An FPGA-based accelerator for Fourier Descriptors computing for color object recognition using SVM

Fethi SMACH[1,2], Johel MITERAN[1], Mohamed ATRI[3], Julien DUBOIS[1], Mohamed ABID[2] and Jean-Paul GAUTHIER[1]

[1] Le2i Faculté Mirande Aile H. Université de Bourgogne BP 47870 21078 Dijon
[2] Laboratoire CES, ENIS, Sfax, Tunisie.
[3]Laboratoire EμE de Monastir, Tunisie

[1] {fethi.smach}@u-bourgogne.fr

Abstract. Fourier Descriptors can be used as feature vector components in various applications, such as real-time color object recognition or image retrieval. The full process is composed of the feature extraction followed by a classification step performed using Support Vector Machine (SVM). In order to accelerate the computation of Fourier Descriptors, a hardware implementation using FPGA technology is presented in this paper. We evaluated classification performance with respect to lighting variations and noise sensibility. Several experiments were carried out on three databases. Then an efficient architecture for FD computation on FPGAs is proposed and designed as accelerator. The WildCard is used to prototype this implementation. This design can have an operation speed up of approximately 10 compared to the standard software PC implementation.

Keywords: Fourier Descriptors, color object recognition, Field Programmable Gate Array (FPGA), SVM.

1. Introduction

Feature extraction and object recognition are subjects of extensive research in the field of image processing. Color object recognition is widely used in the machine vision industry in real time applications. A central issue is the recognition of objects independently of their position. To do this, the real-time extraction of invariant descriptors with respect to similarity transformations, while taking the local texture into account, remains a crucial challenge: it often consumes most important of the computation time of the recognition process. We therefore focused on the acceleration of feature computation in this paper. In other works, authors have dealt with the classification implementation issue [1] [2] [3].
The recognition process is divided into two parts: the training (the off-line phase) and decision steps (the on-line phase) (Fig 1). The result of the training step is the model determined by the SVM based method [4]. During the decision step, the object is

classified using a feature vector, the classifier and the model which was previously computed.

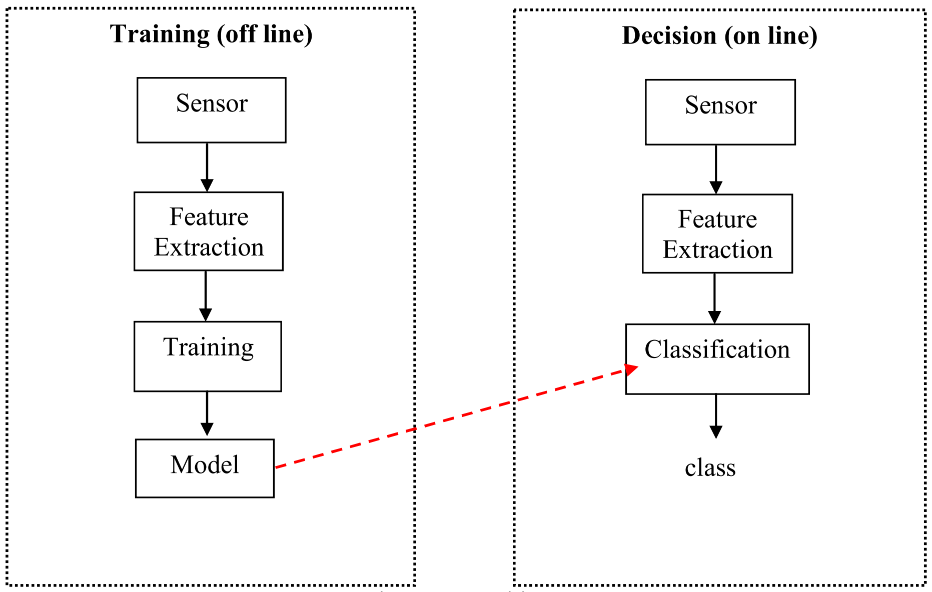

Fig. 1. Recognition steps

Fourier Descriptors are used as feature vector components in various applications, such as object classification, and image retrieval [5] [6]. Gauthier et al [7] proposed a family of invariants in translation, rotation, and scale. H. Fonga [8] extended the Fourier Descriptors, defining Similarity Descriptors and applying them to gray level images. We extended the notion of Fourier Descriptor invariants to color images classification in [9]. As mentioned above, our aim here is to accelerate the computation of Fourier Descriptors with hardware implementation. We propose in this paper efficient hardware architecture for FD implementation on Field Programmable Gate Arrays (FPGAs). FPGAs were originally developed for hardware circuit designs. They may be used as powerful computing systems for image processing algorithms [10] [11] [12] [6]. These computations can be performed much faster than on the host PC, mainly because of the high parallelism allowed by the internal structure of the component. Thus, the FPGA devices on which such applications are built offer a good medium for implementing complex computational tasks characterized by high throughput and low latency requirements, providing orders of magnitude speedup in application processing at a fraction of the cost per processing operation. On the other hand, pre-designed Intellectual Property (IP) cores for FPGA represent a huge intellectual and financial wealth that must be leveraged by any high-level tool targeting reconfigurable platforms. These IP cores come in the form of synthesizable HDL code or even lower level descriptions. They vary drastically with respect to their control and timing protocol specifications, which are intended to be interfaced to HDL-based designs. Several projects have focused on bus wrapping that connects IP cores with microprocessors. In [13], Mukherjee describe a system level approach for interfacing IP blocks generated by the behavioral synthesis tool itself. In [14], Guo proposed an automation of IP core interface generation for reconfigurable computing. The main contributions of this paper are: the application of Fourier Descriptors to color object recognition and the development of a hardware accelerator for feature invariant computing. The work reported in this paper can be combined with the previous work by Miteran and al. who proposed in [15] a hardware implementation of an approximation of the SVM decision function.

This paper is organized as follows: Section 2 is a review of Fourier Descriptors and SVM based classifiers. Section 3 describes the evaluation of classification performances using software implementation. In section 4 we propose our hardware architecture. Section 5 concludes the paper.

2 Review of Fourier Descriptors and SVM classifier

There exists an extensive literature which addresses both the theoretical and applied aspects of invariant descriptors. It is important that such invariants fulfill certain criteria such as low computational complexity and completeness. A complete invariant implies that two objects have the same shape if and only if their invariant descriptors are the same. The invariant property is relative only to a certain transformation. A feature vector of a Fourier Descriptor invariant with respect to similarity transformations (rotation, translation and scale) is used as an input in a Support Vector Machine (SVM) based classifier. This section will first give a brief definition and outline the elementary properties of Fourier Descriptor invariants. This is then followed by a brief description of a SVM classifier.

2.1 Definition of Fourier Descriptors

Fourier Descriptors (FD) are defined as follows. Let f be a square summable function on the plane, and $\hat{f}$ its Fourier transform:

$$\hat{f}(\xi) = \int_{\mathbb{R}^2} f(x) \exp(-j\langle x \mid \xi \rangle)\, dx \qquad (1.1)$$

Where $\langle . \mid . \rangle$ is the scalar product in $\mathbb{R}^2$.

If (λ, θ) are polar coordinates of the point ξ, we shall again denote $\hat{f}(\lambda, \theta)$ the Fourier transform of f at the point (λ, θ). Gauthier defined the mapping D_f from $\mathbb{R}_+$ into $\mathbb{R}_+$ by

$$D_f(\lambda) = \int_0^{2\pi} \left| \hat{f}(\lambda, \theta) \right|^2 d\theta \qquad (1.2)$$

D_f is the Fourier Descriptor of the image f, i.e. the feature vector which describes each image and will be used as an input in the supervised classification method.

2.2 Properties of Fourier Descriptors

Fourier descriptors, calculated according to the equation (1.2), have several elementary properties which are crucial for invariant object recognition [7]:
Fourier descriptors are motion and reflexion-invariant:

- If M is a "Motion" and f and g are images such as $g(x) = (f \circ M)(x)$, where $(f \circ M)(x)$ is a composed function: f applied to $M(x)$. Thus, images g and f have the same descriptors D:
  $$D_g(\lambda) = D_f(\lambda), \forall \lambda \in \mathbb{R}^2 \qquad (1.3)$$
- If there exists a reflexion $\Re$ such that $g(x) = (f \circ \Re)(x)$,

$$D_g(\lambda) = D_f(\lambda), \forall \lambda \in \mathbb{R}^2 \qquad (1.4)$$

Motion descriptors are scaling-invariant:

- if k is a real constant such as $g(x) = f(kx)$,

$$D_g(\lambda) = \frac{1}{k^4} D_f(\frac{\lambda}{k}), \forall \lambda \in \mathbb{R}^2 \qquad (1.5)$$

The Fourier transform $\hat{f}$ will be computed from the FFT estimation.

2.3 Review SVM-based classification

It has been shown that the SVM method provides very good results in many practical cases [16], [17]. SVM is an universal learning machine developed by Vladimir Vapnik [4] in 1979. A review of the basic principles follows, using the example of a 2-class problem (whatever the number of classes, the problem can be reduced, by a "one-against-others" method, to a 2-class problem). The SVM performs a mapping of the input vectors (objects) from the input space (initial feature space) R^d into a high dimensional feature space Q; the mapping is determined by a kernel function K. It finds a linear decision rule in the feature space Q in the form of an optimal separating boundary, which leaves the widest margin between the decision boundary and the input vector mapped into Q. This boundary is determined by solving the following constrained quadratic programming problem:

Maximize:

$$W(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathrm{x_i}, \mathrm{x_j}) \tag{1.6}$$

Under the constraints

$$\sum_{i=1}^{n}\alpha_i y_i = 0 \tag{1.7}$$

and $0 \le \alpha_i \le T$ for i=1, 2, …, n where $\mathrm{x_i} \in R_d$ are the training sample set vectors, and $y_i \in \{-1,+1\}$ the corresponding class label. *T* is a constant needed for non separable classes. *K(u,v)* is an inner product in the feature space *Q* which may be defined as a kernel function in the input space. The condition required is that the kernel *K(u,v)* be a symmetric function which satisfies the following general positive constraint:

$$\iint_{R_d} K(\mathrm{u},\mathrm{v})\,\mathrm{g}(\mathrm{u})\,\mathrm{g}(\mathrm{v})\,\mathrm{d}\mathrm{u}\,\mathrm{d}\mathrm{v} > 0 \tag{1.8}$$

Which is valid for all *g≠0* for which $\int g^2(u)\,du < \infty$ (Mercer's theorem).

The choice of the kernel K(u, v) determines the structure of the feature space Q. A kernel that satisfies the equation (1.8) may be presented in the form:

$$K(\mathrm{u},\mathrm{v}) = \sum_k a_k \Phi_k(\mathrm{u})\Phi_k(\mathrm{v}) \tag{1.9}$$

Where a_k are positive scalars and the functions Φ_k represent a basis in the space *Q*. We use a Radial Basis Function SVM (RBF):

$$K(\mathrm{x},\mathrm{y}) = e^{\left(\frac{-\|\mathrm{x}-\mathrm{y}\|^2}{2\sigma^2}\right)} \tag{1.10}$$

The separating plane is constructed from those input vectors, for which $\alpha_i \neq 0$. These vectors are called *support vectors* and reside on the boundary margin. Mapping the separating plane back into the input space R_d, gives a separating surface which forms the following nonlinear decision rules:

$$C(\mathrm{x}) = \mathrm{Sgn}\left(\sum_{i=1}^{Ns} y_i\alpha_i \cdot K(\mathrm{s}_i,\mathrm{x}) + b\right) \tag{1.11}$$

This robust method is not often used for high speed decision problems such as fast video, because of the complexity of the decision rule. Nevertheless, we have shown that real-time performance can be obtained. Indeed, we proposed in previous studies [15] [5] a FPGA based implementation of an approximation of the support vector machine decision rule. If we combine this implementation of the decision function with the implementation of the Fourier Descriptors computation described below, it will be possible to implement the full real time recognition process using a single FPGA component.

3. Performance evaluation

Performance evaluation is a critical step which has to be performed in order to validate an object recognition algorithm. The test protocol used for performance evaluations is a standard cross-validation method (SVM classification error measurements based on multiple tests using separated training and decision sample sets). We tested our approach using several standard databases, and we evaluated the robustness against noise addition and light variation.

3.1 General evaluation

The first database is the COIL-100 [18] which is composed of color images of 100 different objects, where 72 images of each object were taken at pose intervals of 5°. The images were pre-processed in such a way that each of them fits the size of 128x128 pixels. The second and third databases are composed of images of human faces. Indeed, face recognition is a difficult problem for which many methods have been examined [19] [20].
The ORL database [21] used in this paper is composed of 400 gray level images of size 112x92; there are 40 faces with ten images per face. The images are taken at different moments in time, with varying lighting conditions, facial expressions (open/closed eyes, smiling/not-smiling), and facial details (glasses/no glasses). All the subjects are an up-right, frontal position (with tolerance for some pose variation).
The AR-faces database was created by Martinez in the computer vision center [22]. It contains over 4.000 color images corresponding to 126 people’s faces (70 men and 56 women). Images feature frontal view faces with different facial expressions, illumination conditions, and occlusions (sunglasses and scarf). Each image in the database consists of a 786x576 array of color pixels (RGB).
The error rate is shown in table 1; we have compared our descriptors to other classification families of invariants, such as Zernike moments [23].
Other methods in the literature testing the COIL-100 database provide error rates ranging from 12.5% to 0.1%. See for instance [24].

SVM, RBF Kernel $\sigma_{opt} = 0.1$	COIL	ORL	AR-faces
LAFs [24]	0.1%	NA	NA
Nearest-Neigbor [20]	NA	2.1%	NA
Gabor wavelet [19]	NA	15%	NA
Eigenface – SVM [22]	NA	NA	5%
Fourier Descriptors	0.09%	9.5%	2.31%
Zernike Moments	0.22%	25%	10.61%

Table1: Performance evaluation (error rate using cross-validation)

It is clear that the Fourier Descriptors outperform the Zernike Moments in all cases, and our results are similar to or better than (for COIL and AR-faces databases) performances obtained by other authors using the same databases [24] [22].

3.2 Robustness against noise

In order to study the robustness of Fourier Descriptors against noise addition, we evaluated the classification error obtained using a noisy database. This database was created by adding some Gaussian noises to the COIL images. In order to test several noise levels, we created databases with different standard deviations S_d ($0.08 < S_d < 0.23$). Some examples of noisy images are depicted in Fig 2.

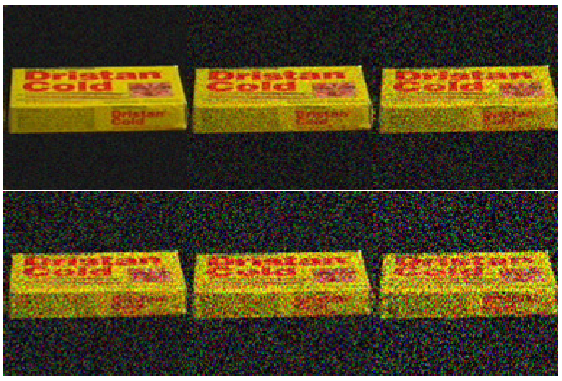

Fig. 2. Sample of COIL noisy object

Table 2 presents our results with noisy databases. Results show noise has little influence on classification performance.

St. Dev. of Gaussian noise	Zernike Moments	Fourier Descriptors
0.08	0.29%	0.36%
0.16	0.34%	0.40%
0.23	0.43 %	0.38 %

Table 2: Robustness against noise (error rate using cross-validation)

3.3 Robustness with respect to lighting variation

We performed several robustness tests with lighting variations using a self made database of 15 objects. We provided images corresponding to two lighting conditions (Fig 3). We trained the system with images taken in the first lighting conditions and we tested the data set obtained with the second lighting conditions.

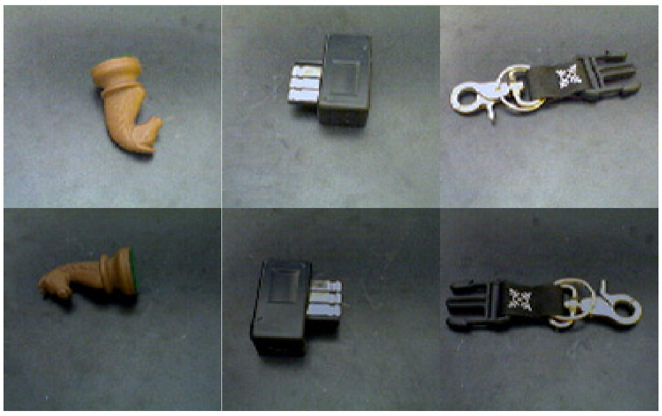

Fig 3: Different Lighting conditions

In this experiment we introduced a pre-processing step which consisting of contour extraction, based on a simple Sobel filter. The results are depicted in Fig 4. The horizontal axis represents the learning sample percentage and the vertical axis represents the error rate. We observe that, as expected, contour extraction improves the results, since the error $e< 5\%$ is obtained when only 4% of samples are used during the training step, while without contour extraction the error is $e\approx 10\%$. A lower image number is therefore required using contour extraction as a preprocessing step.

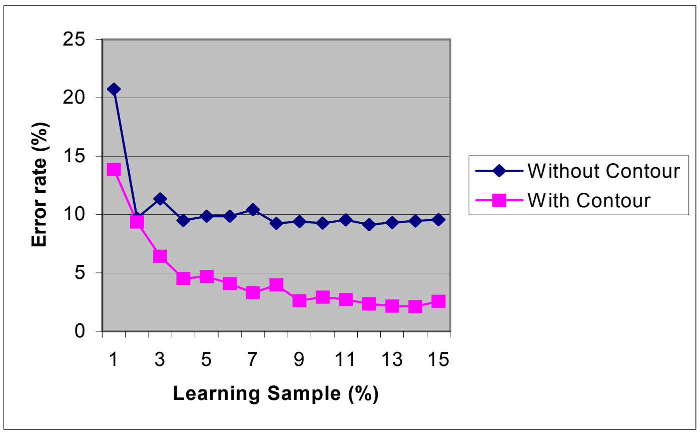

Fig 4: Influence of contour-extraction on classification error

These experiments show that Fourier Descriptors can be used for real time applications such as face recognition [20] and quality control by artificial vision. For all these applications, high speed custom hardware is often useful and sometimes necessary.

4. Hardware Implementation

4.1 General Specification

Recently, some research has been launched on the subject of hardware and software (HW/SW) Co-design. The Co-design approach consists of several steps such as, high level HW/SW Co-simulation partitioning and system prototyping. The Co-design approach analyzes the timing of the different portions of the algorithm. The time extensive parts are implemented in hardware if resources thus permit. This is known as “Hardware Acceleration”. Using the Co-design methodology in embedded systems development provides the capability of meeting strict design constraints in terms of power, size and timing. We analyzed the full recognition process in order to determine which part of the algorithm needed to be accelerated.
As mentioned in the introduction, the object recognition process is divided into two steps: training and decision. During both steps the input image is resampled to 128x128 pixels, and a standard FFT is computed for each color channel (Red, Green, and Blue) (Fig 5). The three corresponding Fourier Descriptors are computed from the FFT values. The final size of the vector used for classifier training is d=63x3=189 (the first component value for each channel is used for normalization). The result of the training step is the model (set of support vectors) determined by the SVM based method.

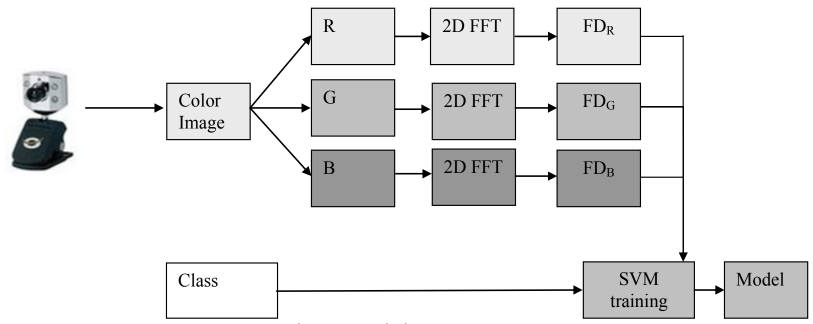

Fig. 5. Training process

During the decision step, which is the only one computed in real-time, the Fourier Descriptors are computed in the same way, and the model determined during the training step is used to perform the SVM prediction. The output is the image class (Fig 6).

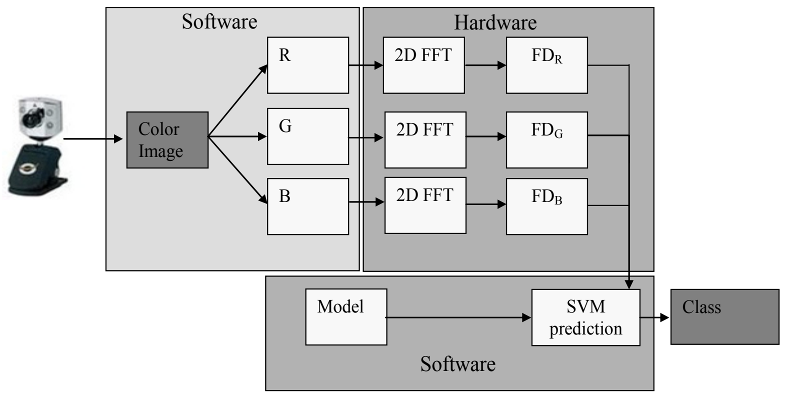

Fig. 6. Decision process

In our applications (face or object recognition for quality control by artificial vision), the complexity of the decision function is lower than that of the feature computation. Typically, we obtain around 100 support vectors, so the SVM prediction can be performed in 5 ms on a standard PC, whereas the FD extraction is completed in 30 ms. we therefore decided to implement FD computation on hardware to accelerate the process.

4.2 Prototyping Platform and design methodology

In order to implement the full HW/SW co-design, we used a prototyping platform developed by Calgary University and the MPEG/ISG group [26] [27]. The platform is based on a standard PC and a single PCMCIA FPGA-based board. This board is the WILDCARD PCMCIA (Xilinx Virtex II XCV3000) card from Annapolis Micro

Systems [28] and is plug-compatible into a laptop Cardbus slot. The WILDCARD has a very compact architecture with a Virtex II 3000K FPGA from Xilinx, and two SRAM banks, each 256 KB in size. The WILDCARD uses the 32-bit CardBus interface, using a dedicated chip to provide this bus support. The architecture block diagram is presented in Fig 7.

Each of the two memory blocks, referred to as the right and left memory banks, is 64Kx32-bit RAM module, with a 16-bit address bus and 32-bit data word. The FPGA can write and read from the right and left memories independently. The host interface is through a 32-bit CardBus (PCMCIA) controller that operates at a 33 MHz clock frequency. Data transfers to and from the PC host are performed via the control of a set of C program driver calls that interface with the CardBus controller which, in turn, interfaces with the LAD bus to send data to, and retrieve data from, the FPGA [28].

A standard HDL–based design methodology was used. We model the algorithm using the VHDL hardware description language; we functionally verify the correctness of the algorithm in the custom architecture, and then we synthesize the architecture onto a set of resources to produce a circuit mapped to target the FPGA device. VHDL modules that come with the card are written to target a specific synthesis tool.

The VHDL modules are easily interconnected to a PCI interface using the Application Programming Interface (API) provided by the WILDCARD vendor. These modules are hardware accelerators that can be controlled and called from the host with a high abstraction level. Indeed, from the host's point of view, the accelerator's calls can be considered as usual C functions. The set of hardware accelerators is a co-processor which provides a high level of parallelism (intrinsic parallelism of the FPGA, and parallelism obtained by the association of the host and the FPGA). Most of the platform's IP have been developed in an image compression context to achieve real time performances [29] [30]. We propose to take advantage of the Wildcard platform capacities for our color object recognition application.

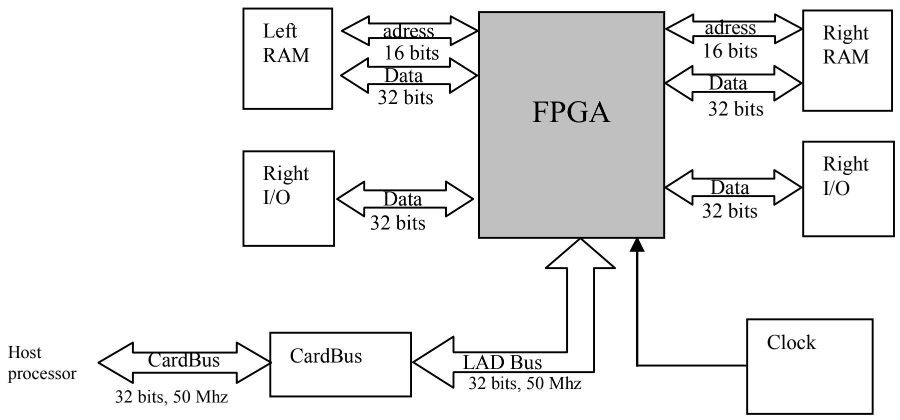

Fig. 7 WILDCARD Block Diagram [28].

4.3 Architecture for FD implementation

Global architecture

The aim of our work is to achieve the hardware implementation of Fourier Descriptors computation according to equation (2). The processing can be split in two computational steps:

- Computation of the Fourier transforms $\hat{f}$ of the image f, using 2D FFT.

- Computation of some integral expression of $\hat{f}$ over circles in the frequency plane.

The proposed architecture is shown in detail in Fig 8. It is constituted of five main units: Input/Output FIFO memories, 2D-FFT accelerator, Fourier Descriptors accelerator, a set of two temporary memories, and a controller based on a Finite State Machine (FSM) which controls the process. The proposed architecture has two basic FIFO (RAM) blocks of 4K. The FIFO_in and FIFO_out are in charge of data exchange between the host and the 2D FFT accelerator. These units simplify task scheduling and prevent asynchronism.

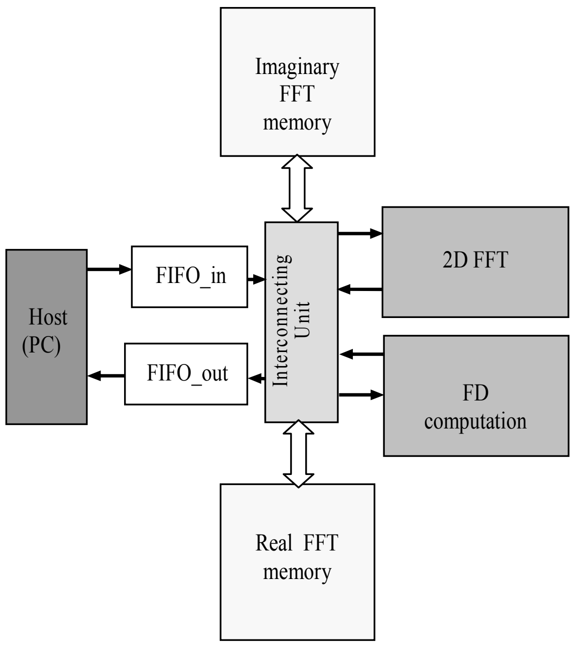

Fig 8: Data-flow architecture of the hardware accelerator

2D FFT computation

The 2D FFT was designed to support large size images. For instance, we propose in this paper a core implementation for 128x128 color images. The 2D FFT is processed for each color channel (red, green and blue). The accelerator is based on the Xilinx standard Logic Core 1D FFT IP Core configured for 128 points, operating using 16-bit data. This FFT core implements the Cooley-Tukey algorithm, using pipeline and streaming I/O. This solution offers continuous data processing. The core is able to perform transform calculations on the current data frame, while simultaneously loading for the next data frame, and unloading the results of the previous data frame. Depending on the performance required, the architecture can easily be adapted. In the 1 D configuration, the design is a low cost solution in term of HW resources. The performance can be increased easily by adapting the IP to process several rows or columns in parallel or by multiplying the number of IP to process the three color channel simultaneously.
The 2D FFT is separated into rows and columns, so the process is split into two steps:

- In the first step the FFT-IP is used to compute the FFT 1D of each row of the input image. The results are stored in two memories for real and imaginary parts of the FFT (steps labeled 1 and 2 in Fig 9).
- In the second the FFT-IP is reactivated to compute the FFT 1D of the column of the FFT image formed by the previously stored results (steps labeled 3 and 4 in Fig.9).

The final results are the real and imaginary data parts of the FFT stored in two separate memories (Fig 9).

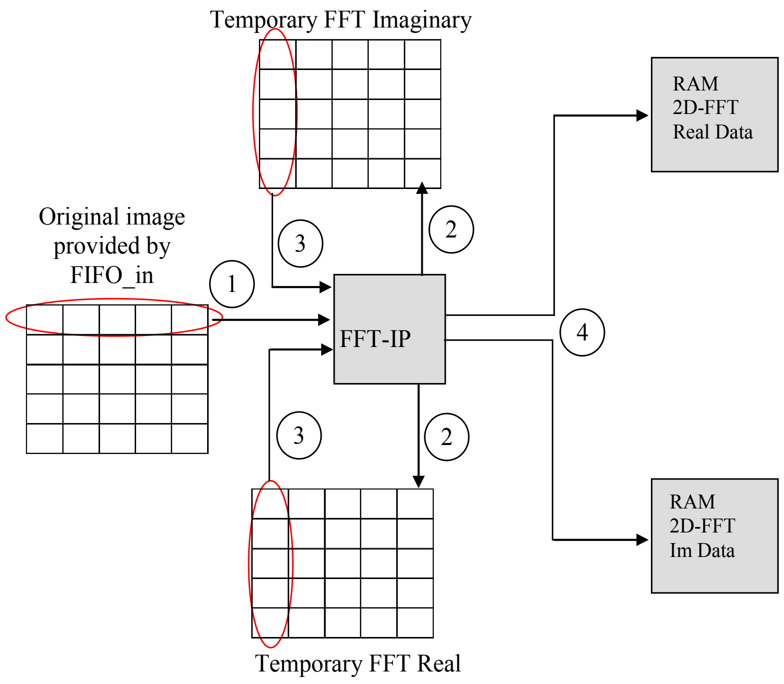

Fig 9: Diagram block of 2D-FFT computation

Circular integration for FD computation

The integral values to be computed according to eq. (1.2) are approximated by accumulating the FFT values which belong to a centered crown. Each crown is one pixel wide in the Fourier space, and provides one Fourier Descriptor (see Fig 10). Therefore, 64 FD values are obtained for a 128x128 image. The central value is used for normalization (performed by the host) and the other 63 values are the components of the feature vector used for classification.

For each pixel of the Fourier space, the radius of the crown which it belongs to must be determined. In order to avoid this high-cost computation in terms of time processing and hardware resources, we propose that these fixed values be computed off-line and stored in a 128x128 ROM.

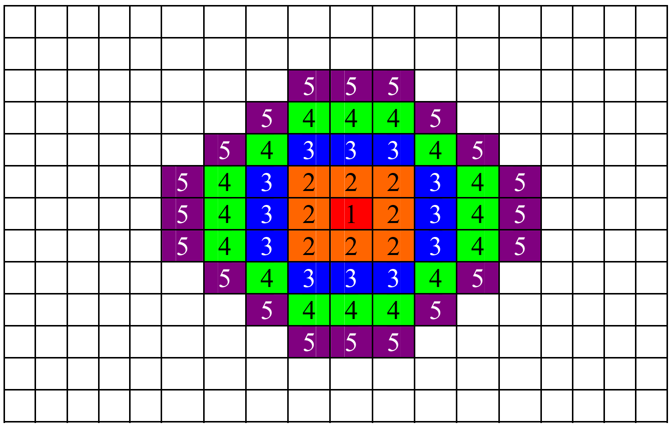

Fig 10: Example of considered circles in the Fourier space

The architecture of the circular integration is depicted in Fig 11. The radius-ROM address is determined by the pixel position (row and column values). The radius value read in this radius-ROM is used as an address of the FD-RAM (64 words of 16 bit width), where FD final values are stored. The FD-RAM data output is connected to an adder in order to form an accumulator. The other input of the adder is the squared value of the FFT module. The FD-RAM content is initialized to 0 for each new image. The final FD values are read after the last pixel processing, and transmitted to the host using the FIFO_out. One accumulation is composed of four steps:

1- Read the radius in Radius-ROM, and the FFT values in the 2D FFT RAM,
2- Read the previous FD value in the FD-RAM and compute the squared FFT module,
3- Add the previous FD value and the squared FFT module,
4- Store the resulting value in the FD-RAM.

A control unit schedules the data flow.

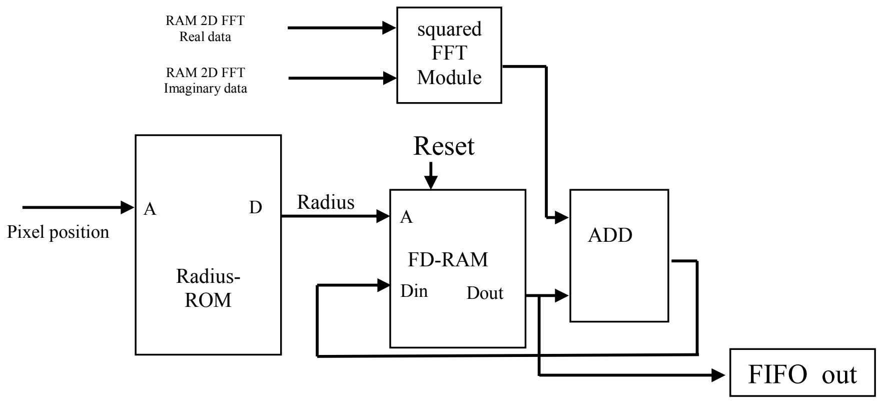

Fig 11: Block diagram architecture for computing Fourier Descriptors

4.4 Experimental Results

The architecture described above was simulated and implemented targeting Xilinx FPGA. The hardware implementation results are shown in table 3, for one color channel 128x128. The working frequency is about 33 Mhz.
The execution time of the same part of the algorithm, using the same optimizations (use of radius-ROM) on µP-based with Pentium 4 (2.81 GHz) is 10 ms. The execution time of the FPGA implementation part is approximately 0.5 ms. Taking into account the time required for data transfer between the host and the Wildcard, the whole execution time of the SW/HW process is around 1 ms. The acceleration factor is therefore about 10.

Logic Utilization	Used	Available	Utilization
Number of Slices	2900	14336	20%
Number of Slice Flip Flops	3495	28672	12%
Number of 4 input LUTs	4072	28672	14%
Number of bonded IOBs	68	484	14%
Number of BRAMs	41	96	43%
Number of MULT18X18s	14	96	15%

Table 3: Synthesis of results of Fourier Descriptor implementation

5. Conclusion

An efficient FPGA based architecture for hardware acceleration of Fourier Descriptors has been presented in this paper.
The classification performance of the proposed method was evaluated using several standard databases, and revealed that our approach can be useful in many applications of object recognition.

The prototyping platform used is based on a Wildcard, allowing an easy to use co-processing system to be designed. The image acquisition and the final classification using SVM are executed in software, while the feature vector computation is implemented in hardware. This has allowed us to accelerate global execution time in order to meet real time constraints.

The obtained acceleration is around 10 for one color channel, and can be increased using a pipelined model for each color channel.

It is important to note as well that Fourier Descriptors can also be used in multiple Regions of Interest of larger images, or as local feature vectors, as well as in many algorithms based on the windowed Fourier transform. In these cases, for which the window size is frequently smaller than the one presented in this paper (typically 32x32 of 16x16 windows) our architecture can easily be adapted to these smaller windows, and duplicated in the FPGA to improve parallelism.

The HW/SW solution presented here may be considered as a first implementation. We also worked in the past on the FPGA implementation of an approximation of the SVM decision function [15]. Our future work will therefore address the integration of the whole recognition process – feature extraction and classification - on the FPGA, freeing the host processor for other tasks.

References

1. R. Reyna Rojas, R. Houzet, M. F. Albenge, D. Esteve, “Implementation of the SVM neural network generalization function of for image processing”, International Workshop on Computer Architecture for Machine Perception (CAMP’2000), Padova (Italy), 11-13 Septembre 2000, pp. 147-151.
2. J. Zhu, P. Sutton, FPGA Implementation of neural networks: “a survey of a decade of progress”, in: Proceedings of the 13th International Conference on Field Programmable Logic and Applications (FPL 2003), Lisbon, Portugal, 2003, pp. 1062–1066.
3. F. Yang , M. Paindavoine, “Implementation of a RBF neural network on embedded systems: Real time face tracking and identity verification”, IEEE Transactions on Neural Networks, Vol.14 (N°5), (2003), pp. 1162-1175.
4. V. Vapnik, “The nature of statistical learning theory”, Springer-Verlag, (1995) New York.
5. Y. Raj Bahadur, K. Naveen Nishchal, K Arun Gupta, K. Vinod Rastogi, “Retrieval and classification of shape-based objects using Fourier, generic Fourier, and wavelet Fourier Descriptors technique: A comparative study”, Optics and Lasers in engineering 45, 2007, pp. 695-708.
6. F. Javier Diaz, A. M Buron, J. M. Solana, “Haar wavelet based processor scheme for image coding with low circuit complexity”. Computers and Electrical Engineering 33, 2007, pp. 109-126.
7. J. P.Gauthier, G. Bornard, M. Silbermann, “Harmonic analysis on motions groups and their homogenous spaces” , IEEE Trans. on Systems, Man and Cyb., vol 21, 1991, pp. 159-172.
8. H. Fonga, Analyse harmonique sur les groupes et reconnaissance de formes, PHD thesis, université de Grenoble, 1992.
9. F. Smach, C. Lemaitre, J. Mitéran, J.P. Gauthier , M. Abid, “Colour Object recognition combining Motion Descriptors, Zernike Moments and Support Vector”, Proceeding of IECON’06, IEEE, Paris-CNAM, France, 2006, pp. 3238-3242.
10. D. Crookes, K. Benkrid, A. Bouridane, K. Alotaibi and A. Benkrid, “Design and implementation of a high level programming environment for FPGA-based image

processing", IEE Proceedings on Vision, Image and Signal Processing, vol 147(4), 2000, pp 377-384.
11. M. A. Tahir , A. Bouridane, and F. Kurugoullu, "An FPGA based coprocessor for GLCM and Haralick texture features and their application in prostate cancer classification", Analog Integrated Circuit and Signal Processing, vol. 35, 2005, pp. 205-215.
12. D. Nguyen, D. Halupka, P. Aarabi, A. Sheikholeslami, "Real-time face detection and lip feature extraction using field programmable gate arrays", IEEE Transactions on Systems, Man, and Cybernetics-Parts B: Cybernetics, vol. 36, no 4, 2006, pp. 902-912.
13. R. Mukherjee, A Jones, P. Banerjee, "System Level Synthesis of Multiple IP Blocks in the Behavioral Synthesis Tool", Int. Conf. on Parallel and Distributed Computing and Systems (PDCS), 2003.
14. Z. Guo, A. Mitra, W. Najjar, "Automation of IP Core Interface General for Reconfigurable Computing". 16th International Conference on Field programmable Logic Applications (FPL 2006) Madrid, Spain, 2006.
15. J. Mitéran, S. Bouillant, E. Bourennane, "SVM Approximation for Real-time Image Segmentation by Using an Improved Hyperrectangles-based Method, Real Time imaging". Elsevier, vol. 9 (3), 2003, pp. 179-188.
16. P. Niyogi, C. Burges, P. Ramesh, "Distinctive Feature Detection Using Support Vector Machines", ICASSP 99, 1999, pp. 425-428.
17. B. Schölkopf, A Smola, K-R. Müller, C.J.C. Burges and V. Vapnik, "Support Vector methods in learning and feature extraction", Australian Journal of Intelligent Information Processing Systems, vol. 1, 1998, pp. 3-9.
18. htttp://www.columbia.edu/CAV.
19. E. Hjelmas, "Face Detection: A Survey", Computer Vision and Image Understanding, no. 83, 2001, pp. 236-274.
20. R. Huang, V. Pavlovic, D. N. Metxas, "A hybrid face Recognition Method using Markov random Field"s, in Proceeding of ICPR, 2004, pp. 157-160.
21. ORL face database, AT&T Laboratories, Cambridge, U.K. http://www.cam-orl.co.uk/facedatabase.html.
22. A M. Martinez, "Recognition of Partially Occluded and/or Imprecisely Localized Faces Using Probabilistic Approach", Proceeding of IEEE Computer Vision and Pattern Recognition, CVPR'2000, pp. 712-717.
23. A. Khotanzad, H.H. Yaw, "Invariant image recognition by Zernike moments", *IEEE Trans. PAMI*, vol. 12, no. 5, 1990, pp. 489-497.
24. S. Obrzalek and J. Matas, "Object recognition using local affine frames on distinguished regions", Electronic Proceeding of the 13th British Machine Vision Conference, University of Cardiff, 2002, pp. 113-122.
25. G. D. Guo, S. Li, and K. Chan, "Face recognition by support vector machines", Proceedings of International Conference on Automatic Face and Gesture Recognition, 2000, pp 196-201.
26. Q. Yifeng, W. Badawy, "Tutorial on an Integrated Virtual Socket Hardware-Accelerated Co-design Platform for MPEG4-Part9 ISO/IEC JTC1/SC29/WG11 M12789", Bangkok, Thailand – January 2006, pp. 03-28.
27. P. Schumacher, M. Mattavelli, A. Chirila-Rus, R. Turney, "A Virtual Socket Framework for Rapid Emulation of Video and Multimedia Designs", Multimedia and Expo, ICME 2005, 6-8 July 2005, pp. 872 – 875
28. Annapolis Microsystems Inc, Annapolis WILDCARD System Reference Manual, Revision 2.6, 2003, www.annapmicro.com
29. J. Dubois, M. Mattavelli, L. Pierrefeu, J. Mitéran, "Configurable Motion-Estimation Hardware Accelerator Module for the MPEG-4 Reference Hardware Description Platform", Proceeding of IEEE International Conference on Image processing (ICIP), Genova, 11-14th September, vol. 3, 2005, pp. 1040-1043.

30. A. Kinane, V. Muresan, N. E. O'Connor, N. Murphy, S. Marlow, "Energy-Efficient Hardware Architecture for variable N-point 1D DCT" Proceeding of PATMOS, 2004, pp. 780-788.