VLIW Processor Architecture Exploration for Facial-Feature and component extraction

Nadia Nacer[1], Bouraoui Mahmoud[2], Mohamed Hedi Bedoui[3]
[1,2,3] Laboratory of Imaging and Medical Technology, Faculty of Medicine at Monastir, University of Monastir – Tunisia
[2] National Engineering School at Sousse, University of Sousse-Tunisia
nd.nacer@gmail.com, Medhedi.bedoui@fmm.rnu.t, bouraoui.mahmoud@eniso.rnu.tn

Abstract— ***The methods for face-detection and facial-feature and component extraction exist in the literature are different in their complexity, performance, type, and nature of the images and the targeted application. The facial features and components are used in medical diagnoses, security applications, robotics, and assistance for the disabled. We use these components and feature to determine the state of alertness and fatigue for medical diagnoses. In this work we use plain color background images whose color is different from the skin. The image contains a single face. We are interested in the VLIW processor architecture dedicated for this application. This dedicated architecture must meet two constraints, which are the execution time and the VLIW processor functional unit resources. We have selected and associated a face detection algorithm based on the skin detection (using the RGB space) with a facial-feature extraction algorithm based on tracking the gradient and applying the geometric model.***

Keywords— **Face detection, Face components, Face features, Skin detection, Architecture exploration, VLIW processor, Functional Unit**.

I. Introduction

The facial feature and component extraction is an important step in several applications of assistance for the disabled (face communication), robotics, security (biometrics, surveillance, and driving safety, decreased alertness detection, fatigue detection). The extraction of these components and features allows the facial expressions analysis. The facial expressions consist in a temporary distortion of a facial structure. There are two types: permanent structure (eyes, mouth, deep wrinkles, hair, and brow) and transient structure (swelling, wrinkles). We use these components and features to determine the alertness and fatigue state for medical diagnoses.

The implementation of digital image processing algorithms can be made in hardware, software, or a hardware/software way. The design type depends on the intended application and the selected constraints (execution time, performance, memory space, power consumption,...). If we are interested in real-time applications, the hardware/software solution is ranked as the best solution which satisfies the time constraint and the complexity algorithm. The application implementation on the VLIW (Very Long Instruction Word) processor architectures achieves a compromise between the performance, hardware complexity and algorithm data dependence degree. These architectures have the ability to run more than one operation (instruction) at a time with a reduced hardware complexity.

The embedded systems for acquiring, processing and analyzing facial expressions require increasingly complex algorithms. These algorithms require a computational power and a large memory space offered by the VLIW processor architecture. The implementation of the application of acquiring, processing and analyzing the facial expressions must meet a real-time execution (the camera frame rate), while minimizing the resource consumption.

Several studies in the literature have proposed a face detection implementation on the processor and on the FPGA. The authors in [i] proposed a face detection implementation (Haar-classifier) on a computer (Intel Core 2 CPU (2.80 GHz), 2.98 GB DDR2 SDRAM (800 MHz), Windows XP Professional, and Visual Studio) allowing the treatment of 0.71 frames/s [i]. The same application implemented on the FPGA (Viertx5-Xilinx) of images (320*240) with an execution time of 34.7ms (28.8 frames/s). In [iii] the authors implemented a face detection algorithm (color model) for an image (800*600) on the FPGA (Virtex II) which was used to treat up to 90frames/s. The work done by [ii] was a face detection parallel implementation (Convolutional Face Finder) on the FPGAs (Virtex 4) designing 32 Elementary Processors that operate at 350MHz. In [iv] the authors made a face detection implementation (color model) of an image (176*144) on the Altera APEX FPGA that could handle 434 frames/s. Several other studies have conducted implementations on the FPGA, such as [v] (processing time is 200 ms for an image (180*180) on FPGA Stratix II), [vi] (processing speed is 50 frames/s), [vii] (face detection using neural network for images (256*256) on the Virtex FPGA with an 99.67% occupancy rate). The autors in [xix] propose an FPGA implementation of face detection and facial components extraction for images (192*256) (490 frames/s) on the Virtex 5.

The face-detection and facial-expression extraction with various methods is not an easy task. It requires hardware resources with a varied execution time depending on the method used, the image size, and the frame rate. Our objective is to design a dedicated VLIW processor architecture for a localizing and extracting facial-feature and component application. We aim to exploit the potential-parallelism algorithm, the data-dependence-degree algorithm and the memory-management-by-exploring-the-adequacy-architecture algorithmic. We will study and choose the optimization technique to be applied to the algorithm, to determine the number of Functional Units (FU) and the memory size of the dedicated VLIW processor architecture. In this work we are interested in two parts: face detection and interest-region localization (components) (eyes, forehead and mouth). We have chosen and combined several methods and algorithms that have a reduced complexity and a maximum performance. We have validated these algorithms using C language.

II. Material and Methodology

II.1. Face-detection and facial-component extraction algorithm:

There are various methods for the face detection and facial component extraction described in the literature. These methods present a difference in their complexity, performance, type, and nature of the images. In this work we use plain color background images whose color is different from the skin and which contain a single face. The image size (n,m) is (192*256) (figure1.a).We have chosen a face detection algorithm based on the skin detection (using RGB space) developed in [viii,ix,x,xi] and an algorithm of extracting the facial components based on tracking the gradient developed in [xii] and the geometric model developed in [xiii].The method consists of two stages. The first stage is the face detection. We have chosen a simple algorithm of detection that does not have a lot of calculations and constraints. The algorithm eliminates the regions that do not have the skin color using the RGB space [viii,ix,x,xi] (Figure 1.b).This first stage comprises the following step: segmentation and binarization, determination of the coordinates of the face, and of the extraction face. A pixel is said a skin pixel, if the components R (Red), G (Green) and B (Blue) of the color satisfy the following conditions (segmentation and binarization) [viii,ix,x,xi]:

$$\boldsymbol{If}(R > 95)\ and\ (G > 40)\ and\ (B > 20)\ and$$
$$((\mathrm{Max}[R,G,B] - \mathrm{Min}[R,G,B]) > 15)\ and$$
$$(Abs\ (R - G) > 15)\ and\ (R > G) and$$
$$(R > B)\boldsymbol{then}$$
$$skin\ pixel\ (1)$$
$$\boldsymbol{Else}$$
$$not\ skin\ pixel\ (0)$$

After selecting the skin pixels, we obtain a binary image whose various-pixel values are equal to 0 or 1.The principle of determining the coordinates of the face is to achieve a vertical (1) one (VIb) and a horizontal projection (HIb) of the binary image.

$$HIb = \sum_{y=1}^{m} Ib(x,y) \qquad VIb = \sum_{x=1}^{n} Ib(x,y) \qquad (2)$$

We search the positions of the first two values that are different from the zero of both sides of the projection vectors HIb and VIb. These positions are the coordinates of the face in the original image (L1, C1, L2, C2) (Figure 1.a).

The second stage is the facial-component extraction. Several extraction methods use the color information [xiv,xviii]. These methods have two limitations. They work only with color images, so illumination changes can affect the results. There are some methods based on the gradient information that are robust to the illumination changes [xv,xvi,xvii]. We have chosen a method developed in [xii]. The method is based on the fact that human faces are constructed in the same geometrical configuration. To model the face with a geometric model, the method uses a gradient tracking to locate each component of a frontal face on the plain color background. Thus, it presents robustness against small rotation angles, lighting conditions, skin color or accessories (mustache, glasses, beards) [xii]. We search through this method the extraction of the two eyes, the mouth, and the forehead. The method uses a gradient tracking to locate each facial component [xii]. By exploiting the property of the face (since it is parameterized), it is easy to retrieve an object characteristic referring to its coordinates in the face with regards to the center of the face and the level of the eyes. To locate the facial components, we first determine their horizontal axes, and then we determine the area of each component by applying a geometric model [xiii]. This assumes that the vertical distances between the eyes and the nose and between the eyes and the mouth are proportional to the horizontal distance between the centers of the eyes [xiii]. Distances related to this model will be described later. The proposed method comprises the following steps:

- Grayscale conversion (Figure 2.a),

The grayscale RGB space transformation algorithm is based on the following formula [xiv]:

$$I = R * 0.299 + G * 0.587 + B * 0.114 \qquad (3)$$

where I expresses the gray level for a given pixel, and R, G and B are the color components of the pixel

- Calculating the gradient in the horizontal direction (Figure 2.c):

$$\nabla Iy = \frac{\partial I}{\partial \mathrm{y}}\vec{j} \qquad (4)$$

where y and j is the index and the horizontal direction, and I is the grayscale image.

- Eye level location (Neyes): Realizing the horizontal projection of the gradient image, and searching for the maximum horizontal projection (Figure 2.d).

$$HIy = \sum_{y=1}^{m} \nabla Iy(x,y) \qquad N_{eyes} = Max(HIy) \qquad (5)$$

- Location of the central axis (Ncenter): Search for the position of the highest gray level on the eye axis (Figure 2.b):

$$N_{center} = Max(\nabla Iy(N_{eyes}, y)) \qquad (6)$$

- Location of the nose (Nnose): Search for the highest gradient in a narrow window by horizontal projection, below the eyes axis and around the central axis with a width of Δx pixels (Figure 2):

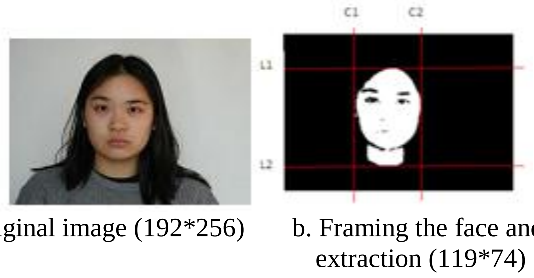
$$\boldsymbol{H_{N_{nose}}Iy} = \sum_{\boldsymbol{y=N_{center}-\Delta y}}^{\boldsymbol{y=N_{center}+\Delta y}} \boldsymbol{\nabla Iy(x + N_{eyes} + \Delta x, y)} \qquad (7)$$

$$N_{nose} = Max(H_{N_{\mathrm{nose}}}\ Iy) \qquad (8)$$

- Location of the mouth (Nmouth): the same principle as the previous step (Figure 2):

$$H_{N_{\mathrm{mouth}}}\ Iy = \sum_{y=N_{center}\ -\Delta y}^{y=N_{center}\ +\Delta y} \nabla Iy(x + N_{nose}\ + \Delta x, y) \qquad (10)$$

$$N_{\mathrm{mouth}} = Max(H_{N_{\mathrm{mouth}}}\ Iy) \qquad (9)$$



a. Original image (192*256) b. Framing the face and face extraction (119*74)

Figure 1: Detection and extraction of the face

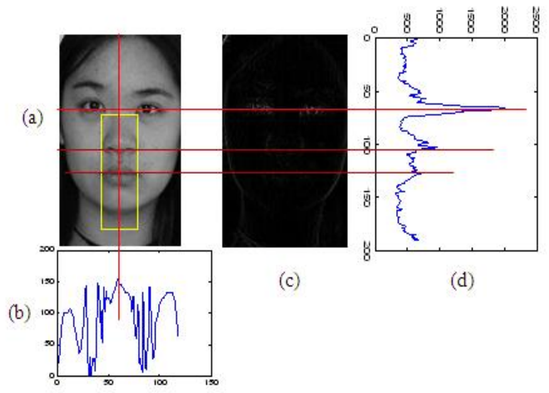


Figure 2: Location of the axes of the eyes, the nostril, the mouth, and the median

- Application of the geometric model (Figure 3).

The distance between the two-eye centers is D. The geometric face model (Figure 3) and the related distances are described below [xiii]: The vertical distance between the center of the mouth and the two eyes is D. The vertical distance between the center of the nostrils and the two-eyes is 0.6 D. The mouth width is D. The nose width is 0.8 D. The vertical distance between eyebrows and eyes is 0.4 D. with D = Neyes – Nmouth Once the eyes' axes and the mouth's axe are located, we apply the geometric model of the face [13]. Figure 3 shows the results obtained for the extraction of the facial components (eyes, forehead, and mouth).

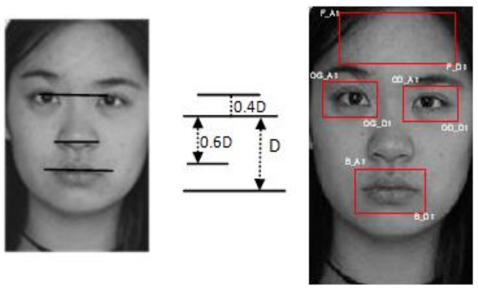


Figure 3: Application of the geometric model and extraction of the eyes, the mouth, and the forehead.

II.2. Vliw Processor Architecture Exploration Of Face Detection And Component Extraction Algorithm:

A. Trimaran and Its Development Environment:

Trimaran is a tool compilation and integrated performance analysis [xx,xxi]. This tool is used to search for adequate optimizations for the process of Instruction Level Parallelism (ILP). It is highly configurable; it can reach a wide range of architectures that can be targets of embedded processes and VLIW processors [xx]. Trimaran is particularly suited for research in an architecture and compiler-level tool. It also facilitates the architectural exploration by varying the ILP, we can determine the number and the types of the FU and the registers.

Trimaran is composed by: a parameterized architecture, called HPL_PD (Hewlett -Packard Labs PlayDoh) [xxi], and a Machine DEscription Service (MDES) to describe the HPL-PD architecture. A compiler with a large optimization technique. A configurable simulator using the MDES: It provides information on the execution time, the branch frequencies, and the resource use (operators, memory space). This information helps the designer to select the optimization technique. A graphical interface for configuring and operating the Trimaran tool [xx,xxi].

B. Exploration with Trimaran:

Before the exploration and simulation of the face detection and facial-feature component-extraction algorithm, the Trimaran tool replaces the function call with a copy of its text: this is the inlining optimization. The tool allows a choice of three technical optimizations: basic block, superblock and block hyper [xxi,xxii,xxiii].

Basic block: It is a sequence of instructions with a single input point (the first instruction) and a single output point (the last statement). In this case all instructions of the block are always executed. It is therefore possible to change the order of these instructions without worrying about the control flow of the algorithm. The weak point of this block type is that they have a small size because after five instructions, the potential parallelism is low [xxi].

Super block: It includes the identification of frequently executed blocks [xxii].

Hyper block: The super block has the same properties as a super block allowing having the instructions with a predicate, which may correspond to multiple paths. Longer blocks are interesting to increase the instruction-level parallelism and minimize the branch [xxiii].

III. Results and Tables

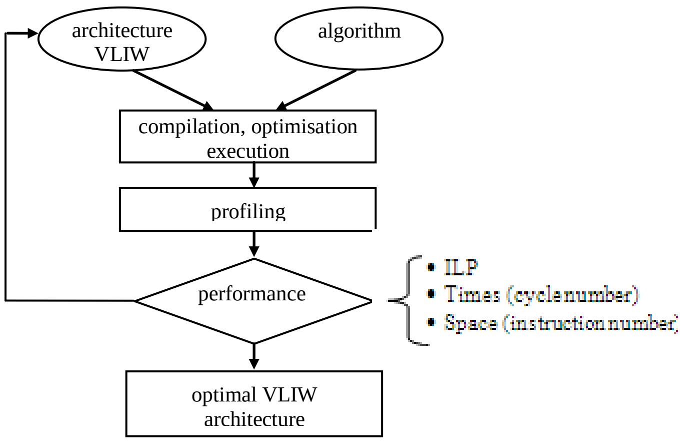

architecture VLIW
algorithm
compilation, optimisation execution
profiling
performance
ILP
Times (cyclenumber)
Space (instruction number)
optimal VLIW architecture

Figure 4: flow of algorithmic and architectural exploration

III.1. WLIW Architecture Exploration

Table1: Configuration of VLIW processor basis architecture

Parameter	*VLIW Architecture*
ISSUES	*4*
IALUs	*2*
FALUs	*2*
MU	*2*
BU	*1*
l1-icache (size,assoc,latency)	*32 KB,4,7*
l1-dcache (size,assoc,latency)	*32 KB,4,10*
l2-ucache (size,assoc,latency)	*32 KB,8,100*

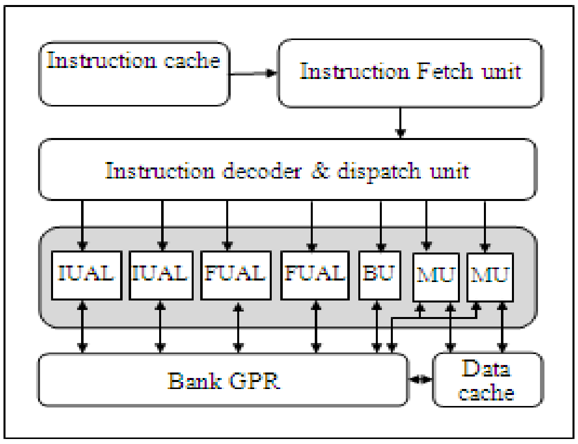


Figure 5: VLIW processor architecture basis.

Using the trimaran may determine the number of units used by the CLIW-dedicated processor. Figure 4 shows the methodology of exploring the VLIW processor architecture by using the compilation, the optimization, the profiling, the performance test, and the modification of the architecture. When performance is that we seek, we can determine the optimal VLIW processor architecture for the algorithm.

The initial configuration of the VLIW is as follows: two Integer Units (IALU), two Floating Units (FALU), two Memory Units (MU), and a Branch Unit (BU) (Table 1), Instruction cache, Data cache, Instruction Fetch Unit, Instruction decoder and dispatch Unit, Bank GPR (Figure 5).

We will use this initial architecture (Figure 5) and apply the different optimization technique. We seek the adequate technique for the algorithm. We realize an algorithmic and architectural exploration to determine the number of functional units necessary for the application execution. The choice of the optimization technique and the architecture depends on the execution performance: instruction number, cycle number and ILP
(Figure 4).

A. Exploration with a Basic Block

The figure 6 and table 2 gives a statistical for functions whose consumption of the execution time and the functional unit's number is important

The execution with the basic block optimization of the face-detection and facial feature extraction algorithm on the basis architecture (figure 5) requires 239,010 cycles, and 291,506 instruction with an ILP = 1.22. We note that the cycle's number of the IUAL is the highest (136,323 cycles) which represents 46.77%, while the cycle's number of the floating UAL is very low (only 3 cycles). Figure 6 shows the percentage for each instruction type of the application algorithm.

B. Exploration with a Hyper Block

With the hyper block, the total number of instructions is increased (318,005 Instructions) and the total cycle number has decreased to become 212,050 cycles. The ILP has risen and become 1.49. The implantation results by the hyper block are represented in the figure 6 and table 2.

C. Exploration with a Super Block

The implementation of a super block optimization gives us a total cycle's number equal to 223,489 cycles with an ILP equal to 1.47 (figure 6, table 2).

D. Comparative Study Of Different Optimization Technique

From Figure 6, we note that the algorithm does not require a large number of floating units, so we can reduce the number of these units and make it equal to 1. For the three optimization technique, the integer unit percentage is the highest.

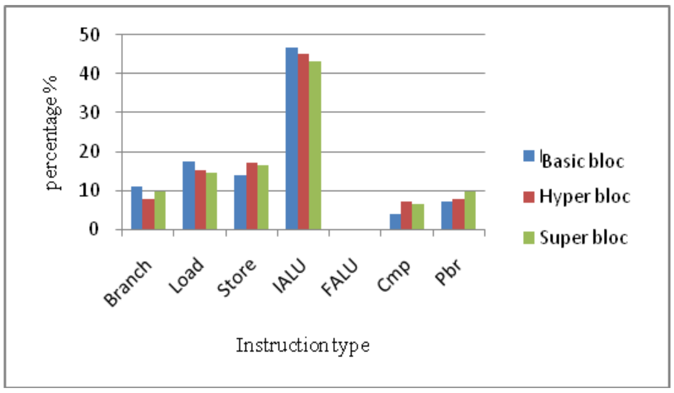


Figure 6: Percentage of different instruction for different optimization technique.

Table 2: Performance of optimization technique of the algorithm with basis architecture

	Cycle number	Instruction number	ILP
Basic block	239,010	291,506	1.22
Hyper block	212,050	318,005	1.49
Super block	223,489	329,115	1.47

According to Table 2, the hyper block is the best and adequate optimization technique for the algorithm. In terms of performance, it presents a minimum cycle number and a maximum ILP. So we chose and applying this technique to determine the optimal architecture of dedicated VLIW processor.

III.2. Architectural Exploration of the Algorithm Based On Hyper Block Optimization

In the earlier study, it was noted that the algorithm did not require a large number of floating units; thus, we will vary the number of different units, except the floating units which is always equal to 1 (table 3).

Table 3: Measuring performance based on a functional units number.

FU number IALU/FALU/MU/BU	Total cycle number	ILP	Instruction number
2/2/2/2	212,050	1.49	318,005
2/1/2/2	212,050	1.49	318,005
3/1/3/3	211,694	1.50	318,005
4/1/3/3	**211,158**	**1.50**	**318,005**
5/1/3/3	211,158	1.50	318,005
5/1/4/4	211,158	1.50	318,005
5/1/5/5	211,158	1.50	318,005
6/1/6/6	211,158	1.50	318,005

By varying the number of units, the total number of cycles ranging from 212,050 cycles for the architecture 2/2/2/2 to 211,158 cycles for the architecture 4/1/3/3. From the latter, the modification of the architecture does not affect the total cycle's number. The parallelism rate (ILP) stabilizes at 1.50 from the architecture 4/1/3/3. The instruction number has stabilized at 211,158 instructions from the same architecture (table 3).

Table 4: configuration of optimal VLIW Processor architecture

Parameter	VLIW Architecture
ISSUES	4
IALU	4
FALU	1
MU	3
BU	3
l1-icache (size,assoc,latency)	1 MB,4,7
l1-dcache (size,assoc,latency)	512 KB,4,10

Table 5: Performance of algorithm on the optimal VLIW processor

Performance	value
Unit number	4/1/3/3
ILP	1.50
Cycle number	211,158
Execution times (frequency = 1MHz)	211.158 µs
Instruction number	318,005
Instruction memory size	1 Mo
Data memory size	512 Ko

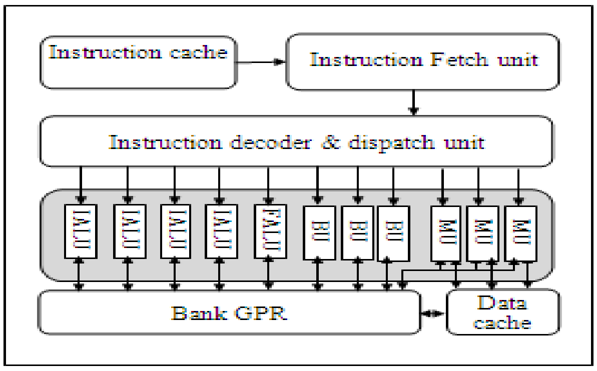


Figure 7: Optimal dedicated VLIW Processor Architecture

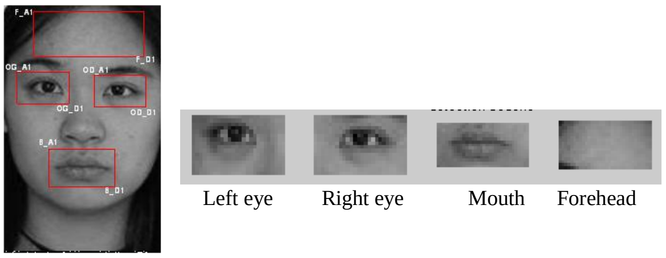


Figure 8: extraction of the eyes, the mouth, and the forehead.

Table 4 present the necessary hard resource (UFs, Memory size) used for the optimal architecture of the dedicated VLIW processor (figure 7) to the face-detection and facial-feature and component extraction application.

The performances of the application algorithm implementation on the dedicated VLIW processor (figure 7) are presented in table 5. The algorithm execution time is 211.158µs with a frequency of 1MHz. the results of face detection and facial components extraction (left eye, right eye, mouth and forehead) are shown in figure 8.

IV. Conclusion

In this application we use two algorithms that have reduced complexity with a maximum performance: The face detection algorithm is based on the use of the colour information and The algorithm of localizing the facial components. The latest is based on the gradient which provides robustness against illumination changes and geometric model. We are explored and designed a dedicated VLIW processor architecture for the application algorithm. We are exploited the potential-parallelism algorithm, the data-dependence-degree algorithm and the memory-management-by-exploring-the-adequacy-architecture algorithmic. The optimal architecture of dedicated VLIW processor allows us to process an image on 211.158 µs with a frequency of 1GHz. This enables us to consider the implementation of other algorithmic parts of analyzing the facial expressions.

References

i. Junguk Cho, ShahnamMirzaei, Jason Oberg, Ryan Kastner, "Fpga-based face detection system using Haar classifiers" Proceeding of the ACM/SIGDA international symposium on Field programmable gate arrays, California, USA, Pages 103-112, 2009

ii. Nicolas Farrugia, Franck Mamalet, S´ebastien Roux, Michel Paindavoine, Fan Yang, (2007) "F Implantation parallèle de la detection de visages : Méthodologie et implantation sur FPGA". Colloque GRETSI, pp 573-576, Troyes, 11-14 septembre 2007.

iii. Mohammad S. Sadri, Nasim Shams, MasihRahmaty, IrajHosseini, ReihaneChangiz, ShahedMortazavian, ShimaKheradmand, RoozbehJafari, (2004) "An FPGA Based Fast Face Detector", GSP 2004 pp. 1039, 2004.

iv. Paschalakis, S., Bober, M., (2003) "A Low Cost FPGA System for High Speed Face Detection and Tracking", In Proc. 2003 2nd IEEE International Conference on Field Programmable Technology (FPT '03), Tokyo, Japan, Dec. 15-17, pp. 214-221, 2003.

v. L. PIERREFEU, J. JAY., (2007) "Détection et localisation de visage pour un système grand public d'authentification de visage implantable sur FPGA". Colloque GRETSI, Troyes, pp 361-364, 11-14 septembre 2007.

vi. T.Theocharides, G.Link, N.Vijaykrishnan, M.J.Irwin, and W.Wolf, (2004). "Embedded hardware face detection" Proceedings of the 17th International Conference on VLSI Design (VLSID 04), 2004.

vii. Smach, .M Atri, J. Mitéran and M. Abid, (2006) "Design of a Neural Networks Classifier for Face Detection" Journal of Computer Science 2 (3): 257-260, 2006.

viii. Garcia C. ,Tziritas G., (1999) "Face Detection Using Quantized Skin Color Regions Merging and Wavelet Packet Analysis" IEEE Transactions on Multimedia, 1(3), p.264-277, September 1999.

ix. BencherietChemesse-Ennehar, BouallegAbd El halim, Tebbikh Hicham, (2007) "Segmentation de la Couleur de Peau par Seuillage selon Différents Espaces de Couleur" JIG'2007 - 3èmes Journées Internationales sur l'Informatique Graphique, pp 207-211, INRIA Sophia-Antipolis, France, 2007.

x. F. Gasparini and R. Schettini, (2006) "Skin segmentation using multiple thresholding in Internet Imaging" vol. 6061 of Proceedings of SPIE, pp. 1–8, San Jose, Calif, USA, January 2006.

xi. J. Kovac, P. Peer, and F. Solina, (2003) "2D versus 3D colour space face detection" in Proceedings of the 4th EURASIP Conference on Video/Image Processing and Multimedia Communications, vol. 2, pp. 449–454, Zagreb, Croatia, July 2003.

xii. F.ABDAT, C.MAAOUI et A.PRUSKI, (2007) "Suivi du gradient pour la localisation des caractéristiques faciales dans des images statiques" Colloque GRETSI, Troyes 11-14 septembre 2007.

xiii. Frank Y. Shih, Chao-Fa Chuang, (2004) "Automatic extraction of head and face boundaries and facial features" Information Sciences 158 pp. 117–130, 2004.

xiv. Michael J. JONES and James M. REHG. (1999) "Statistical Color Models with Application to Skin Detection" In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, volume 1, page 1274, Fort Collins, Colorado, June 1999.

xv. Deng X., Chang C. H. and Brandle E. (2004) "A new method for eye extraction from facial image". Proc. 2nd IEEE international workshop on electronic design test and applications (DELTA), 2 NO.4:29–34, Perth, Australia, 2004.

xvi. Maio D. and Maltoni D. (2000) "Real-time face location on grayscale static images" Pattern Recognition, 33:1525– 1539, 2000.

xvii. Tsekeridou S. and Pitas I. (1998) "Facial feature extraction in frontal views using biometric analogies" Proc. 9th European Signal Processing Conference, 1:315–318, September 8-11 Island of Rhodes, Greece, 1998.

xviii. *Hui-Yu Huang, Yan-Ching Lin, An Approach for Mouth Localization Using Face Feature Extraction and Projection Technique, Smart Innovation, Systems and Technologies Volume 21, pp 247-257, 2013*

xix. *Nadia, M Bouraoui, B Mohamed, Med Hedi BEDOUI , FPGA architecture for facial features and components extraction, International Journal of Computer Science, Engineering and Information Technology (IJCSEIT), Volume3, Numéro 2, Pages 51-64, 2013*

xx. *Ranjan Kumar Behera, Deepak Kumar, K. S. Pandey, Concept, Design and Performance Evaluation of VLVIW Processor, International Journal of Scientific Engineering and Technology, Volume No.2, Issue No.7, pp : 719-723, 2013.*

xxi. *Lakshmi N. Chakrapani, John Gyllenhaal, Wen-mei W. Hwu,Scott A. Mahlke, Krishna V. Palem, and Rodric M. Rabbah, Trimaran: An Infrastructure for Research in Instruction-Level Parallelism. Springer-Verlag Berlin Heidelberg 2005, p: 32-41,2005*

xxii. *W. Hwu, S. Mahlke, W. Chen, P. Chang, N. Warter, R. Bringmann, R. Ouellette,R. Hank, T. Kiyohara, G. Haab, J. Holm, and D. Lavery. The superblock: An effective technique for VLIW and superscalar compilation. Journal of Supercomputing,Jan. 1993.*

xxiii. *S. Mahlke, D. Lin, W. Chen, R. Hank, and R. Bringmann. Effective compiler support for predicated execution using the hyperblock. In Proceedings of the 25thAnnual International Symposium on Microarchitecture, pages 45–54, Dec. 1992.*