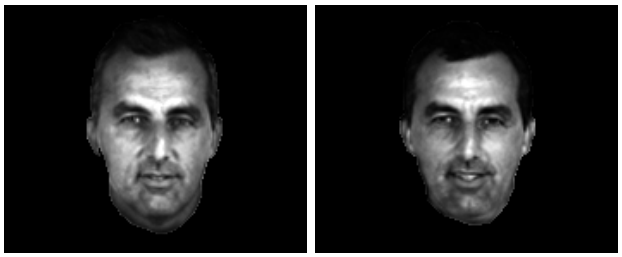


Figure 20 Two different pictures from the same subject are closer between them than to others.

The difference between equal identities and different ones is one order of magnitude higher (1236 against 19375). The distance between a face and another picture different from a face would be out of the range, far above ten millions.

The ranges of all the results were studied in order to have precise data types when implementing this system in VHDL. For instance, and accordingly to the restrictions imposed in Chapter 3.I, Eigenfaces will be multiplied by 1024. Consequently, it will result in values from - 84 up to 77, which can be stored in 8 bits and Integer data type with a 34% of safety margin. The losses of accuracy are below 1%, taking into account the rounding carried out.

It is highlighted that there is no need for computing the square root of the Euclidean Distance at all, so it will not be calculated in any version of the recognition system. The fact of having norms instead of modules only makes the distance to follow a quadratic function shape.

As first demonstrator of the recognition system, it was rebuilt every initial face, with the reference of the dataset of identities (values of projection onto the Eigenfaces) and the base of the subspace of human faces obtained previously as the only inputs. The accuracy of the method can be seen in the Figure 21, since the synthetic images hardly can be distinguished from the original ones.

Through this meticulous process, it has been created the database of identities and a software recognition system. Furthermore, it has been explored how accurate it is, and the main concepts are understood. However, until this point, there have only been made tests with images already processed by the designer, and used as part of the training set. In Chapter 3.II.3 face from outside the database will be studied, and in Chapter 3.III.2 real images will be investigated. It is the turn to develop a second application that will make use of the theory studied and achieve interesting results.

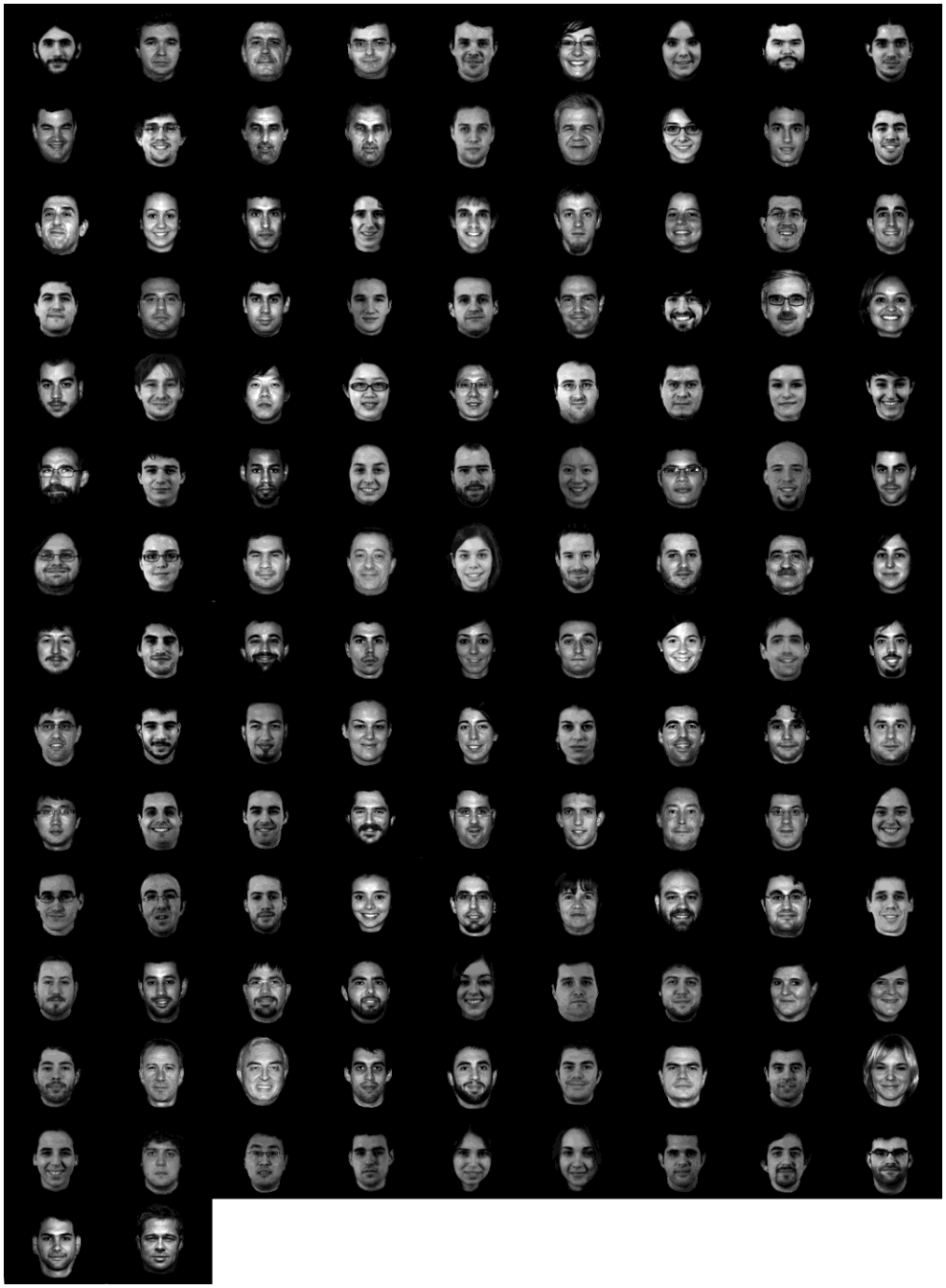


Figure 21 Synthetically rebuilt image of all the database of identities.

II.2 Fade Faces

It has been studied that every one of the previous faces are points of the subspace of faces. This assertion brings up the following question: is it possible to represent artificially faces by assigning random values of coordinates of the human face subspace?

Trying to answer such question, it was written the following MATLAB code (Table 4), that basically attempts to rebuild faces given random coordinates of the subspace of faces:

```
% Rebuilt random people from outside of the base
people=input('How many random people you want? ');
factor=1000;                    % Change this value and the face
                                % will be close/far from the origin
for i=1:people,                 % Coordinates of the person
  prj4=double(int16(factor *(0.5-rand(127,1))));

  rebuilt4=U*prj4+u;            % Rebuilding people
  NEW_IM4=uint8(vec2mat(rebuilt4(:,:)',C));

  imwrite(NEW_IM4,'_-_Mutant.pgm');
                                % Write down an image
  image_new4=uint8(vec2mat(rebuilt4(:),C));
  imtool(image_new4);           % Watch with MATLAB the image
end
```

Table 4 Rebuilt random people from outside of the base.

As a result, there were obtained faces with random features, and by changing the factor value it was possible to make stronger their weight, going from almost the average face – origin of the base – outward to saturated faces (Figure 22). It is observed that real faces follow a non-random distribution, giving more importance to the first values, which, at the same time, are the most significant ones.

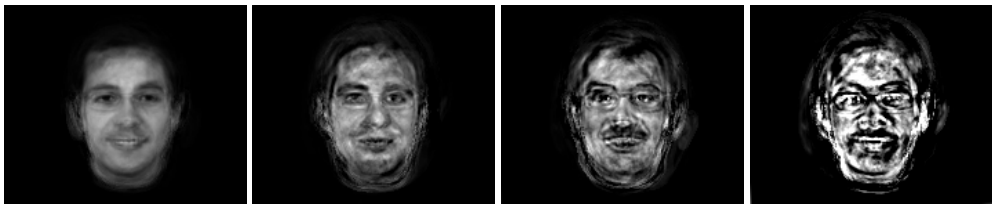


Figure 22 Progression of faces with factor equal to 300, 800, 1200 and 3000.

In conclusion, it would be necessary to study the distribution of the coordinates in order to create random people from scratch. This is far from the purpose of this Master Thesis and thus is not going to be included.

Even though random points are not always real faces, certain points can be the transition between two individuals. For instance, straight lines can link two points, and features along the path experiment a linear transformation. In Table 5 it is shown the MATLAB code for watching the transition between two identities of the database of faces.

```
% Fade faces: transition between individuals.
subject1=input('First subject: ');
subject2=input('Second subject: ');

point1=prj(:,subject1);
point2=prj(:,subject2);

transitions=uint8(input('number of transitions between subjects: '));
increment=point2-point1;

line=zeros(N-1,1);      % Will contain the points from subject1 to subject2
for i=0:transitions+1,
   advance=(double(i)/double(transitions+1))*increment;
   line(:,i+1)=point1+advance;
end

rebuilt_fade1=U*line+repmat(u, 1, transitions+2);
NEW_fade1=uint8(vec2mat(rebuilt_fade1(:),C));

imwrite(NEW_fade1,'Fade.pgm');
imageFade=vec2mat(NEW_fade1,C);
imtool(imageFade);
```

Table 5 MATLAB code for watching transitions between subjects.

This code was used for obtaining a configurable number of faces between two of the identities of the database. Three examples can be observed in Figure 23, where transitions seem to be very natural; beard, smile and facial features from the second subject appeared step by step in the first one as if it were turning into someone else, until finally the second individual is reached.

Figure 23 Morphing or fading between different individuals.

There have been employed images from the original database of faces, although it is mandatory to test different faces in order to assure the universal applicability of the system. This is precisely the following issue to be presented.

II.3 Rebuilding External Faces

Represent faces which were already in the database is not a demonstrator of being a universality base for faces. The vectors should be tested with other pictures different from the ones utilised in the calculus of the subspace of Eigenfaces.

First, there were considered images from individuals whose identities were already in the database, but the picture analysed was taken years after. This leads to the question: have the features of one person changed that much for not recognising him as the same individual?

For instance, a younger version of Brad Pitt in addition to snapshots of CEI workers served for obtaining the Eigenfaces mentioned above. The results of projecting an older Brad Pitt in the base of faces proportionated certain coordinates, and the combination of those values with the vectors rebuilt the older Brad Pitt. The image can be seen in Figure 24.

Figure 24 Rebuilding a new Brad Pitt with Eigenfaces.

It would be worth noticing that, even with only 128 images, the base is accurate enough to identify the same person after some years. The Euclidean Distance was the minimum between the younger and older exterior versions of the same person than between the older and the others, with once again one order of magnitude of difference. Despite this good result, it should be considered to increase this number of images in the future, what will enrich the variety of features presented in the database.

Before recognising this new picture, it was necessary to repeat the previously described steps for having a picture with a face detected without its background and normalised afterwards. It is a request for the future of the full recognition system to consider these steps in the detection algorithm, and try to include the option of doing it automatically. Nevertheless, it will be tested with collaborative subjects in Chapter 3.III.2.

One more example can be seen in picture Figure 25. In this case, the database has inside two different images of J. A. Cobos, and a third one was used for rebuilding and identifying. The distance in this case was 15 and 14 times smaller for each one of

the two matches, when with Brad Pitt was 11 times. It can be inferred that having pictures of the same age and a greater number of them in the database do help the task of identifying people. The synthetic rebuilding has been improved, and the features are better imitated.

Figure 25 Person with the same age and more presence in the database.

Following the usual procedure, it is desired to rebuild a totally extern person from the database. In this case, the subject Pierce Brosnan - in the James Bond movie - was not as well defined as in other occasions. The system tried to do its best in order to adjust to the facial morphology of the actor, and the returned image can be watched in Figure 26.

Figure 26 Database extern person reproduced by the recognition system.

Pierce Brosnan was not found between the identities of the database, as it was expected, and the distances with other individuals were into the usual range. It would be a good strategy for a real recognition system to store every candidate detected, known or unknown, in the server. Eventually, both the database and the Eigenfaces could be updated offline, and thus, this would lead to better performance in the recognition process. Moreover, there are several techniques for improving a recognition system. For instance Saishanmuga et al. [53] have implemented a genetic algorithm, taking advantage of neuronal networks for recognition systems, following the example that Turk and Pentland proposed in 1991.

It was presented the last result of a system modelled with the help of MATLAB. From now on, there will be real recognition systems working on portable devices, in particular in a HiReCookie node and in a laptop afterwards.

III. C Code Development

Once mastered the face recognition technique, it will be implemented in a high performance Wireless Sensor Node. It is a good method to start with a simple design, easier to debug than the final system, and gradually incrementing the level of complexity. Following this approach, it was developed the first embedded recognition system in software, running in a MicroBlaze soft-core processor. The main features, what the last implementation would need in the future, were settled in this process. Consequently, it supposed a step forward the validation for certain parts of the final application.

Examples of those advances are: the protocol established and followed along all versions implemented in HiReCookie, as well as memory mapping and development of functions for the RAM.

In this section of Chapter 3, there will be seen the main functions of the software developed, and in Chapter 4.I it will be presented the methodology employed for contrasting the results returned. Once verified the C code produced for this first embedded recognition system, it was also used as reference for one further version in CUDA language.

III.1 Commands and RAM Functions

The diagram of the initial system and interconnections was already explained and also shown in Figure 16. In spite of the commands already proposed for taking pictures, it is necessary to establish a communication protocol between the external client – or a robot - and the embedded microprocessor of the FPGA. Novel functions for face recognition are going to be designed, and they should have a place in the TCP socket application, in order for launching the execution of the algorithm.

In Figure 27 it is shown the client running on a personal computer. This represents the remote control station that a user actually can manage. Without having any knowledge of what the system does, he could request petitions to the FPGA and it would entail a computation and a simple answer or a value would be returned.

```
This program is able to ask for images to the server

If you want an image from the camera 1, type p100! and press enter key
If you want an image from the camera 2, type p200! and press enter key
If you want an image from both cameras, type pb00! and press enter key
...
Developing more command, such as w100! w200! wb00! swd0! swn0! dbXY! rbd1! rbn1!
...
to send files to the server type file! and press enter key

Type exit to terminate
```

Figure 27 User interface to send commands to the FPGA.

It would be worth noting that messages can be received by the device from any part of the World through Internet and from any platform, such as the Ethernet-based internal network of Nao robot. The FPGA is able to manage up to 8 clients in a sequential manner. The first client is always the security client, because in case there were more than itself, it would be the one with communication privileges, although all clients would be listened.

If it was desired, in addition to the command window, it would be possible to launch a second terminal in the computer where is being developed the software of the embedded microprocessor. An image of the information it returns is shown at the left of Figure 28. This second terminal usually is helpful for debugging issues, but it has much less data traffic capacity than an Ethernet connection because it is based on a JTAG interface. Therefore, for this system, Ethernet communication will also be useful for sending large amount of data, such as images, for an easier debugging (see the right part of Figure 28).

Figure 28 SDK terminal compared with Ethernet socket for debugging.

Once studied the fundamentals of the communications, it was described a pattern for the protocol that Nao will follow in order to be interpreted as commands by the FPGA:

- Every petition will be composed by exactly 5 bytes or characters, being the last one always "!".
- The first char will be the operation code; to take a picture, start the recognition task and launch the Stereo Matching procedure among others.

- The second involves which camera is going to take part, 1, 2, both or none.
- The third is the client that is intended to receive information, which can be translated in the end as debugging client, Nao robot or nobody. Only in case a distance XY is requested, this byte would be related to the X of the query pixel.
- The forth char is reserved for a configurable value, as could be the face it is desire to identify or the Y pixel of the image, when detecting distances.

A summary table is presented in order to clarify the list of possible commands, following this protocol and putting some examples of use.

	Bytes	Define	Possible values
Operation	1	Desired operation	'p' = take a picture 'w' = send a taken picture 'r' = face recognition 's' = Stereo Matching 'd' = XY pixel distance
Identification	1	Camera taking part	'1' = Camera 1 '2' = Camera 2 'b' = Both cameras
Send to	1	Who would be the receiver	'n' = Nao / Choregraphe 'd' = Debug client
Configurable value	1	Against who is made the test	#0 = All #1-#128 = subject 1 to 128
End	1	Last char	'!'
Examples	5	All the possibilities are:	p100!, p200!, pb00!, w100!, w200!, wb00!, sbd0!, sbn0!, dbXY! rbd#0!...rbd#255!, rbn#0!...rbn#255!

Table 6 Commands for the communication with the robot Nao.

The returned value for recognition task (rbn_!) would be one byte, in case authentication is requested, but 128 bytes for identification. In order to be launched, identification has reserved the value #0, while subjects are authenticated accordingly with its position (see Annex 1. Table of Identities).

The meaning of bytes will be the distance from the picture image to the studied candidate, so a perfect match will correspond to x00, while bigger values should be compared with a threshold defined in Nao. In case of identification, each byte is the result of the distance with each subject and Nao should choose which one or ones are close enough. For "*sbn0*!" petition, 2 bytes will be returned pointing to the pixel with the deepest depth found. On the other hand, when "*dbXY*!" command is sent, it will be returned the distance obtained for the XY pixels asked in the request, scaled in one byte.

All those commands were tested with a Nao robot at the Castilla La Mancha University and a HiReCookie node inside the CEI laboratory. The robot launched petitions and the HiReCookie answered, and this resulted in a consequent performance from the robot part (send greetings or do not trust somebody, go to the right or left, etcetera).

More than these Nao commands were developed for debugging purposes, and they are summarised in Table 7. A PGM header was also sent with the image when returning pictures, because it was simpler than accommodate the client to every case.

Command	Define
see1!	See the mean face from the RAM
see2!	See the eigenvectors from the RAM
see3!	Send the reference identities stored in the RAM
see4!	See the image which is being processing
see5!	See an test image
name!	Simply send "*Julio*"
edge!	Execute a Sobel filter with the camera 1
file!	Receive files, images and references
set0!	Go to the initial position for storing input data
copy!	Copy the test image into the input space and set the position pointing to test

Table 7 Debugging commands.

Accordingly to this commands, a mapping of the memory was done in the file "*RAM.h*", as can be seen in Table 8. Optimised functions for writing and reading one or burst data from RAM memory were also written, but they are only included in the digital version of this Master Thesis because of the large number of lines of code.

```
#define Image_init    0x00010000 /**< Position in the RAM for the first image */
#define Image_size    0x00006300 /**< Size of an image */
#define Number_Images 0x00000080 /**< Number of Eigenvectors +1*/
/**< Position in the RAM for the average image */
#define RAM_average   (Image_init)
/**< Position in the RAM for the Eigenvectors */
#define RAM_vectors   (RAM_average+Image_size)
/**< Position in the RAM for the reference */
/**< this is conservative, Image_size blank */
#define RAM_reference (RAM_vectors+(Image_size*Number_Images))
/**< Position in the RAM for testing */
/**< this is conservative, N*2 blank */
#define RAM_tst       (RAM_reference+((Number_Images*(Number_Images*2))))
/**< Position in the RAM of an example of image */
#define RAM_image_in  (RAM_tst+Image_size)
/**< Position in the RAM for an output image */
```

```
#define RAM_image_out (RAM_image_in+Image_size)
/**< Position in the RAM for my mean subtracted image */
#define RAM_image_sub (RAM_image_out+Image_size)
/**< Position in the RAM for the coordinates */
#define RAM_point_coo (RAM_image_sub+(Image_size*2))
/**< Position in the RAM for the distance result */
/**< this is conservative, 4 blanks */
#define RAM_comp_dist (RAM_point_coo+(Number_Images*4))
//#define RAM_comp_next (RAM_comp_dist+(Number_Images*4))
```

Table 8 RAM memory mapping.

With all this information, it only remains to describe the procedure for initialising the recognition system. Once the socket is established, three files should be transmitted from the client to the RAM memory: the average face, the Eigenfaces and the reference – or coordinates of the known candidates. For the described task it will be used "*set0!*". This command is also useful if future updates are required. A screenshot shows this process in Figure 29.

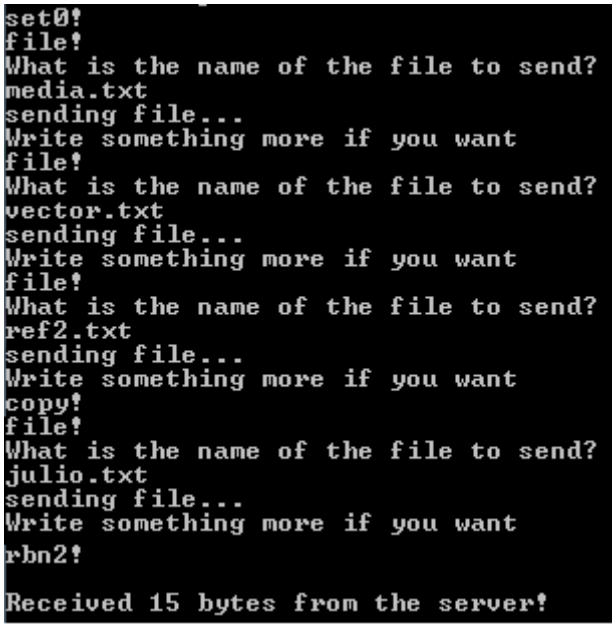


Figure 29 Initialization of the recognition system.

Another possibility is to send one test image: once the position is set to "*test*", a "*file!*" command would do the rest. Once an image is in this position, a "*copy!*" command would put the test image in the input of the system.

All those commands will be employed in the subsequent versions, described in next sections.

III.2 SDK Recognition Functions

First, it is outlined the modules that all the HiReCookie versions will include. In Figure 30 it is depicted all the initial block and interconnections.

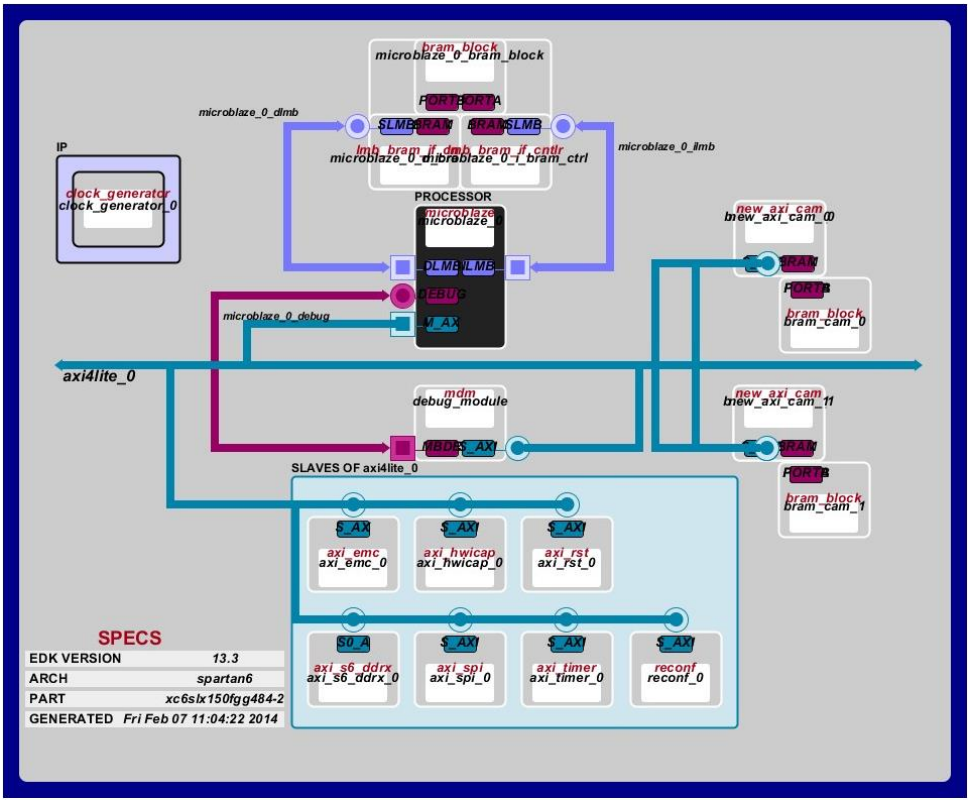


Figure 30 Interconnections and blocks of the HiReCookie systems.

There is a clock generator that manages all the clock signals, two 64KBs cache memories for the MicroBlaze microprocessor, an AXI bus interconnecting the memory controllers (RAM, BRAMs and Flash), two camera controllers, one timer, two peripherals for the Ethernet connection (RST and SPI) and two more for reconfiguration issues (hwIcap and reconf).

For this software version, it is only required to include C code in the program that executes MicroBlaze. Nevertheless, in the next hardware versions, a new hardware accelerator block will be included, and the same distribution will be illustrated with a simpler diagram.

In the current case, the idea is basically to reproduce the online recognition system previously studied but translating the MATLAB code onto C. Once the part of the software running on the embedded microprocessor reaches the recognition command, a function eigenface_s is called with the number of the face to authenticate. When the purpose is to identify against the whole database, the function is the same except for the calculus of the distance, which is executed for every value, as can be seen in Table 9.

`void eigenface_s(uint8 n_face)`
`{`
 	
`sub_face_s();`
`sub_send_s();`
`coord_face_s();`
`coord_send_s ();`
 	
`dist_face_s(n_face);`
`dist_send_s (n_face);`
 	
`}`
	`void eigenface_all_s()`
	`{`
	`uint8 index=0;`
	`sub_face_s();`
	`sub_send_s();`
	`coord_face_s();`
	`coord_send_s ();`
	`for(index = 0; index < 128; index++)`
	`{`
	`dist_face_s(index+1);`
	`dist_send_s (index+1);`
	`}`
	`}`

Table 9 Scheme of the functions in embedded software.

After calling every subroutine of the algorithm, it is possible to send partial information to debug through Ethernet; what is more, in Chapter 4 the verification of this system will be studied due to this property.

It would be worth noting that data types already studied when making operations on MATLAB were respected in order not to overflow any variable. The same tests were checked with the new software, and results agree between different versions.

The first HiReCookie recognition application is completed, although it has a weak point: it needs 8 seconds to finish all the process. Due to this unacceptable time, it will be designed two hardware systems in Chapter 3.IV.

III.3 Towards CUDA

Before going into details of the hardware version, it is presented the contribution made for obtaining the GPGPU recognition system in CUDA language. After studying the main concepts of parallel computing and the FastCUDA philosophy, it was created a first C program for personal computers.

In order to save design time and effort it was employed the already developed and tested embedded software, and therefore a correct version was successfully completed. The time consumed in this case for accomplishing the task was reduced abruptly, down to 19 milliseconds in a laptop and 15 milliseconds in a personal computer. The characteristics of each platform will be seen more in detail in Chapter 5.I

The last code led to a CUDA version of the recognition algorithm. The improvements achieved and strategies followed are a topic for a different project and they are not going to be explained in this research work, although the designing process was supervised by the author of this Master Thesis. Nevertheless, it can be anticipated that software functions turned out to be CUDA kernels – "*subMeanFace*", "*reduceDims*" and "*getDistances*" - and it was essential to know how to access to data in parallel. By coalescing data latency is reduced dramatically and the use of the GPGPU becomes more effective.

This face recognition system was also proven with the same average face, Eigenfaces and database of identities than in the HiReCookie. The results were obtained for two GPGPUs and written down; consequently they would be seen in the comparison of Chapter 5.I.

Finally, the CUDA code for recognition was introduced in FastCUDA tools. This work was proven in a Virtex-6 FPGA with the same input files, but results are not mature enough and they have not been included in the study yet.

In the next section, specific hardware versions for a WSN node will be detailed. It is expected to acquire enough speed up for beating all the other solutions in terms of energy consumed, but first it will be implemented a simple design, and step by step, higher performance will be achieved.

IV. Hardware Versions

As it was mentioned before, the Image Characterization did not fit well with the specification of RUNNER, but looking to the future, the system of the starting point was ported to an up to date AXI AMBA specification. This bus is what the IP core would need if eventually it was required to compare it with the Xilinx package.

With only this change, the frequency supported by the bus, and thus the communication between peripherals of the MicroBlaze will be incremented from 83 MHz up to 100 MHz, and more possibilities would be achievable for novel peripherals.

It was necessary to build a first reconfigurable and simple hardware version before dealing with memories and faster accesses to memory. Once validated the system, the possibilities for accelerating it should be explored.

IV.1 Registered Reconfigurable Version

First, it will be see the architecture of the reconfigurable hardware for face recognition. In addition to the previous design of the system shown in Figure 30, in Figure 31 it can be observed a new peripheral. It will be tested in an ISE version 13.3 project, which makes use of the XPS platform for the inclusion and design of the architecture.

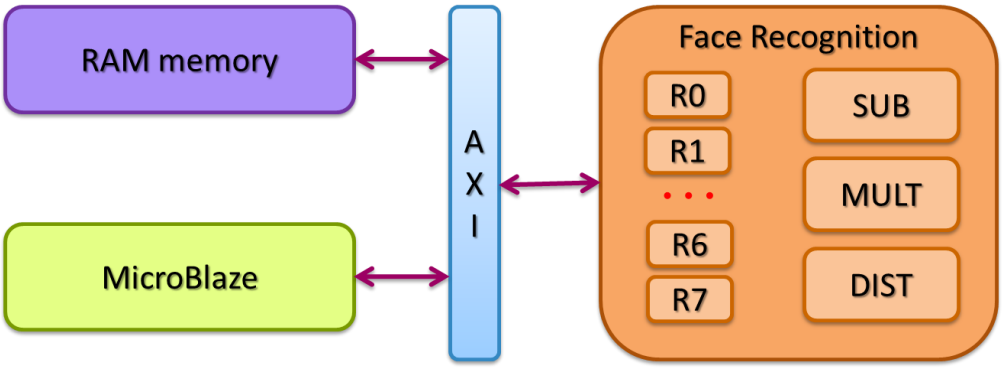


Figure 31 Architecture for a simple hardware.

It was developed a simple peripheral for reconfiguration, called "*reconf*", which together with the "*hwIcap*" employed bus macros in order to define the border of the reconfigurable area with the static one and link signals between them. The architecture used for this block takes advantage of this module for the reconfiguration task.

It was required to define a procedure for synchronising the reception of messages between each one of the sides of the registers. In this application there were 4 input registers and 4 more for output data, although only three input and one output register were necessary.

The output register (number 5) was the one that MicroBlaze should read in order to reach the results of the calculus, whereas register 0 was where MicroBlaze wrote

the operation code, register 2 where data input should be introduced and finally, register 1, where a counter is incremented every time a software loop selected the written line. This code is correct provided the hardware is faster than software, which in principle is guaranteed.

It would be worth noting that this system works with a configurable number of Eigenfaces; the software program in the MicroBlaze can be changed, and the hardware will work with less than 127 vectors if it was desired, without making any hardware change.

The hardware block consist of two finite state machines, whose mission was to detect the command to execute and to change a counter register; and three instantiated block which made the three main known functions. The implementation of multiplications was made with DSP blocks of the FPGA, one operation per clock cycle. The logic inferred for this implementation can be seen in Table 10.

Advanced HDL Synthesis Report	
Macro Statistics	
# MACs : 1	# Registers : 483
9x8-to-32-bit MAC : 1	Flip-Flops : 483
# Multipliers : 1	# Comparators : 4
16x16-bit multiplier : 1	15-bit comparator greater : 2
# Adders/Subtractors : 11	16-bit comparator greater : 1
15-bit adder : 3	32-bit comparator equal : 1
16-bit addsub : 1	# Multiplexers : 293
16-bit subtractor : 1	1-bit 2-to-1 multiplexer : 261
22-bit adder : 2	15-bit 2-to-1 multiplexer : 11
32-bit adder : 1	16-bit 2-to-1 multiplexer : 2
9-bit subtractor : 3	2-bit 2-to-1 multiplexer : 4
# Counters : 1	2-bit 7-to-1 multiplexer : 1
4-bit up counter : 1	32-bit 2-to-1 multiplexer : 13
	32-bit 4-to-1 multiplexer : 1
	# FSMs : 2

Table 10 Advanced HDL Synthesis Report for the registered system.

A summary table with the resources utilization and the percentage of the total amount can be seen in Table 11. This hardware system succeeded in reducing the time down to 7 seconds for the recognition task, mostly due to the acceleration achieved by using hardware multipliers. However, the synchronisation in data makes the hardware wait for the software counter, and this issue should be solved for the last hardware version, by using additional memory or FIFOs. In the following

section it will be shown how Block RAM memory will accelerate the behaviour of the system.

```
Device utilization summary:
Selected Device : 6slx150fgg484-2


Slice Logic Utilization:
 Number of Slice Registers:          485  out of  184304    0%
 Number of Slice LUTs:               630  out of  92152     0%
    Number used as Logic:            630  out of  92152     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:  863
   Number with an unused Flip Flop:   378  out of   863   43%
   Number with an unused LUT:         233  out of   863   26%
   Number of fully used LUT-FF pairs: 252  out of   863   29%
   Number of unique control sets:      22

IO Utilization:
 Number of IOs:                      148
 Number of bonded IOBs:                0  out of   338    0%

Specific Feature Utilization:
 Number of DSP48A1s:                   2  out of   180    1%
```

Table 11 Device utilization summary for the registered system.

IV.2 BRAM Memory Version

The final system should endure an increase in the performance, having certain independence from the software part of the system. In Figure 32 the main modules are depicted. The major difference with the previous model is the appearance of three Block RAMs. These memories are going to make possible to have available the data at any point of the hardware operation. Besides, any delay but the required in processing should appear.

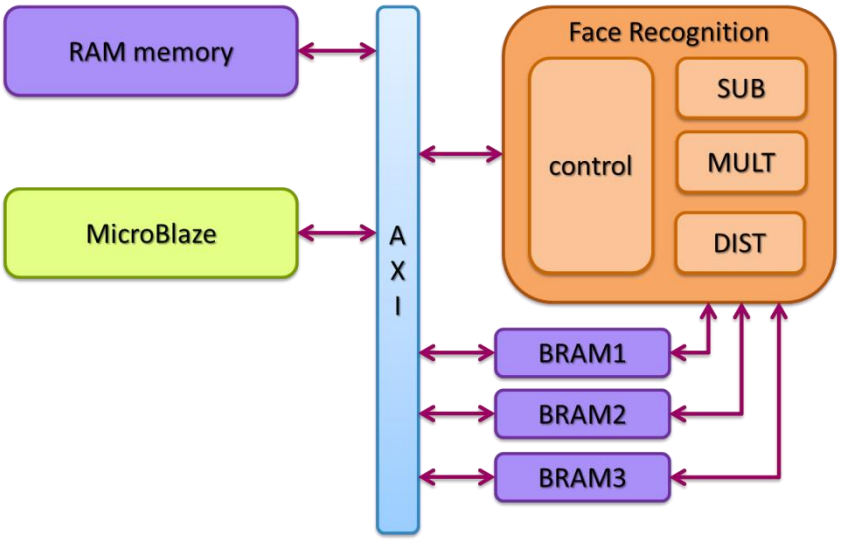


Figure 32 Block diagram for the last hardware version.

The strategy changes, and now, although the finite state machine will also seek for commands, in this case, one single edge is able to launch the whole system. Addresses will be automatically handled by this hardware, and a complex control of the process will be necessary for taking into consideration the latency between blocks. Big multiplexers will select one among several signal buses. In the Figure 33 the block diagram of the system appears.

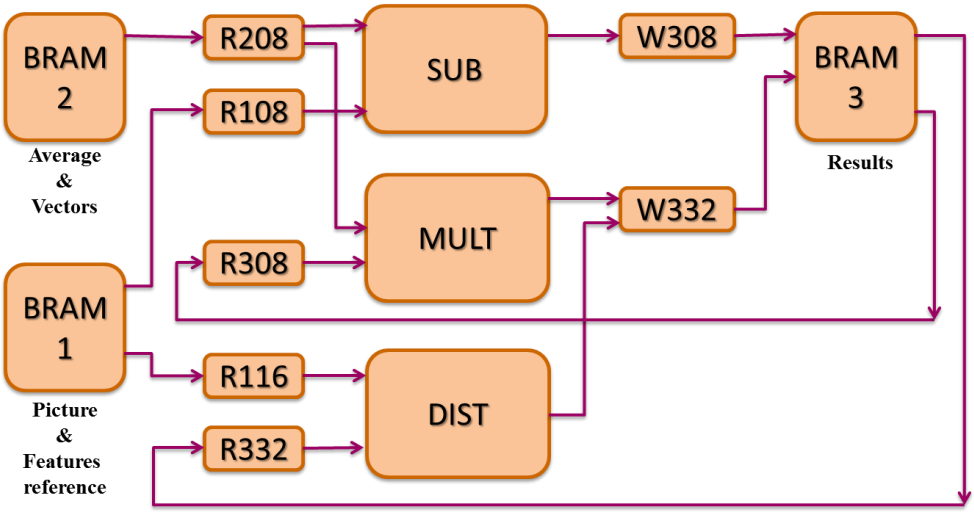


Figure 33 Detail from the inside of the peripheral.

It would be worth highlighting the meaning of the names in the blocks. Each large block about the outsides of the picture represents one block RAM. The BRAM1 is where the picture taken by the camera will be, for the first part of the algorithm, when the average face – placed in BRAM2 – should be subtracted to it. Once this step is finished, the BRAM3 stores the subtracted image resulting. The MicroBlaze then reads that the hardware has finished and asks for a second part.

This time, the previous result with the Eigenvectors should be multiplied and accumulated. The average face will be replaced by the first vector before launching the projection part, and a first coordinate will be calculated afterwards. This process is repeated until the end of the list of Eigenfaces. It should be noted that this is the most computationally expensive step.

One more time, MicroBlaze reads that the second step has finished and launches the third and last petition. The Euclidean Distance is going to be calculated for the configurable value of identity, which is a configurable initial address in the peripheral.

The rest of the blocks are either a block for reading (R) or for writing (W). The first number of their names identifies the BRAM in touch with them while the other two are the size of the data (08, 16 or 32).

Advanced HDL Synthesis Report	# Accumulators : 3
Macro Statistics	32-bit up accumulator : 1
# Multipliers : 2	32-bit up loadable accumulator : 2
18x18-bit multiplier : 1	# Registers : 805
9x8-bit multiplier : 1	Flip-Flops : 805
# Adders/Subtractors : 25	# Comparators : 7
1-bit adder : 1	16-bit comparator equal : 2
16-bit adder : 4	32-bit comparator greater : 5
16-bit subtractor : 5	# Multiplexers : 180
17-bit adder : 1	1-bit 2-to-1 multiplexer : 70
17-bit subtractor : 1	16-bit 2-to-1 multiplexer : 16
18-bit adder : 1	17-bit 2-to-1 multiplexer : 1
18-bit subtractor : 1	18-bit 2-to-1 multiplexer : 1
2-bit adder : 3	2-bit 2-to-1 multiplexer : 9
32-bit adder : 6	32-bit 2-to-1 multiplexer : 41
6-bit adder : 1	36-bit 2-to-1 multiplexer : 1
9-bit subtractor : 1	4-bit 2-to-1 multiplexer : 7
# Counters : 2	8-bit 2-to-1 multiplexer : 30
2-bit up counter : 1	8-bit 4-to-1 multiplexer : 3
4-bit up counter : 1	9-bit 2-to-1 multiplexer : 1
	# FSMs : 7

Table 12 Advanced HDL Synthesis Report for the BRAM system

The hardware inferred from the synthesis process can be seen in Table 12, whereas the percentage of utilization in this case is included in Table 13.

```
Device utilization summary:
Selected Device : 6slx150fgg484-2


Slice Logic Utilization:
 Number of Slice Registers:          738 out of  184304    0%
 Number of Slice LUTs:              1186 out of  92152     1%
   Number used as Logic:            1186  out of  92152    1%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used: 1288
   Number with an unused Flip Flop:   550  out of   1288    42%
   Number with an unused LUT:         102  out of   1288     7%
   Number of fully used LUT-FF pairs: 636  out of   1288   49%
   Number of unique control sets:      22

IO Utilization:
 Number of IOs:                      457
 Number of bonded IOBs:               0  out of    338     0%

Specific Feature Utilization:
 Number of DSP48A1s:                  2  out of    180     1%
```

Table 13 Device utilization summary for the BRAM system.

During the designing process it was found one difficulty that it is worth noticing: due to the large amount of memory in use in this system, a bigger percentage of the available memory was occupied by the face recognition BRAMs. This led to extremely longer paths, and failing timing constraints were found.

Many techniques were tried, from rerouting paths up to re-describe VHDL components. Nevertheless, the answer to this problem was overcome by reducing to the half the BRAM3. The parity bit of the subtraction was not considered any more, by truncating always the last bit and considering it always as 0. The loss in precision was not observable in images and it was admissible, for the reasons exposed in Chapter 4.I. The reduction in this area, together with the attribute "*SIGIS*" for clock and reset signals assured the stability of the data.

The maximum frequency according to the report could be 115MHz, higher enough for the requested speed. The block has been proven at 100MHz, and it is possible to reach even higher frequency by studying some optimization in the VHDL description. However, the results achieved are successful and no more dedication neither effort is considered for this Master Thesis.

Chapter 4. VERIFICATION

The most time-consuming task of the project was undoubtedly the debugging procedure. For hardware systems, simulations are not always accurate enough, or simply the synthesis tool decides to implement a higher level of logic and the behaviour is slightly different from expected and, as a result, the system does not work.

There were applied five levels of debugging process, which are the following sections of this Master Thesis.

I. *Scripts in MATLAB*

There were developed five MATLAB scripts for debugging the numerical result returned through the Ethernet communication. The first script to take into consideration is "*verif_input_txt.m*", and the average face part of this script is shown in Table 14. Nevertheless, the whole script would test also the Eigenfaces, coordinates and two input test images.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This script test the correctness between the input files (*.txt), and          %
%  the previous data obtained with Matlab in the main program base.m   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[stat,struc] = fileattrib;
PathCurrent = struc.Name;
% There is no need for recomputing the algorithm in Matlab except if it
% is the first time for you
YesOrNot=input('Re-execute algorithm in Matlab (=1) or read from files (=0)?');
if YesOrNot==1
   base;                % Re-execute the recognition algorithm
   YesOrNot=1;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
if YesOrNot==1
   file1=u;              % Take the golden reference of the average face
else      % Read the previously obtained average face
   ID1=fopen([PathCurrent '/_mean_raw.pgm'],'r');
   fgetl(ID1);
   file1=fread(ID1, inf, 'uint8');
   fclose(ID1);
end

ID2=fopen([PathCurrent '/mean.txt'],'r');
[file2, counter1]=fread(ID2, inf, 'uint8');
fclose(ID2);

verif1(:,1)=file1;
verif1(:,2)=file2;
verif1(:,3)=file1-file2; % Error between versions

acum1=0;
for i=1:counter1,
   acum1=acum1+abs(verif1(i,3)); % Accumulate errors
   if verif1(i,3)~=0
      % disp([num2str(i),' has failed!']);
      % break;
   end
end
disp(['There is an error of: ',num2str(acum1*100/256/counter1),' % in mean.']);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Table 14 Part of the first MATLAB script.

As can be easily deduced, the program compared two results: the golden reference or secure solution, and the new one. In this case, it is compared the input file, to be sent into the RAM memory, against the calculated by MATLAB. It is worth noting that M and C programs truncate real precision into integer in a different way, what could cause false errors if this difference is not considered.

Similar codes were implemented for the rest of the scripts. After verifying the input file to send to the FPGA, it is actually sent to the client, and with the debugging commands, it can be asked to the FPGA about the stored value in RAM. The returned average face, coordinates and test images are evaluated with

"*verif_input_RAM.m*". However, the Eigenfaces are matched with a different script "*verif_input_vector.m*", mostly because of the long time required for this part of the verification.

After a recognition task, two more scripts could be used. The script "*subtraction.m*" copes with data transformations required to fit 32 bits architecture of the MicroBlaze and peripherals into words of 8 bits. Afterwards, the habitual test is performed.

Finally, the last two parts of the recognition task can be monitored by the script "*rbn.m*". With more than 200 lines of code, this script adapts all the possible implementations of the algorithm, it calls previous scripts when it is needed and it makes possible detecting the exactly place where a fail has taken place.

The accuracy was measured for different cases, by accumulating the absolute value of the error, and expressed as a final percentage. An example of the response in MATLAB when executing the last script can be seen in Table 15. The case shown is the less accurate method, and even so, loses are affordable in this system.

```
>> rbn
Re-execute the calculus of the base? (0=no 1=yes) 0
Is it necessary to recalculate __resJulio_raw.pgm? (0=no 1=yes) 1
Re-execute the calculus of the base? (0=no 1=yes) 1
Number of who is in the input?: 9
Are you testing the registered version (=0) or the BRAM version (=1)?: 1
There is an error of: 39 (equal to 0.0021329 % in the subtraction!
There is an error of: 744285 (equal to 0.00029878 % in the coordinates!
What is the number of the person you are identifying?: (0 = all)50
1 has failed! (with a difference of: 36)
There is an error of: 36 (equal to 0.16786 %) in the Euclidean Distance!
```

Table 15 Returned messages from MATLAB.

The scripts in MATLAB contributed significantly to the debugging process, and they measure loses in accuracy. However, a very fast and intuitive debugging method is explained in the next section.

II. Image Results

As well as analyse the values returned, it is possible to watch in real time the image just processed by the HiReCookie. To carry out this action, there were two types of tests. In the first one, a normalised imaged is introduced as a test image through Internet, while in the second possibility, the picture is actually taken by using a real camera of the WSN platform.

Figure 34 shows the results for an ideal comparative, as if there were a previous detection part working in the system. The greyscales values of the right side of the picture are the coordinates of individual 1 and its distance with subject 55. After a linear transformation in order for having values between 0 and 255, the distance was scored as 55, what means the images are different. To understand this clearly, it should be known that usual values for different people are about 40, longest distances around 80 and matches below 10.

Figure 34 Subtracted image of individual 1.

In Figure 35, a real picture is shown. In this case the person is not from the database, although she scores 34 when comparing with subject 49. It would be worth noting that a black pixel is a 0 in luminance, and because of that, the background remains after the subtraction. Nevertheless, Eigenfaces cannot explain those pixels and they are not taken into account, as if they were in black. So if a rebuilding of this faces was executed, the background would be black as well.

Figure 35 Unknown person, but within the usual distance between no matches.

It was performed a second test with the real subject 9 in front of the camera, and he was a perfect match with himself and rejected from other identities of the database. The Figure 36 shows the observable results of this test.

Figure 36 Original picture, real subtracted and ideally subtracted.

Although it can be hard to center the head accordingly to the desired position, illumination and pose, the algorithm managed to identify the subject. It has been demonstrated the feasibility of the recognition system for cooperative environments, without the necessity of using a detection algorithm.

III. Simulations

The two hardware systems were simulated before doing the synthesis. There are nine simulations for the BRAM version – subtraction, projection, distance, the global system, three reader blocks and two more writers - and one for the whole registered version. It is considered that a full explanation of every simulation would require too much space, so only one of them is presented here, the one corresponding to the global simulation of the BRAM system. Figure 37 is the subtraction part of the algorithm, Figure 38 the projection onto the subspace and Figure 39 the Euclidean Distance. The three captures of the simulation have blocks in groups of colours, and similar colours between images imply the same type of operation.

In the first screenshot, for instance, in orange are selected the inputs - "*start*" and "*valid*" signals, "*address*", "*data_in*" and "*data_out*" –of the red part, which represents the core of the operation – a subtraction triggered by "*do_it_s*", between "*data_in_s*" and "*refer_s*" with a valid result indicated by the signal "*write_s*"and shown in "*data_out_s*" – while in purple the results are written in the correct address of BRAM3 – "*start*" selects the correct "*init_addr*" and "*valid*" starts putting "*data_in*" in "*data_out*" with the address "*bram_addr*".

Similar pattern is followed for the other two parts of the simulation. In all cases, yellow shows the external addresses, in red the operation taking part and in blue is displayed a counter of the progression of the process.

Once studied one case of the ten simulations performed during this research work, it will be seen in the next section when it was necessary to use an embedded oscilloscope.

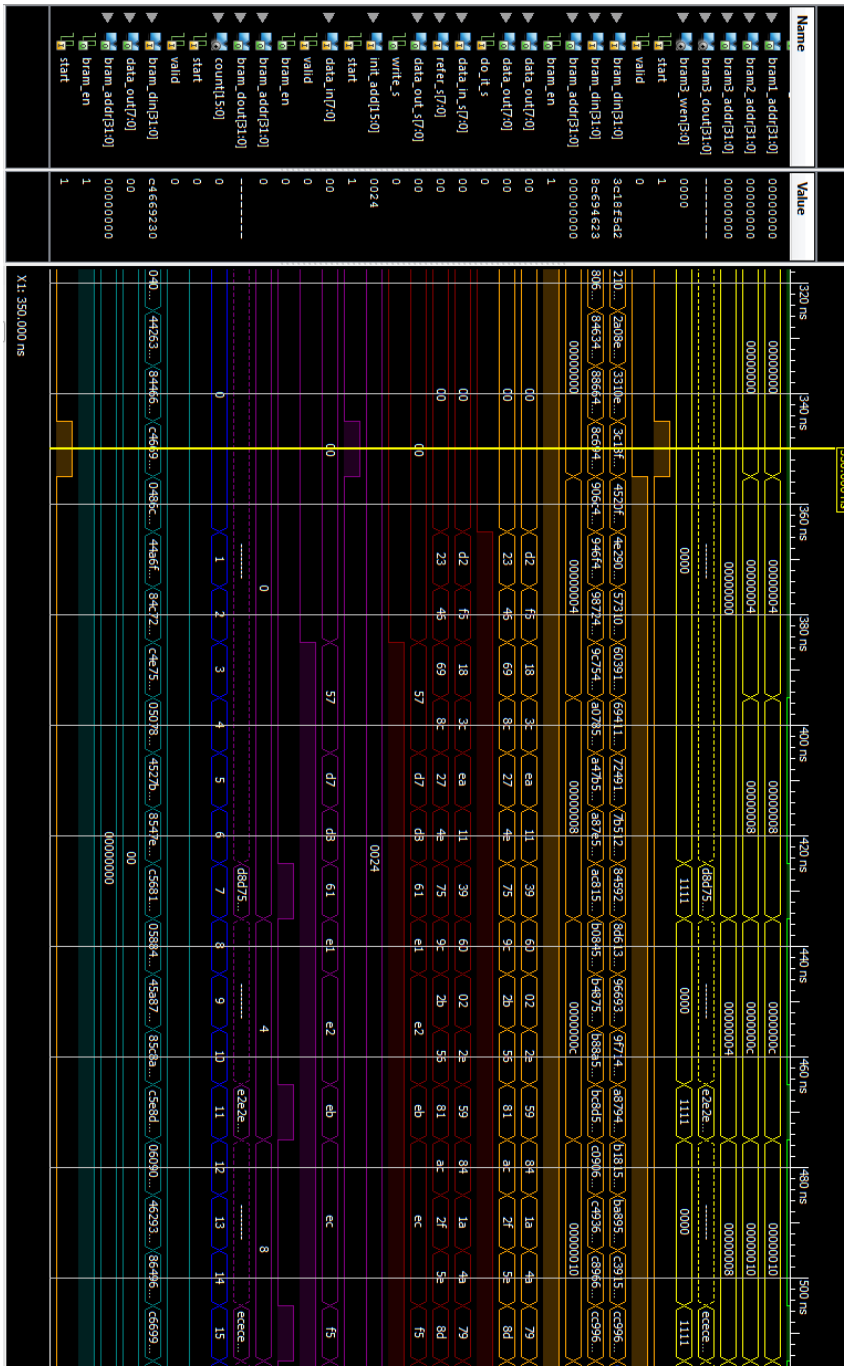


Figure 37 Simulation of the subtraction.

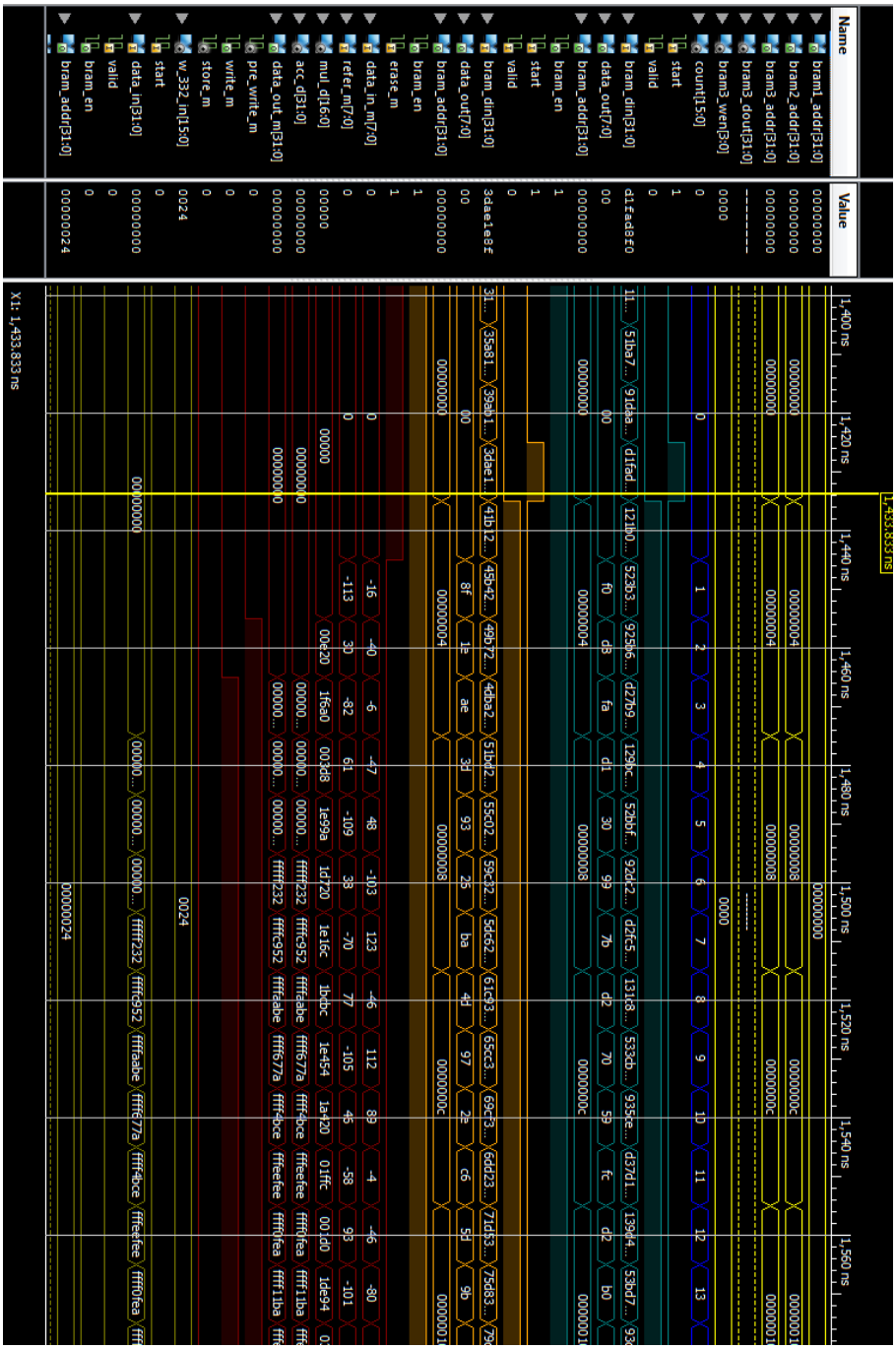


Figure 38 Simulation of the projection.

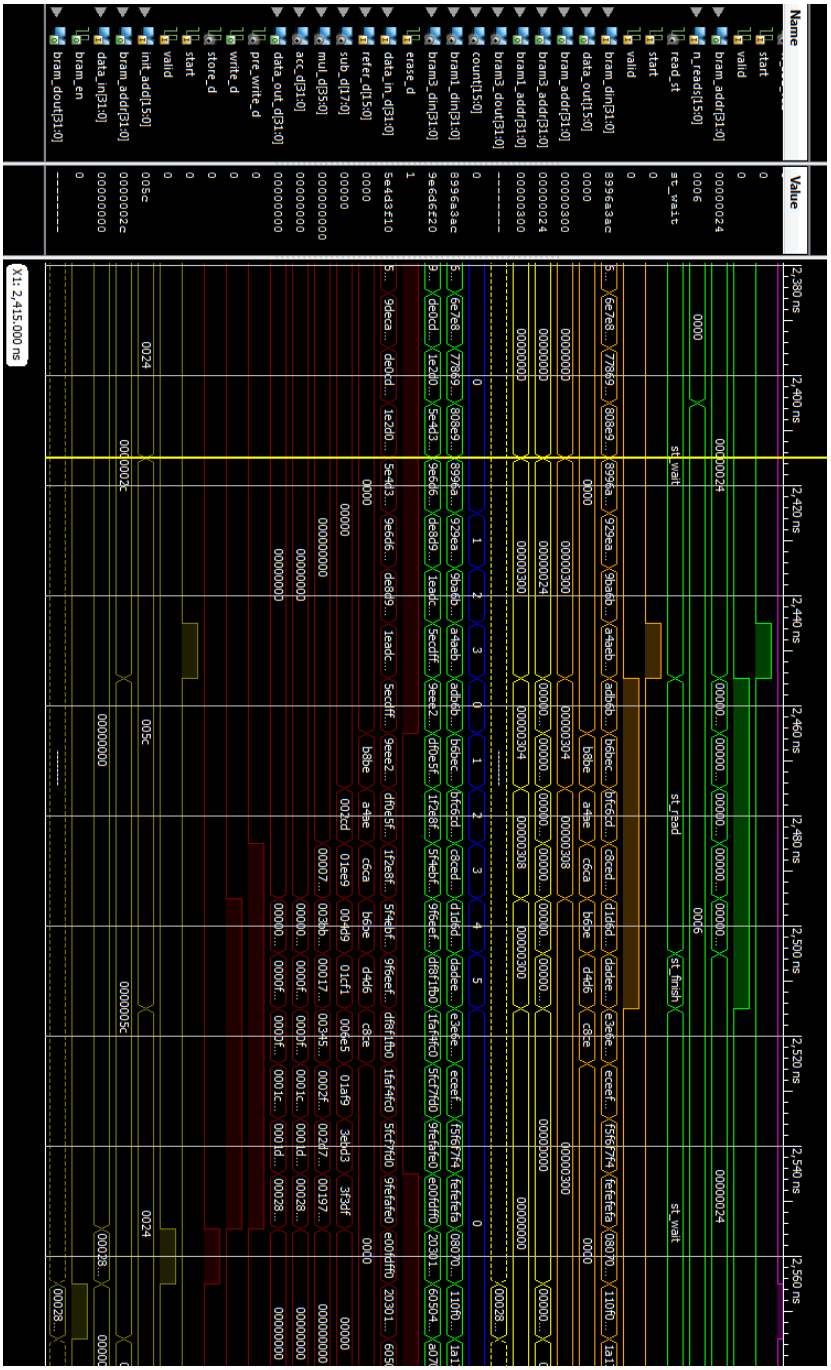


Figure 39 Simulation of the distance.

IV. ChipScope Analyzer

When neither the scripts, nor images or simulations helped to debug an unexpected behaviour, it was necessary to use ChipScope Analyzer. This program acts as it was an embedded oscilloscope for the FPGA. The new horizon that this tool opens is huge, making possible – although not always so easily – to find failures where before it was unthinkable.

Nevertheless, every time the user needs to change a signal, the whole design should be re-synthesised, and this process used to take about 40 minutes for the actual system. So it only will be employed if there is not any other alternative for solving the issue.

ChipScope was used in this project for fixing one last error. Between simulation and results obtained through the debugging socket, there was no match if the algorithm was completely executed, but they do agree if only part one or part two were launched. It was though that some optimisation in the implementation process could be changing the final system, but only with an embedded oscilloscope this phenomenon could be observed.

It was added to the design a new peripheral, which ChipScope will use for reproducing the values of the signals on the program waveform. Finally, it was found that this behaviour was due to the apparition of latency in the process of reading and writing in the last part of the algorithm. This delay did not appear in simulation, and it was caused by the implementation tool, when it decided to put two registers instead of one for improving performance.

The system was finally fixed by changing the VHDL description in the part of the system involved, and the correctness was successfully verified. The capture of the results can be seen in Figure 40, where no extra latency is added when comparing to the initial system.

With the system completely verified at different levels, some tests together with Nao Robot are going to be related in the next section.

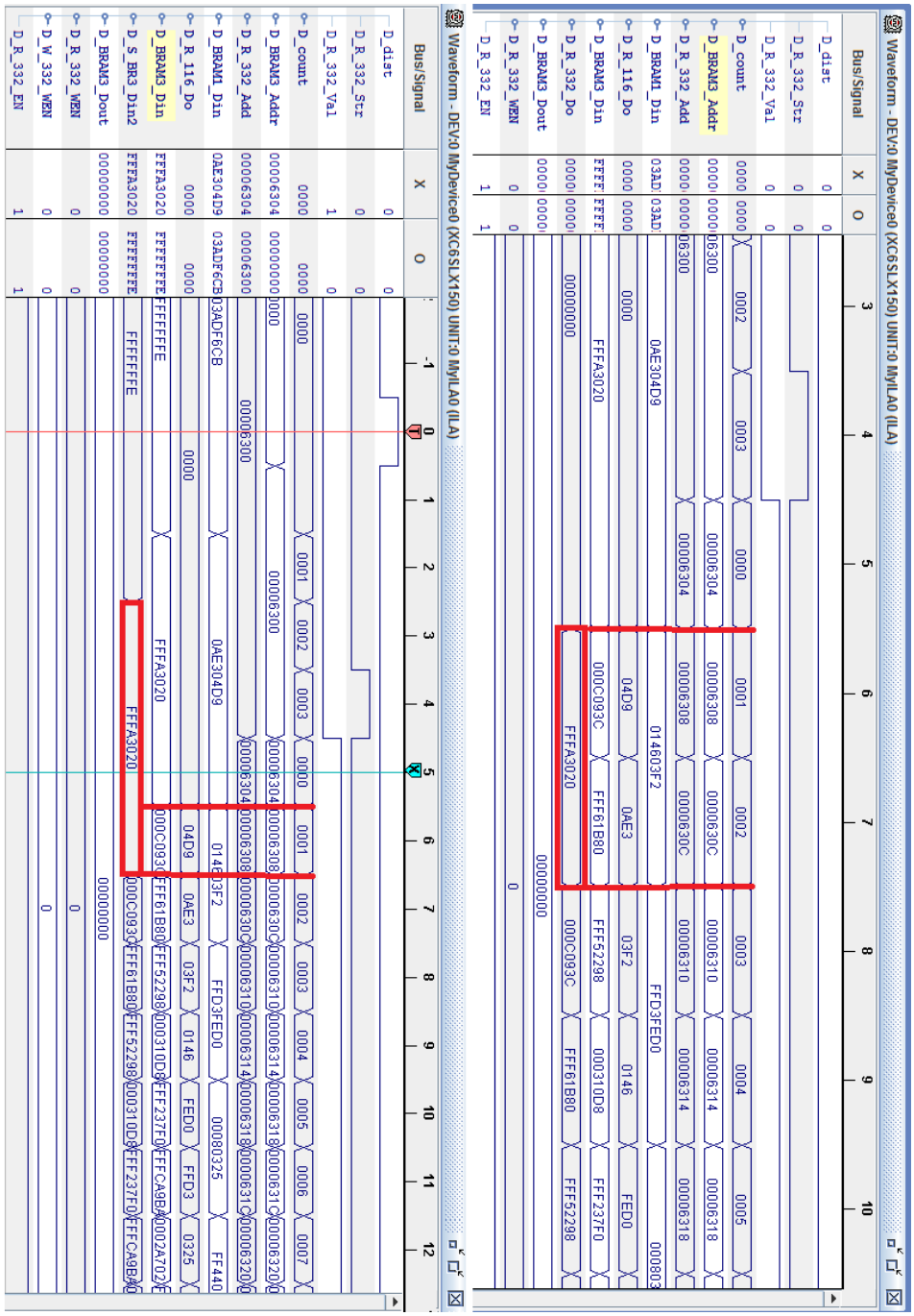


Figure 40 ChipScope Analyzer.

V. *Nao Robot*

Up to three tests were carried out using Nao Robot and the WSN node. The robot was a property of a research group in Castilla La Mancha University, and thanks to previous collaboration they lend the robot to CEI for these three verifications. The first one was to check the correct interpretation of the commands described in Chapter 3.III.1.

Through Internet, petitions were sent by the robot and the HiReCookie answered all the possible combinations. Every one of the answers had a response from the part of Nao; for instance Nao sent greetings to the person identified, and asked to go unknown ones, it avoided imaginary obstacles or navigated in an unknown environment. These performances were made with Choregraphe, the toolkit from Aldebaran, in a pretty intuitive way of programming (see Figure 41).

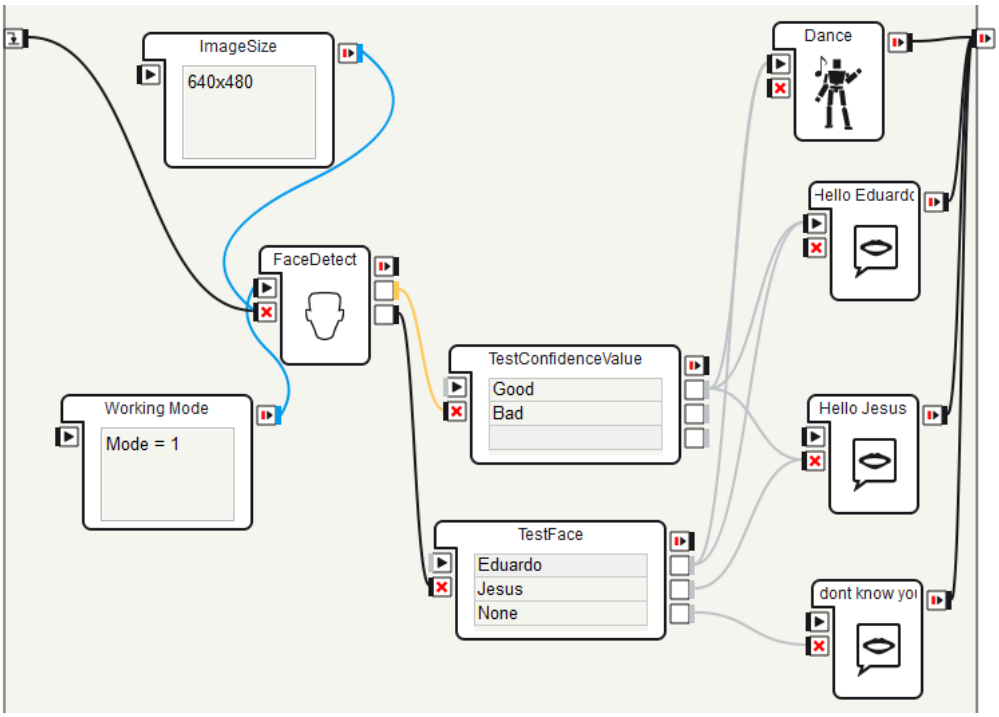


Figure 41 Choregraphe tool for developing Nao performances.

It was a remarkable result that, by dedicating little time, the communication interface was successfully integrated within the first attempt. The work was quite good coordinated, especially considering that each research group worked from its location. It was not necessary to be in the same place robot and platform; in fact the

distance between them was 200 kilometres, and the performance was followed via teleconference.

The second test accomplished was in the CEI laboratory. Nao came to Madrid in order to prepare a final demonstrator. A local network was established, between robot, Choregraphe software of the robot and the visual cortex. This time, real implementations of the recognition system were prepared in the HiReCookie. The robot can be seen in Figure 42 in the library of CEI, during the trial.

Figure 42 Autonomous robot Nao, from UCLM.

Last, a final review with Spanish Authorities took place in InetSiS offices. The resolution was truly positive, succeeding in passing the audit.

Chapter 5. COMPARISON BETWEEN ALTERNATIVES AND FURTHER WORK

As the work is almost already done, it only remains to present the results achieved and study the possible impact this could bring for not finished applications. In this chapter, a summary table with the main features and results of each solution is included. Due to the flexibility of the last hardware version, it is proposed to change some parameters to develop other systems similar to the included in the study. The last part of the chapter is dedicated to open scopes related to this study: on the one hand, the face detection for a fully operative recognition system; on the other hand, the incoming results of FastCUDA tools.

I. Comparison between Different Alternatives

Platforms from different nature have been employed in this research work, and the results achieved with them are presented in Table 16 and discussed as follows:

- A Laptop Asus. It has a quad-core microprocessor, and a portable version of the GPGPU GeForce GT 540M. Therefore, it was used for analysing two versions of the recognition algorithm, one in CUDA and another in C language. The performance of Laptop Asus in terms of time was very promising, although if the energy is what really matters, it must be said that this platform was beaten by the FPGA.
- Personal computer DELL. It has a previous generation of a desktop quad-core microprocessor. Nevertheless, the GPGPU that carries this platform is one of the most powerful – and expensive – of the market. The personal computer was used for contrasting the previously obtained results with a system that may have a more computing power. The time required to do the task with this device is even lower than with the previous one, but when it comes to energy, the personal computer becomes the real loser, not to speak about the cost of the platform.
- FPGA Spartan-6 in a High Performance Wireless Sensor Network node. The core of this Master Thesis. By using the available resources suitably, this core has beaten all the previous systems. It would be worth noting that its possibilities are not over yet, and further work is possible, although such research overtakes the

already achieved purpose of this Master Thesis. In Chapter 5.II.1 it is presented how it would be possible to increase the speed of the last solution even more. Moreover, it should be considered that this platform has the ability to switch completely off the FPGA if processing tasks are not required. This specific feature makes this platform far the best option of all the considered.

Device type	Version	Main features	Average power (W)	Time (ms)	Energy (J)	Price on Market (€)
Laptop Asus	CPU 2.20GHz i7-2670QM	6GB RAM 6MB cache	45	19	0.855	120
	GPGPU 1.33 GHz GeForce GT 540M	2GB 98 Cores	35	4.2	0.147	92
PC DELL	CPU: 2.80GHz i7-870	4GB RAM 8MB cache	170	15	2.550	140
	GPGPU 1.15 GHz Tesla G2075	4GB 448 Cores	238	2.1	0.500	1470
FPGA Spartan-6 XC6SLX150	CPU 100MHz MicroBlaze	32 MB RAM 64KB BRAM	1.31	8000	10.48	116
	Hardware Registered	3 input 1 output	1.33	7000	9.31	116
	Hardware BRAMs	32 KB x2 + 32 KB	1.35	30	**0.0403**	**116**

Table 16 Energy and price comparison of the systems

Only the processing unit of each device has been considered. However, for the four first results, there have been taken into account tables of average power consumption - thermal design power in the i7-2670QM, average power consumption while computing in the others – found in manufacturer's documentation, while in the remaining three cases – the HiReCookie platform, which has independent power islands - a real oscilloscope was used for measuring the voltage in a Shunt resistor, after amplified with an INA333 instrumentation amplifier.

Even though only the processing units of the devices have been taken into consideration, the whole device will be consuming energy in a real system. It has been measured that, in average, the 85% of the total amount of energy consumed in the WSN node case was due to the FPGA. The large amount of peripherals in a computer suggests that this percentage is lower for these devices, what would make even higher the difference if it was considered the whole machine.

However, it was not possible to measure this data in this research work, so the lector is invited to study the relationship between the percentages of energy consumed in computers by the processing unit to the whole device itself, in comparison with the known 85% for the FPGA's processing unit of the HiReCookie platform.

II. Flexibility of the BRAM Version

The possibilities of exploitation of the last hardware system were not over after all the improvements explained before. The description of the VHDL code was highly configurable, and it is desired to expose two actual possibilities, envisaged during the designing process. The first one is related to the possibility of accelerating even more the system, while the second deals with area reduction.

II.1 More DSPs (Multipliers)

The current solution utilised two DSP48A1s, one 9x18bit for multiplying and accumulating the projection of every coordinate of the system, and one 18x18bit for subtracting, multiplying and accumulating the Euclidean distance. One possibility for speeding up the system could be the following:

- To split the BRAM1, BRAM2 and BRAM3 in, for instance, 4 BRAM each one, with independent output ports and addresses, or even one BRAM each but with multiple outputs and one single address port with the ability of pointing to four data in parallel.

With this approach it would be affordable to have – following the example – four DSPs for every duty. Since the projection stage is the most time consuming task – more than 99% of the time of the whole process - it is expected to reduce up to four times the interval required and also to reduce the energy consumed in the end.

The reason why this will not lead to an observable increment of the energy is that DSPs are there but not always in use. In Figure 43 it can be observed that in a clock region there are up to 8 DSPs, so the example would be the limit for a simple reconfigurable hardware. If it would be desired to have better performance than four times less time, other approaches, such as the Virtual Borders of the DREAMS tool, should be considered.

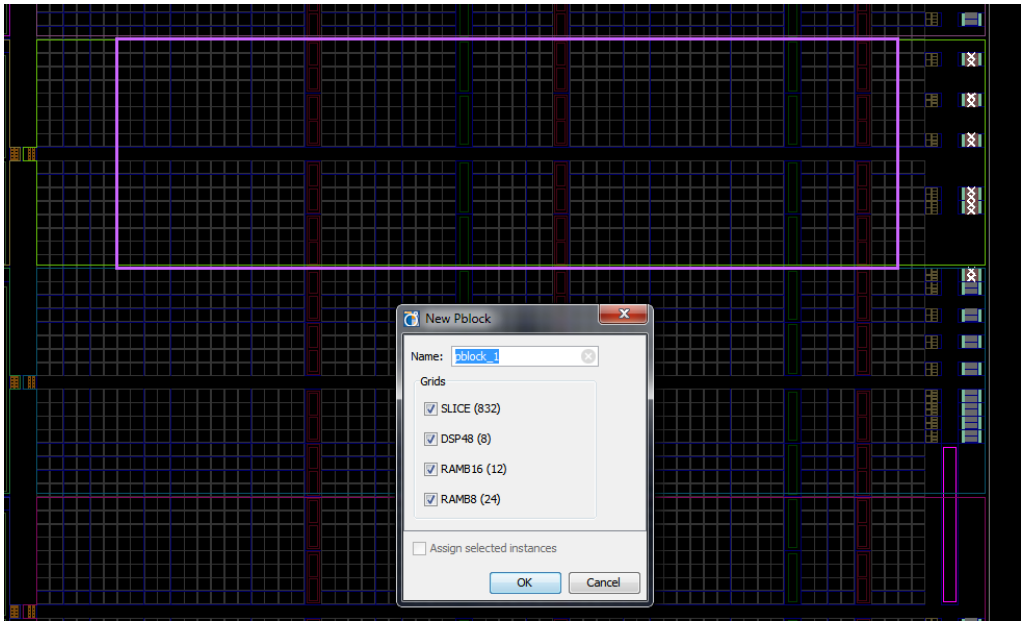
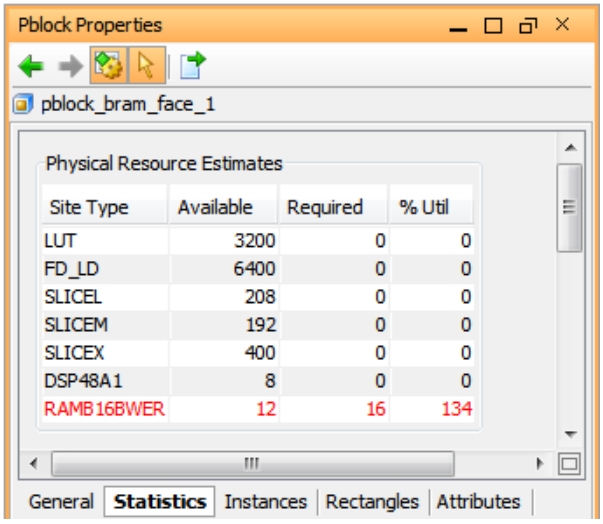


Figure 43 Resources of a clock region.

II.2 Less BRAM Utilization

As it was mentioned in Chapter 3.IV.2, too much BRAM could cause timing problems. In addition to this fact, the amount of RAM blocks of a clock region is limited. The easiest solution to this problem is to use less BRAM memory, for instance, 16 KB instead of 32KB. This will only imply for the system, as it is designed, to call two times every hardware function, by writing the corresponding value into the software accessible registers. The blocks that manage the memories are thought to deal with this situation, so it would not be complex to reduce resources utilisation – up to a point – by using this technique.

In Figure 44 it is shown the available resources in a clock region in comparison with the BRAM block. It would be highlighted that in every clock region it is only available a total amount of 24KB of BRAM. Once again, if reconfiguration is going to be made with bus macros, the amount of memory should be reduced as it has been explained, or another approaches as the Dream Tool should be employed.



Site Type	Available	Required	% Util
LUT	3200	0	0
FD_LD	6400	0	0
SLICEL	208	0	0
SLICEM	192	0	0
SLICEX	400	0	0
DSP48A1	8	0	0
RAMB16BWER	12	16	134

Figure 44 Required resources and available in a clock region.

Due to the interest in using this hardware system together with future J. Valverde's PhD Thesis, he is encouraged to use the results here exposed. Because of his approach limitation of addressing only 4kB of BRAM, it is proposed to use 1KB BRAM blocks for input memory blocks, and 2KB for the output BRAM.

III. *Face Detection Incorporation*

Thanks to the close collaboration with Santiago Muñoz, many results related with CUDA code were obtained. He took advantage of the recognition software for developing the first CUDA code of the recognition system. Afterwards, he designed a face detection system running in GPGPUs; it would be a good idea to include his code to the HiReCookie platform in order to complete the feedback circuit.

As it was mentioned in Chapter 2.III, Eigenfaces can be used for detection, but it is not a very smart procedure. There would remain two alternatives, either to include a software embedded program, or to implement a hardware detection system. The Viola-Jones algorithm used in CUDA by Santiago is quite complex to be executed in real time by MicroBlaze, so it only will be considered the hardware version.

However, there is a previous work that matches with the aim proposed; Matai et al. [54] have designed and implemented an FPGA-Based Real-Time Face Recognition System, based on Viola-Jones algorithm for detection, and Eigenfaces for recognition. The only contribution that can be made is to implement this system using FastCUDA

tools, and it is left as a future line because at the present it is at the development stage yet.

IV. FastCUDA Tool Work

As FastCUDA project was taking shape, it was necessary to contribute in some of its stages, such as the research of algorithms to be implemented, and the adjustment of the tool given by the partners. This last assignment was part of the final deliverable of the work, and some syntheses of CUDA kernels were made for including results in the report.

For merely having an idea of the complexity of this work, it would be said that it required using a virtual machine with a Linux distribution, in order to compile a Graphical User Interface based in Java, which used scripts (BASH, PERL, TCL and more) for translating CUDA code to SystemC. The SystemC is included afterwards in a Vivado Project that makes possible to unroll loops, and finally, different implementations for FPGA resulted.

During the process, some reports were parsed, and unfortunately, the files were different between Vivado versions, so this had to be fixed. In the end, the tool started producing correct VHDL codes to be included as blocks of the FastCUDA system. A screenshot of the Tool is shown in Figure 45.

This chapter finishes with this last section, in Chapter 6 will be seen the conclusion and futures work related to this Master Thesis, as well as the publications, presentations and dissemination.

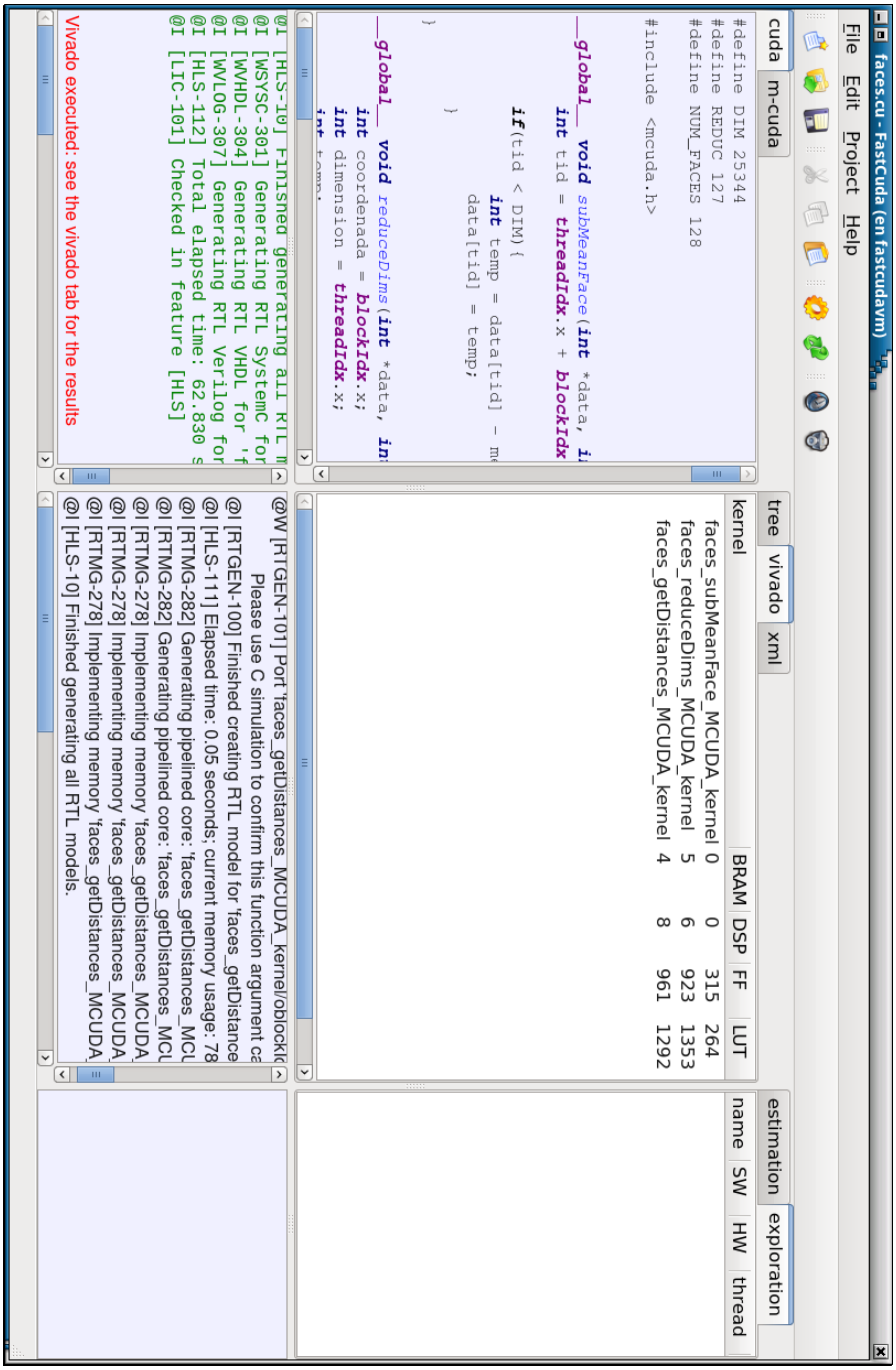


Figure 45 Screenshot of the FastCUDA tool.

Chapter 6. CONCLUSIONS AND FUTURE WORK

After the progress achieved in RUNNER, commercial robots will take advantage of this research work: Aldebaran-Robotics have announced the intention to inherit the advances in their – still developing nowadays - future robot ROMEO, shown in Figure 46.

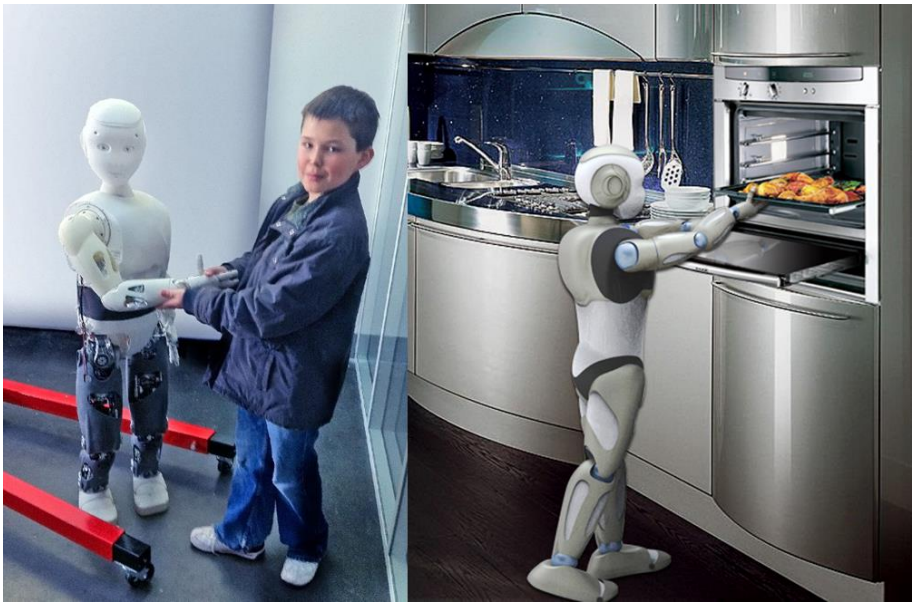


Figure 46 Left side: prototype of ROMEO, right side: simulation of the future application.

These new advances in the area of vision systems together with robotics can be exploited with goodwill purposes. As it was studied by Avvenuti [55] and at the same time proposed by Aldebaran-Robotics in their Web Site [56], WSN and robots can help people with difficulties, such as elder people suffering Alzheimer Disease, or even kids with Autism, on the process of learning and establishing more successful social interactions with other people. Voice recognition and interpretation, together with object/face recognition, are the key factors for the inclusion of android-robots for domestic chores.

On the other hand, great efforts are being made by research groups and companies in order to facilitate the use of FPGAs among software developers, with the purpose of taking advantage of the better efficiency and parallel computing of those devices over microprocessors. As it has been seen, these two branches have been the backbone in this Master Thesis, so the conclusions they have brought are detailed in the following section.

I. Conclusions

- In this Master Thesis it has been established the specifications for developing a face recognition system in a variety of platforms at the same time: MATLAB running in a personal computer, C code in an embedded microprocessor (MicroBlaze), a simpler reconfigurable hardware for an FPGA-based platform, a flexible hardware for higher performance, and finally a C and a CUDA code for a laptop and a personal computer.
- After having studied the opportunities that recent solutions can bring to the design of a novel face recognition system, it has been proven the feasibility of the algorithm chosen with a first version using MATLAB tools. However, the scope of the applicability of the CUDA code, being capable to port it to most of the architectures that have been tested throughout this project, once the flow proposed in FastCUDA is fully developed and settled, gives a high opportunity to a one-effort-fits-all approach for SW, GPU and HW based systems.
- In addition to the image processing task required for building a database of faces and a set of identities, two applications were developed in order to verify the main concepts presented in the theory of the Eigenfaces algorithm.
- One software and two hardware versions of a reconfigurable face recognition system have been implemented in a real FPGA-based WSN node, followed by consecutive processes of verification and testing the correctness of the different versions. Besides, it has been given the required information for designing two more systems for both a laptop and a desktop, and supervised their development.
- Finally, it has been created a verification system that facilitates the debugging process when designing a new recognition system, as well as it tests the numerical precision of every solution by comparing the results with a golden reference. The results show to be uneven, in the sense that some implementation details which might look like second order decisions, have a critical impact in performance, so a detailed knowledge of the architecture, and what is accelerated and what is not, is required.

- The results of all the systems were presented in a summary table. It was obtained that the hardware version requires 3.65 times less energy than the best of the other solutions. Nevertheless, the methodology and design flow followed with this type of solutions is much more complex and more time-consuming.
- As an extension to the study, it has been explored novel approaches such as MATLAB HDL Coder, Vivado HLS and FastCUDA tool flow. These techniques will make easier and faster the task of designing hardware, while bringing software programmers the possibility of taking advantage of FPGAs, and therefore it is expected a big expansion in this direction.

II. Future Work

Taking into consideration the performance obtained in the recognition system implemented in hardware, the speed could only be increased by following these two methods, which inevitably would increment as well the resources utilisation:

- On the one hand, it would be interesting to split the last recognition system designed in hardware into multiple parallel processing elements in order to analyse the achievable performance.
- On the other hand, it is expected to have an improvement in the speed if the RAM memory is accessed in a burst manner, filling the internal BRAM memory at higher rate.

As it was explained before, it would be interesting to include the first step of the full recognition process, the face detection, to obtain multiple normalised faces, However, it would require to be executed in real time.

- It is possible to use the Eigenfaces algorithm with the purpose of detecting faces in the HiReCookie platform. However, it seems to be more practical to implement a Viola-Jones algorithm and shift different windows along the image.
- It is desired to take advantage of the FastCUDA tool flow in order to create new hardware blocks. One last version made by this automatic process would enrich the proposed comparison, having shared memory and coalesced accesses.
- It was proposed to use a new set of images, incrementing the number of them, and including different illumination conditions and poses. Moreover, it should be considered to develop some script for automatically normalise the pictures.
- In order to make the results achieved more attractive, it is possible to substitute the actual client, based on a terminal black window, for a user-friendly Graphical

User Interface application, which would show images in real time by the simple action of pressing buttons.

In the upcoming future, professors could have a device similar to Google Glasses for having a daily register of the students. Maybe this dream is closer after this research work.

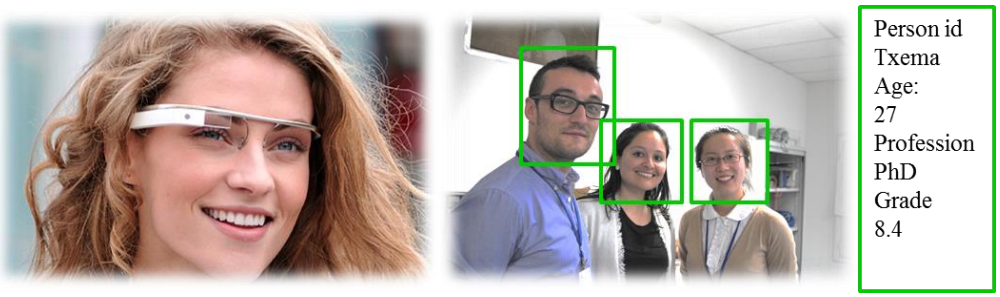


Figure 47 New scenario for a recognition system.

III. *Publications and Presentations*

A certain number of presentations and documents have been produced into the framework of this Master Thesis:

- During the performance of this work, was maintained a direct collaboration in the creation of a starter guide for the use of the reconfiguration capabilities of the HiReCookie platform. The resulting document is called HiReCookie Reconfiguration Guide [57], and it is expected to serve as the manual of how to use ISE tools for reconfiguration, for future students at CEI.
- During the Posters Session of the Annual Meeting 2013, one demonstration took place in the Centre of Industrial Electronics. Thorough this opportunity, it was presented the progress on the RUNNER project, and visitors were able to use the platform, take pictures with it and ask any question they had about the work. Moreover, there was collaboration with the presentation of one of the conferences held in this event.
- A second demonstration was organised for the "*Seminario Anual de Automática, Electrónica Industrial e Instrumentación 2013 (SAAEI'13)*". In this case, the attendants could observe the functioning of an application that uses WSN nodes in security areas. The application monitored with cameras the scenario when an event was triggered by a luminosity sensor.

- In order to verify the correct performance of RUNNER in Spain, an external audit was carried out by *"Centro para el Desarrollo Tecnológico Industrial (CDTI)"*. In the meeting was necessary to explain to National Authorities the details of the work completed so far, as well as it was demonstrated the operation with an integration of the recognition system and the robot Nao.
- On November fourth, 2013, a Monday Seminar was held by the author of this Master Thesis. The purpose of such meetings is to present to workmates and professors the existing achievements, difficulties found and future of the undertaken projects.
- It is scheduled to write a paper with the methodology and results of this work, together with Santiago Muñoz, after the redaction of this Master thesis. Hopefully it would be accepted and presented, but it is still in the first stage of the process of creation.

IV. Dissemination

This Master Thesis has certain continuity with current works, and it will also lead to future final degree projects:

- The Stereo Matching block was designed by Federico Pérez in software and hardware, using respective versions of the Face Recognition module as template. For taking advantage of the hardware acceleration, he will start from the last version of the face recognition block in order to use BRAMs.
- As it was mentioned above, GPGPUs versions of the recognition and detection systems have been developed by Santiago Muñoz, what will result in his final degree project. The author of this Master Thesis led his first steps with the basis of the algorithm and continued supervising his work all along its development.
- One extension of this work is the final degree project called: "*Motion detection with 3D cameras for a collision warning system*". This work will study the motion detection algorithms in existence, and will implement a new block for the HiReCookie platform with the purpose of detecting and avoiding projectiles.
- Juan Valverde in his PhD thesis is working on a novel reconfigurable architecture that will face the triangle: consumption, computing power and reliability. This design is being implemented on HiReCookies, and the Face Recognition hardware has been provided for inclusion as an example of applicability.
- On December second, 2013, a final review of the project RUNNER took place between the members of the council. In the meeting it was concluded that every group succeeded in accomplishing their respective task. What is more, Aldebaran

will launch a novel generation of robots taking advantage of the satisfactory contributions of the consortium.

REFERENCES

[1] Intel Corporation, “www.intel.es,” January 2014. [Online]. Available: http://www.intel.es/content/www/es/es/processor-comparison/processor-specifications.html?proc=75133.

[2] MediaTek Inc, “www.mediaTek.com,” November 2013. [Online]. Available: http://www.mediatek.com/_en/03_news/01-2_newsDetail.php?sn=1127.

[3] MediaTek Inc, “www.mediatek.com,” January 2014. [Online]. Available: http://www.mediatek.com/_en/Event/201307_TrueOctaCore/tureOcta_.php.

[4] K. Su, J. Li and H. Fu, “Smart City and the Applications,” in *International Conference on Electronics, Communications and Control (ICECC)*, Zhejiang, 2011.

[5] G. Cardone, P. Bellavista, A. Corradi and L. Foschini, “Effective collaborative monitoring in smart cities: Converging MANET and WSN for fast data collection,” in *Kaleidoscope 2011: The Fully Networked Human? - Innovations for Future Networks and Services, Proceedings of ITU*, Cape Town , 2011.

[6] M. Sen, A. Dutt, S. Agarwal and A. Nath, “Issues of Privacy and Security in the Role of Software in Smart Cities,” in *International Conference on Communication Systems and Network Technologies (CSNT)*, Gwalior, 2013.

[7] L. Wei, C. Ming and L. Mingming, “Information Security Routing Protocol in the WSN,” in *Fifth International Conference on Information Assurance and Security*, Xian, 2009.

[8] Y. Wang, X. Wang, B. Xie, D. Wang and D. Agrawal, “Intrusion Detection in Homogeneous and Heterogeneous Wireless Sensor Networks,” *Transactions on Mobile Computing*, vol. 7, no. 6, pp. 698 - 711, 2008.

[9] R. Heitmeyer, “Biometric identification promises fast and secure processing of airline passengers,” *ICAO Journal*, vol. 55, no. 9, pp. 10-11,27, 2000.

[10] Cognitec Systems, “www.cognitec-systems.de,” 2014. [Online]. Available: http://www.cognitec-systems.de/facevacs-videoscan.html.

[11] A. Gallego, J. Mora, A. Otero, R. Salvador, E. de la Torre and T. Riesgo, “A Novel FPGA-based Evolvable Hardware System Based on Multiple Processing Arrays,” in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW)*, Cambridge, MA , 2013.

[12] M. Lombardo, J. Camarero, J. Valverde, J. Portilla, E. de la Torre and T. Riesgo, “Power Management Techniques in an FPGA-Based WSN Node for High Performance Applications,” in *International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, York, 2012.

[13] J. Ascenso and F. Pereira, “Lossless compression of binary image descriptors for visual sensor networks,” in *International Conference on Digital Signal Processing (DSP)*, Fira, 2013.

[14] J. Valverde, A. Otero, M. Lopez, J. Portilla, E. d. l. Torre and T. Riesgo, “Using SRAM Based FPGAs for Power-Aware High Performance Wireless Sensor Networks,” *Sensors*, pp. 2667-2692, 2012.

[15] M. Kaddachi, L. Makkaoui, A. Soudani, V. Lecuire and J. Moureaux, “FPGA-based image compression for low-power Wireless Camera Sensor Networks,” in *International Conference on Next Generation Networks and Services (NGNS)*, Hammamet, 2011.

[16] H. Fu, H. Ma and L. Liu, “Robust Human Detection with Low Energy Consumption in Visual Sensor Network,” in *International Conference on Mobile Ad-hoc and Sensor Networks (MSN)*, Beijing, 2011.

[17] T. He, S. Krishnamurthy, J. A. Stankovic, T. Abdelzaher, L. Luo, R. Stoleru, T. Yan, L. Gu and G.

Zhou, “VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance,” *ACM Transactions on Sensor Networks,* vol. 2, no. 1, pp. 1-38 , 2006.

[18] J. Portilla, T. Riesgo and A. De Castro, “A Reconfigurable Fpga-Based Architecture for Modular Nodes in Wireless Sensor Networks,” in *Southern Conference on Programmable Logic, 2007. SPL*, Mar del Plata, 2007.

[19] P. Latha and M. A. Bhagyaveni, “Reconfigurable FPGA Based Architecture For Surveillance Systems In WSN,” in *International Conference on Wireless Communication and Sensor Computing (ICWCSC)*, Chennai, 2010.

[20] J. Camarero, Design and implementation of a reconfigurable hardware system for 3D vision applications, Madrid: Final Degree Project UPM ETSII, 2012.

[21] SMEs project co-funded under EU's seventh framework (FP7), “FastCUDA,” January 2014. [Online]. Available: http://fastcuda.eu/.

[22] Inetsis, “http://gforgegroup.com,” 2014. [Online]. Available: http://gforge.inetsis.es/.

[23] OpenCV.jp, “OpenCV.jp,” 2013. [Online]. Available: http://opencv.jp/opencv-1.0.0_org/ChangeLog.

[24] libface Face Recognition Library, “http://sourceforge.net/projects/libface/,” February 2011. [Online]. Available: http://libface.sourceforge.net/file/Home.html.

[25] P. Kruizinga, “The Face Recognition Home Page,” 2013. [Online]. Available: http://web.archive.org/web/20030602192115/http:/www.cs.rug.nl/~peterkr/FACE/face.html.

[26] P. Belhumeur, J. Hespanha and D. Kriegman, “Eigenfaces vs. Fisherfaces: recognition using class specific linear projection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 19, no. 7, pp. 711-720, 1997.

[27] A. Gupta, K. Ravi, M. Gupta and K. Gupta, “ADT: Age determination technique,” in *Proceeding of International Conference on Methods and Models in Computer Science*, Delhi, 2009.

[28] N. Kumar, A. Berg, P. Belhumeur and S. Nayar, “"Attribute and simile classifiers for face verification,” in *International Conference on Computer Vision*, Kyoto, 2009.

[29] S. Z. Li and A. K. Jain, Handbook of Face Recognition, Second Edition ed., London: Springer, 2011.

[30] W. Zhao, R. Chellappa, P. J. Phillips and A. Rosenfeld, “Face recognition: A literature survey,” *ACM Computing Surveys (CSUR),* vol. 35, no. 4, pp. 399-458, 2003.

[31] R. Grimsdale, F. H. Sumner, C. Tunis and T. Kilburn, “A system for the automatic recognition of patterns,” *Proceedings of the IEEE on Radio and Electronic Engineering,* vol. 106, no. 26, pp. 210-221, 1959.

[32] F. Jin, J. Liu and W. Hou, “The application of pattern recognition technology in the diagnosis and analysis on the heart disease: Current status and future,” in *Control and Decision Conference (CCDC)*, Taiyuan, 2012.

[33] V. Blanz and T. Vetter, “Face recognition based on fitting a 3D morphable model,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 25, no. 9, pp. 1063-1074, 2003.

[34] A. M. Bronstein, M. M. Bronstein and R. Kimmel, “Three-Dimensional Face Recognition,” *International Journal of Computer Vision,* vol. 1, no. 64, pp. 1, 5-30, 2005.

[35] M. Turk and A. Pentland, “www.vision.jhu.edu,” 2008. [Online]. Available: http://www.vision.jhu.edu/teaching/vision08/Handouts/case_study_pca1.pdf.

[36] Yale University, “www.vision.ucsd.edu,” 2001. [Online]. Available: http://vision.ucsd.edu/~iskwak/ExtYaleDatabase/Yale%20Face%20Database.htm.

[37] Massachusetts Institute of Technology, Center for biological and computational learning, “Face Recognition Database,” MIT, 2003. [Online]. Available: http://cbcl.mit.edu/software-datasets/heisele/facerecognition-database.html.

[38] AT&T Laboratories Cambridge and Cambridge University Computer Laboratory, “The Database of Faces (The ORL Database of Faces),” The Digital Technology Group, 2002. [Online]. Available: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

[39] National Institute of Standards and Technology, “www.nist.gov,” 2011. [Online]. Available: http://www.nist.gov/itl/iad/ig/colorferet.cfm.

[40] M. Grgic and K. Delac, “Face Recognition Homepage,” VCL, 2013. [Online]. Available:

http://www.face-rec.org/databases/.

[41] M. Grgic and K. Delac, “Face Recognition Homepage,” VCL, 2013. [Online]. Available: http://www.face-rec.org/algorithms/.

[42] Y. Jiang and P. Guo, “Comparative studies of Feature Extraction methods with application to face recognition,” in *International Conference on Systems, Man and Cybernetics, . ISIC*, Montreal, Quebec, 2007.

[43] OpenCV Dev Team, “FaceRecognizer - Face Recognition with OpenCV,” 2013. [Online]. Available: http://docs.opencv.org/trunk/modules/contrib/doc/facerec/.

[44] Mashape, “list of 50+ Face Detection / Recognition APIs, libraries and software,” 2013. [Online]. Available: http://blog.mashape.com/post/53379410412/list-of-50-face-detection-recognition-apis.

[45] The MITRE Corporation, “Open Source Biometric Recognition,” OpenBR, 2013. [Online]. Available: http://openbiometrics.org/.

[46] C. S. S. Prasanna, N. Sudha and V. Kamakoti, “A principal component neural network-based face recognition system and ASIC implementation,” in *Proceedings on International Conference on VLSI Design*, 2005.

[47] Xilinx Inc, “LogiCORE IP Image Characterization v2.0,” 2011. [Online]. Available: http://www.xilinx.com/support/documentation/ip_documentation/v_ic/v2_0/pg015_v_ic.pdf.

[48] M. Kirby and L. Sirovich, “Application of the Karhunen-Loeve procedure for the characterization of human faces,” *Transactions on Pattern Analysis and Machine Intelligence,* vol. 12, no. 1, pp. 103-108, 2000.

[49] M. Turk and A. Pentland, “Face recognition using eigenfaces,” *Proceedings on Computer Vision and Pattern Recognition,* pp. 586,591, 1991.

[50] S. Zhang and M. Turk, “www.scholarpedia.org,” 2012. [Online]. Available: http://www.scholarpedia.org/article/Eigenfaces.

[51] P. Kulkarni, D. Ganesan, P. Shenoy and Q. Lu, “SensEye: a multi-tier camera sensor network,” in *Proceedings of the 13th annual ACM international conference on Multimedia*, New York, 2005.

[52] Y.-C. Tseng, Y.-C. Wang, K.-Y. Cheng and Y.-Y. Hsieh, “iMouse: An Integrated Mobile Surveillance and Wireless Sensor System,” *Computer,* vol. 40, no. 6, pp. 60-66, 2007.

[53] V. Saishanmuga and S. Rajagopalan, “A Neuro-Genetic System for Face Recognition,” *International Journal of Computer Science Issues (IJCSI),* vol. 9, no. 3, pp. 263-267, 2012.

[54] J. Matai, A. Irturk and R. Kastner, “Design and Implementation of an FPGA-Based Real-Time Face Recognition System,” in *Field-Programmable Custom Computing Machines (FCCM),* Salt Lake City, UT, 2011.

[55] M. Avvenuti, C. Baker, J. Light, D. Tulpan and A. Vecchio, “Non-intrusive Patient Monitoring of Alzheimer's Disease Subjects Using Wireless Sensor Networks,” in *World Congress on Privacy, Security, Trust and the Management of e-Business*, 2009.

[56] Aldebaran-Robotics, “www.aldebaran-robotics.com,” 2014. [Online]. Available: http://www.aldebaran-robotics.com/en/Solutions/For-Autism/The-Ask-NAO-initiative.html.

[57] M. López, J. Camarero, B. López and J. Valverde, HiReCookie Reconfiguration Guide, Madrid: CEI ETSII UPM , 2013.

Annex 1. Table of Identities

N	Name	N	Name	N	Name
1	David Aledo	44	Fátima Hernández	87	Alejandro Pozo
2	Pedro Alou	45	Elena Hernanz	88	Roberto Prieto
3	JL Aparicio	46	Pedro Herranz	89	Remi Pujo
4	Rafael Asensi	47	Juan Herrero	90	Regina
5	Jorge Arraez	48	Fermin Holguin	91	Alberto Rodríguez
6	María Arias	49	Cristina iglesias	92	Antonio Rodríguez
7	Laura Bonacasa	50	Diego Isla	93	Alfonso Rodríguez
8	Yann Bouvier	51	Sisi Zhao	94	Ángela Rojas
9	Julio Camarero	52	Leo Laguna	95	Víctor Rosello
10	Rubén Carnero	53	Eduardo Lezcano	96	Nieves Rubio
11	Giuse Cattalanoto	54	Manuel Llinas	97	Rubén Salvador
12	JA Cobos 1	55	Miguel Lombardo	98	Ignacio Sánchez
13	JA Cobos 2	56	Blanca López	99	Miguel Sánchez
14	Jorge Cortes	57	Carlos López	100	Pablo San Román
15	Justo Cubero	58	Fernando López	101	Mayid Shawi
16	Dejana Cucak	59	Victoria Maigler	102	Marcelo Silva
17	Javier Cuellar	60	Manu	103	Ismael Simon
18	Javier de Frutos	61	Daniel Martel	104	Tamara Sotorrio
19	E de la Torre	62	Alfonso Martín	105	Giuliano Sperandio
20	Miriam del Viejo	63	Marta Martín	106	Vladimir Svikovic 1
21	Daniel Díaz	64	Alfonso Martínez	107	Teresa Riesgo 1
22	Verónica Díaz	65	Sergio Mate	108	Teresa Riesgo 2
23	Borja Díez	66	David Meneses	109	Javier Torre
24	Benoit Duret	67	Sergio Merino	110	Yago Torroja
25	Yana Esteves	68	Teresa Merino	111	Javier Uceda

26	Luis Fernández	69	JM Molina	112	Alberto Valdés
27	Alej Fernández	70	Mariana Molina	113	Juan Valverde
28	JM Fernández	71	Javier Mora	114	Pablo Varela
29	Narciso Ferrero	72	Ángel Morales	115	Miroslav Vasic
30	Iván Flores	73	Félix Moreno	116	Filip Veljkovic
31	Airan Frances	74	JA Moreno	117	Sanna Vesti
32	Ángel Gallego	75	Gabriel Mujica	118	Mónica Villaverde
33	Óscar García	76	Kathleen Muller	119	Vladimir Svikovic 2
34	Alejandro García	77	Ana Neira	120	Wei Li
35	Julio García	78	Noemí Nogar	121	Fernando Pascual
36	Carmen González	79	Jesús Oliver	122	Elena Quesada
37	Miguel González	80	Andrés Otero	123	Natalia Pozhilova
38	Luis Guijarro	81	Zoran Pavlovic	124	Carlos Tejedor
39	Guixuan	82	Pengming Cheing	125	Rafael Zamacola
40	Danping He	83	David Pérez	126	Enrique Sánchez
41	Wei He	84	Federico Pérez	127	Adrián Peña
42	Nico Hensgens	85	Carlos Pizarro	128	Brad Pitt
43	Álvaro Hernández	86	Jorge Portilla		-