A LOW-POWER VLSI ARCHITECTURE FOR FACE VERIFICATION USING ELASTIC GRAPH MATCHING

Jean-Luc Nagel, Patrick Stadelmann, Michael Ansorge and Fausto Pellandini
Electronics and Signal Processing Laboratory,
Institute of Microtechnology, University of Neuchâtel,
Rue A.-L. Breguet 2, CH-2000 Neuchâtel, Switzerland
Phone: +41 32 718 34 46; Fax: +41 32 718 34 02
e-mail: jean-luc.nagel@unine.ch

ABSTRACT

This paper introduces a novel low-power VLSI architecture dedicated to algorithms based on elastic graph matching. The targeted application is face verification for low-power mobile devices (e.g. mobile phones, personal digital assistants, wearable computing devices). A description of the overall verification system is provided jointly to a detailed discussion of the full-custom graph matching coprocessor.

1. INTRODUCTION

This paper presents a low-power VLSI architecture for Elastic Graph Matching (EGM) algorithms applied to biometric face verification. By carefully partitioning the EGM algorithms and optimizing the implementation resources, the designed low-power VLSI architecture features an optimal dataflow regularity, avoids computational redundancies through data reuse, and efficiently exploits the parallelism so that the operating frequency and supply voltage can be lowered, which minimizes power consumption. Moreover, the proposed architecture is modular, and can be adapted to process other biometric modalities.
The paper is organized as follows. Section 2 shortly recalls properties of EGM algorithms, with focus on algorithms based on morphological feature extraction. The proposed VLSI architecture is then described in Section 3, considering first the whole face verification system, followed by a detailed discussion of the graph matching coprocessor, including data flow organization, selected instruction set, and complexity estimation. The conclusions are finally provided in Section 4.

2. ELASTIC GRAPH MATCHING

Elastic graph matching was chosen as the base algorithm for face verification due to its intrinsic compensation for face expression variations and small pose variations [1][2]. In [3] EGM is comparing advantageously to other methods such as eigenfaces or neural network based methods.
EGM is a holistic method relying on a *labeled graph* structure (Fig. 1) composed of local features centred on a set of *nodes* connected by *edges*. A correspondence is searched between features of a reference graph (which is orthogonal in our case) and features of a test graph, which minimizes the *euclidean distance* between these features. Node locations of the test graph can be individually displaced from their orthogonal arrangement (thus the name "elastic" matching) in order to further minimize the distance score, where each node displacement is in turn penalizing the score so as to take into account the topological structure. The correspondence or *graph matching* is done in two phases as described in [1]: 1) *rigid matching* is carried out by moving the rigid graph - including possible rotation - in a given window of the image, while 2) *elastic matching* is performed by displacing nodes around their best rigid position in a pseudo-random order.

2.1 Morphological Elastic Graph Matching

The features extracted at each node represent a local information on the pixel neighborhood. In [1], Gabor filtered images are used to extract texture information. These filters provide well localized features in both spatial and frequency domains of the image, but they are computationally expensive. Another type of features, based on mathematical morphology, is used by Kotropoulos et al. [4], the face image being successively dilated and eroded by progressive circular structuring elements (SE). These features are more illumination dependent than Gabor features, but require simpler computations. For instance using binary SEs on grayscale images, erosion (dilation) merely consists in finding the minimum (maximum) value on the support of the SE. The architecture for EGM presented in this paper was tested with 19 features extracted at each node, corresponding to the local pixel value, and to 9 dilations and 9 erosions obtained with SEs ranging from 3×3 to 19×19 pixels (Fig. 1). A detailed description can be found in [5]. It should be noted that face verification is performed using luminance images, whereas colour images are generally used for efficient face detection [7].
Hence 19 features are associated to each of the 64 nodes of the graph. Following the “curse of dimensionality” postulate [8], it is known that too many features can decrease the verification rate. The 19 features are thus reduced to 3 *most expressive features* using a Principal Component Analysis (PCA) [9]. These features are further transformed into 3 *most discriminating features* using a Linear Discriminant Analysis (LDA) [9]. Even after LDA, the total number of features on all nodes is too high to use e.g. an N-dimensional Bayesian classifier (N=64·3). But assuming simplifications, namely that the likelihood of a class is normally distributed in the feature space, and that the features are independent, implying that the covariance matrix of the features is diagonal, then the minimum distance classifier is known to be the euclidean distance. The classification is then considered as a two-class separation problem which can be handled with a Bayesian classifier.

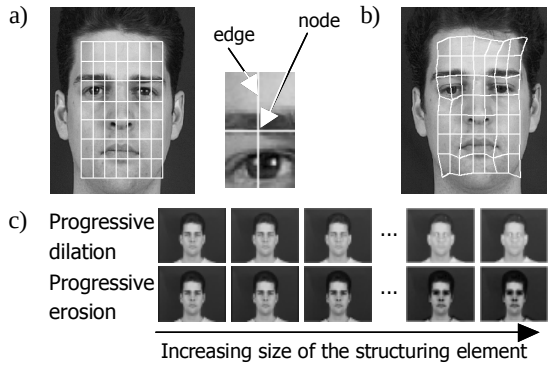

Figure 1: *a) Reference grid; b) Matched grid; c) Multiscale dilation and erosion.*

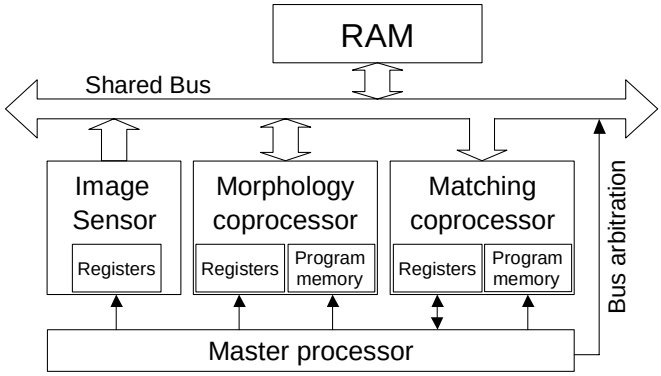

Figure 2: *General system architecture.*

3. EGM ARCHITECTURE

High-level tasks such as classification are often irregular and difficult to process in parallel, so that implementing them on general purpose digital signal processors seems best suited compared to designing dedicated hardware. However, for the considered algorithm, it is worth partitioning the tasks into morphology and graph matching, and assigning these tasks to two specialized coprocessors. This solution allows a highly regular and parallel internal structure of the coprocessors, modularity, optimal data reuse and dataflow organization, low system clock frequency, and low-power consumption.

3.1 System description

The face verification system (Fig. 2) is composed of a CMOS image sensor (e.g. 320×240 pixels), a master processor for high-level tasks (classification, database management, interface control, initialization), a shared data RAM, and two coprocessors dedicated to multiscale morphology [5] and graph matching, respectively. All these components are connected to a shared standard bus (e.g. AMBA [6]) simplifying the insertion of extra “IP-blocks” (e.g. a face detection coprocessor). The whole system could be realized as a System-on-Chip (SoC) to minimize the number of off-chip interconnections, which are a major source of power dissipation. The distribution of morphology and graph matching computation over 2 coprocessors was made because the related operators are completely different, morphology requiring comparison calculations, whereas feature reduction and distance computation involve multiply-accumulate (MAC) operations. Separating these tasks greatly simplifies the sequencing of both coprocessors. Moreover, it is also possible to replace the morphology coprocessor in order to extract other kinds of local features (e.g. Gabor wavelets), thus exploiting the system modularity.
The system components are all slaves of the master processor, which can read from - and write to - local memories and registers of these components (memory mapped registers). Whenever notified by the master processor, each slave (sensor, coprocessors) becomes momentarily master of the shared memory. The master processor holds thus also the role of bus arbiter. Moreover, the same system clock signal drives all components to further simplify the synchronization.

3.2 Matching coprocessor

The matching coprocessor (Fig. 3b) is responsible for PCA and LDA feature reduction, euclidean metric calculation between features, and deformation penalty calculation. All these operations involve MAC operations and therefore can be highly accelerated on an architecture providing vectorized MAC units, and optimized memory and register structures for local data reuse. The coprocessor architecture is divided into an addressing unit (Fig. 3a) and vector MAC units (Fig. 3c). A program memory embedded in a local RAM, and an instruction decoder providing hardware support for nested loops, allow sequencing modifications, thus improving the global modularity. For instance it is possible to modify the pseudo-random order of the nodes during elastic matching, the number of elastic iterations, and also the number of nodes and features (although the latter are limited to a maximum of 64 nodes and 19 features due to the number of registers and their size).
The Addressing Unit (AU, Fig. 3a) contains two pairs of register banks holding the (x,y) coordinates of the "current" - i.e. currently processed - node positions, and the "best" node positions found so far. Every time the euclidean distance related to the "current" node positions outperforms the one of the "best" node positions found, the latter become overwritten by the "current" node positions. An adder and a set of immediate values are used to increment the indexes needed for external memory addressing, namely the morphology level index, node index, and node coordinates.
The MAC units (e.g. MAC Unit 3 in Fig. 3c) comprise a multiplier, an adder, and a local register bank that can perform one-cycle MAC operations. Two local memories furnish the PCA/LDA coefficients and reference feature values. These memories dispose of an embedded output register and are accessed using indirect addressing. When data are read, the next data are already fetched internally to the memories, while the index is incremented. This structure is optimal to increase sequencing regularity and parallelism, whereas the full cycle time is available for memory accesses as well as for index incrementation, allowing for slower low-power components. Register banks are local to each MAC unit in order to minimize the MUX tree, except for a shared register (Fig. 3c) that can be read by the MAC Unit 1 (Fig. 3b) to sum the partial feature distances.

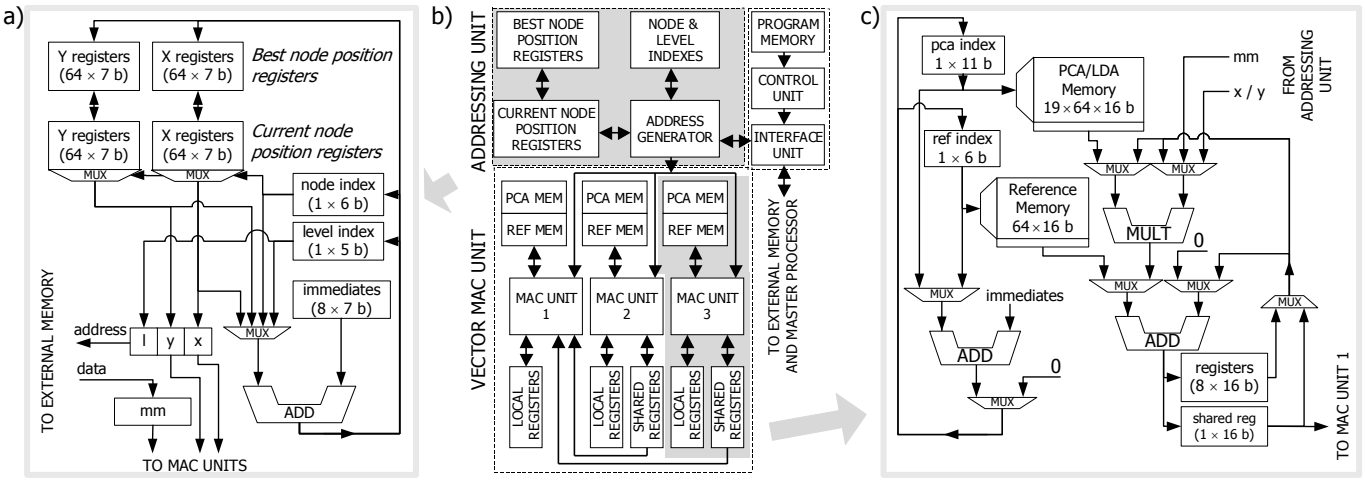

Figure 3: *a) Detail of the addressing unit; b) General organization of the matching coprocessor; c) Detail of a MAC unit (MAC UNIT 3).*

3.3 Data flow

The AU is the only block of the coprocessor to access the external RAM (Fig. 3a,b), where the external address is specified by the concatenation of the morphology level index, and the node coordinates which are indirectly addressed by the node index (Fig. 3a). During the rigid matching phase, the AU fetches the morphological features from the external RAM, and transfers them to the MAC units for feature reduction and euclidean metric calculation, while handling the rigid graph nodes displacement in the image window. The elastic graph matching phase is processed according to Subsection 3.3.3.

3.3.1 Feature reduction

The feature vector of a node is loaded element by element from the external RAM by incrementing the level index, and is sent to the three MAC units in order to compute the three reduced features in parallel. During each processing cycle, a feature value is multiplied by a PCA/LDA coefficient with accumulation of the result in the local register bank.

3.3.2 Euclidean metric

Only the MAC units are used for metric calculation. Reference features loaded from local memories are subtracted from the test features stored in the local registers (Fig 3c). The differences are then squared and accumulated in the shared registers. The special MAC Unit 1 (Fig. 3b) finally adds up the values from the shared registers to determine the feature distance. In case this distance is lower, the "current node position" registers are copied in parallel to the "best node position" registers.

3.3.3 Deformation penalty

During the elastic graph matching phase, the AU displaces the nodes coordinates individually in their respective neighborhood, and transfers the (x,y) values to MAC Units 2 and 3 to calculate the deformation penalty in a supplementary step. The deformation penalty is achieved by summing the squared distance variation of every node to its four connected neighboring nodes when passing from the rigid to the elastic graph [1]. The results are then transferred to MAC Unit 1 for final summation and computation of the total deformation penalty.

3.4 Instruction set

Each instruction is 14 bits wide, with two fields controlling the AU (6 bits) and the MAC units (7 bits), respectively, and an extra bit reserved for special loop instructions. The latter is used to indicate that the next instructions until a corresponding special branch instruction are to be repeated n times, or that the next cycle is a branch cycle. This mechanism minimizes pipeline stalls, since only loop instructions create a one-cycle delay when stacking the loop index and branch address.

Addressing Unit related instruction field (6 bits wide)			
ldnp	*Load pixel indirectly from external memory to mm register* [1]	incn	*Increment node index*
movn	*Move current node position to MAC units* [2]	setl	*Set level index* [3]
movb	*Move all best nodes to current*	setn	*Set node index* [3]
movbi	*Move best node to current indirectly* [2]	inccx inccy	*Increment node position indirectly* [2,3]
MAC Units related instruction field (7 bits wide)			
MAC Units 1, 2, and 3			
multv	*Vector multiplication* [4]	squarv	*Vector square* [5,6]
macv	*Vector multiply and accumulate* [4]	asquarv	*Vector square and accumulate* [5,6]
subv	*Vector subtraction* [7]		
MAC Units 2 and 3			
subcv	*Vector subtraction* [8]	addcv	*Vector addition* [8]
MAC Unit 1 and special instructions			
compd	*Compare distance and move all current nodes to best nodes if smaller*	compdi	*Compare distance and move indexed current node to best if smaller* [2]
sets	*Set score* [3]	setp	*Set pca index* [3]
setr	*Set reference index* [3]	setm	*Set MAC mode* [3,9]
incr	*Increment reference index* [3]	incp	*Increment pca index* [3]
add	*Add shared registers*		

[1] *Level is automatically incremented.*
[2] *Referenced by node index.*
[3] *Limited number of immediate values.*
[4] *One operand from PCA/LDA memory, the other from addressing unit.*
[5] *Source and destination from regs.*
[6] *Use upper or lower 8 bits.*
[7] *One operand from ref. memory.*
[8] *One operand from nodes position.*
[9] *Mode: saturation, padding, rational, upper/lower bits.*

Table 1: *Instruction set.*

Static program size (# of instructions)	557
rigid matching	*49*
elastic matching	*508*
Number of executed cycles	6'746'186
rigid matching	*3'777'864*
elastic matching	*2'968'322*
Number of executed operations	
additions	*20'684'678*
multiplications	*14'992'000*
external loads	*4'575'808*

Table 2: *Complexity figures of the graph matching coprocessor corresponding to a complete face verification.*

MAC instructions are SIMD-like since the same instruction can be executed either on all 3 MAC units, or on 2, or on a single one, depending on the instruction type (cf Table 1). Also, the instruction set provides the architecture with a certain reconfiguration flexibility.

3.5 Complexity estimation

The presented architecture was validated at register transfer level using in-house tools (assembler, simulator) developed in C++, which are at the same time used to estimate the complexity of the algorithm by counting the total number of cycles executed for a single face verification. The results in Table 2 correspond to one face verification using 2552 rigid matching iterations applied to a 128×128 pixels search window, and 10 elastic matching iterations, where all nodes are successively displaced in a 11×11 pixels neighborhood during each elastic matching iteration. Figures related to the VLSI circuit area and to a refined power consumption estimation are foreseen for a later stage of the work.

3.5.1 Simulation

The simulator contains a parser written in GNU Flex and Bison [10], that reads the source code counting 557 program instructions, where the instruction set is used as a base for the Bison grammar. Additionally, the coprocessor architecture is specified by instanciating a set of basic components described in C++, jointly to the corresponding interconnections. The simulation is then performed by dispatching every instruction to the concerned components, followed by a bit-true and cycle-true execution. The correctness of the program was checked this way, collecting as well statistical data on resource usage (memory, registers, multipliers, adders), and verifying the absence of any resources dependency problem.

3.5.2 Results

Following [5], the calculation of the 19 morphology levels on the 128×128 window requires ca 2.3 million cycles. Adding the 6.8 million cycles reported in Table 2, amounts to less than 10 million cycles to perform a complete face verification, excluding image acquisition and face detection. Assuming an operating frequency of 10 MHz, the face verification is achieved in less than 1 second, whereas timing constraints in the architecture can be strongly released, and the power consumption drastically reduced, thanks to the low system clock frequency.

4. CONCLUSIONS

This article presents a dedicated low-power architecture for face verification using elastic graph matching algorithms. The system is subdivided into two specialized coprocessors, which leads to an optimized computation and dataflow regularity. An efficient algorithm decomposition is thus rendered possible, allowing a low-power VLSI implementation by avoiding computational redundancies through data reuse, and by carefully exploiting the parallelism, so that the system clock frequency can be lowered. A specific coprocessor for elastic graph matching was then described jointly to its parametric instruction set. Finally, a complexity figure corresponding to a measure of the requested instruction cycles was given, showing that a face verification can be performed in less than 1 second even with low system clock frequencies.

Future work would consist in describing the architecture in VHDL, and in synthesizing the coprocessor in a standard low-power CMOS technology so as to achieve precise area and power consumption estimations. Moreover, new architectures for elastic graph matching are foreseen to be investigated, including application to complementary biometric modalities.

ACKNOWLEDGEMENTS

This work was part of the *SmartPix* project supported by the Swiss Center for Electronics and Microtechnology, Inc. (CSEM), Neuchâtel and Zurich, Switzerland, under Grant OIH3. Also, the authors would like to thank Dr. C. Kotropoulos, Aristotle University of Thessaloniki, Greece, for his helpful comments on Morphological Elastic Graph Matching.

REFERENCES

[1] M. Lades, J. C. Vorbrüggen, J. Buhmann, J. Lange, C. von der Malsburg, R. P. Würtz, and W. Konen, "Distortion Invariant Object Recognition in the Dynamic Link Architecture", *IEEE Trans. on Computers*, Vol. 42, No. 3, March 1993, pp. 300-311.

[2] L. Wiskott, "Labeled Graphs and Dynamic Link Matching for Face Recognition and Scene Analysis", *Ph.D. Thesis*, University of Bochum, Germany, July 1995.

[3] J. Zhang, Y. Yan and M. Lades, "Face Recognition: Eigenface, Elastic Matching, and Neural Nets", *Proc. of the IEEE*, Vol. 85, No. 9, September 1997, pp. 1423-1435.

[4] C. Kotropoulos, A. Tefas, I. Pitas, "Frontal Face Authentication Using Discriminating Grids with Morphological Feature Vectors", *IEEE Trans. on Multimedia*, Vol. 2, No. 1, 2000, pp. 14-26.

[5] P. Stadelmann, J.-L. Nagel, M. Ansorge and F. Pellandini, "A Multiscale Morphological Coprocessor for Low-Power Face Authentication", submitted to *EUSIPCO 2002*, Toulouse, France.

[6] "AMBA Specification, rev. 2.0", *ARM Limited*, United Kingdom, http://www.arm.com, May 1999.

[7] K. Sobottka and I. Pitas, "A Novel Method for Automatic Face Segmentation, Facial Features Extraction and Tracking", *Signal Processing: Image Communication*, Vol. 12, No. 3, June 1998, pp. 263-281.

[8] K. Fukunaga, "Introduction to Statistical Pattern Recognition", 2nd edition, Academic Press Inc, San Diego, CA, USA, 1990.

[9] D. L. Swets and J. Weng, "Using Discriminant Eigenfeatures for Image Retrieval", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 8, August 1996, pp. 831-836.

[10] J. R. Levine, T. Mason, D. Brown, "Lex & Yacc", 2nd edition, O'Reilly & Associates Inc., Sebastopol, CA, USA, 1992.