Implementation of an RBF Neural Network on Embedded Systems: Real-Time Face Tracking and Identity Verification

Fan Yang and Michel Paindavoine

***Abstract*—This paper describes a real time vision system that allows us to localize faces in video sequences and verify their identity. These processes are image processing techniques and the radial basis function (RBF) neural network approach. The robustness of this system has been evaluated quantitatively on eight video sequences. We have adapted our model for an application of face recognition using the Olivetti Research Laboratory (ORL), Cambridge, U.K., database so as to compare performance against other systems. We also describe three hardware implementations of our model on embedded systems based, respectively, on field programmable gate array (FPGA), zero instruction set computer (ZISC) chips, and digital signal processor (DSP) TMS320C62. We analyze the algorithm complexity and present results of hardware implementations in terms of resources used and processing speed. The success rates of face tracking and identity verification are, respectively, 92% (FPGA), 85% (ZISC), and 98.2% (DSP). For the three embedded systems processing speeds for images size of 288 $\times$ 352 are, respectively, 14 images/s, 25 images/s, and 4.8 images/s.**

***Index Terms*—Digital signal processor (DSP), face localization and identity verification, field programmable gate array (FPGA) device, radial basis function (RBF) neural networks, real-time implementation, zero instruction set computer (ZISC) chip.**

I. Introduction

HUMAN face recognition is an active area of research spanning several disciplines such as image processing, pattern recognition, and computer vision. Different techniques can be used to track and process faces [1], e.g., neural networks approaches [2]–[5], eigenfaces [6]–[8], and the Markov chain [9]. Most researches have concentrated on the algorithms of segmentation, feature extraction, and recognition of human faces, which are generally realized by software implementation on standard computers. However, many commercial and law enforcement applications of human face recognition such as human-computer interfaces, model-based video coding, and security control [10]–[12] need to be high-speed and real-time, for example, passing through customs quickly while ensuring security.

Liu *et al.* [13] realized an automatic human face recognition system using the optical correlation technique after necessary preprocessing steps. Buhmann *et al.* [14] corrected changes in lighting conditions with an analog very large scale integration (VLSI) silicon retina in order to increase the face recognition rate. Matsumoto and Zelinsky [15] implemented in real time a head pose and gaze direction measurement system on the vision processing board Hitachi IP5000. Our aim is to implement on embedded systems an efficient model of unconstrained face tracking and identity verification in arbitrary scenes. Thus, we would elaborate a robustness algorithm that requires moderated computation.

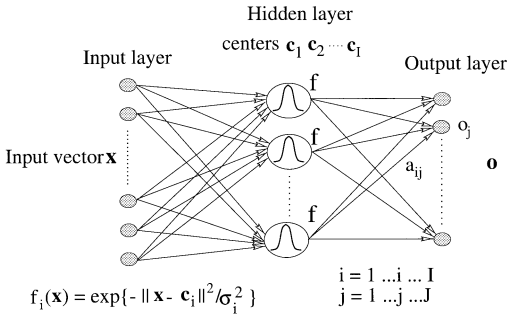


Fig. 1. Radial basis function neural network.

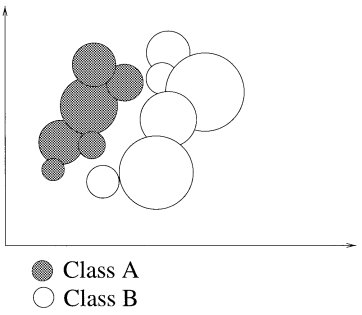


Fig. 2. Decision region mapping in a 2-D space.

Rosenblum *et al.* [16] developed a system of human expressions recognition from motion based on an radial basis function (RBF) neural network architecture. Ranganath *et al.* [17], [18] performed an integrated automatic face detection and recognition system using the RBF networks approach. Howell and Buxton [19] compared RBF networks with other neural network techniques on a face recognition task for applications involving identification of individuals using low-resolution video information. The RBF networks give performance errors of only 5%–9% on generalization under changes of orientation, scale, pose, and lighting. Their main advantages are computational simplicity and robust generalization. Howell and Buxton showed that the RBF network provides a solution which

Manuscript received September 15, 2002; revised March 15, 2003.

The authors are with the Laboratoire LE2I, Aile de l'Ingénieur, MirandeUniversité de Bourgogne, BP 400-21011 Dijon Cédex, France (e-mail: fanyang@u-bourgogne.fr; paindav@u-bourgogne.fr).

Digital Object Identifier 10.1109/TNN.2003.816035

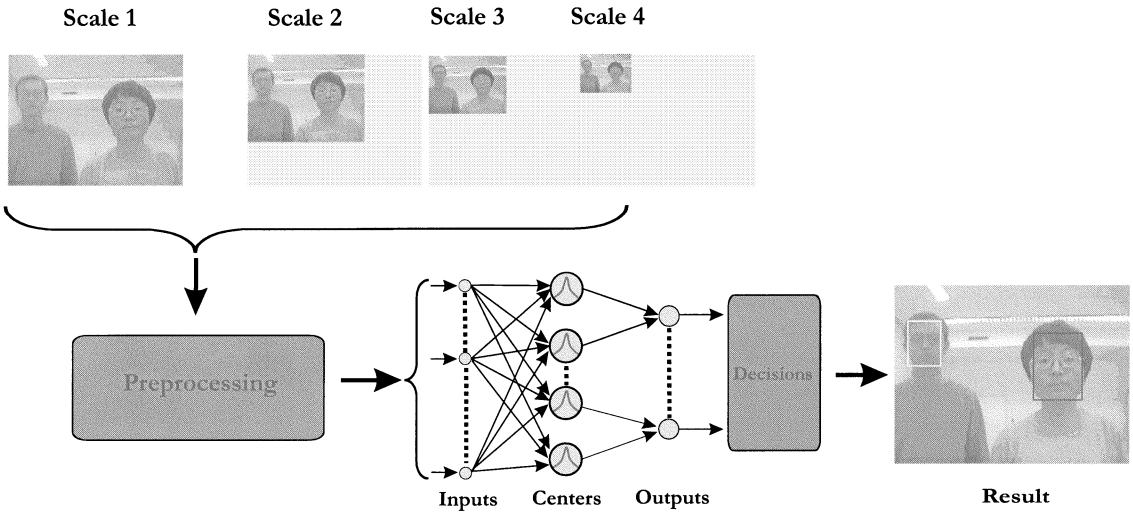


Fig. 3. Structure of the face tracking and identity verification model.

Fig. 4. 2×12 learning faces.

can process test images in interframe periods on a low-cost processor. The simplicity and the robust generalization of the RBF networks approach, with its advantages due to the fact that it can be mapped directly into the existing neural networks chips lead us to elaborate our model using a RBF classifier.

We chose three commercial embedded systems for hardware implementations of face tracking and identity verification. These systems are based, respectively, on most common electronic devices: FPGA, zero instruction set computer (ZISC) chips, and digital signal processor (DSP) TMS320C62. We obtained processing speeds of, respectively, for three implementations: 14 images/s, 25 images/s, and 4.8 images/s.

This paper is organized into two parts: 1) description of the model and 2) hardware implementations. Section II presents the RBF network principle and explains the face tracking and identity verification scheme. Section III provides the experimental results, and we compare our model against other methods using the Olivetti Research Laboratory (ORL), Cambridge, U.K., face database [20]. Section IV analyzes the algorithm complexity, and Section V describes the three hardware implementations and discussion. Section VI concludes the paper.

II. Description of the Model

A. Radial Basis Function Neural Networks

The RBF neural network [21], [22] has a feedforward architecture with an input layer, a hidden layer, and an output layer as shown in Fig. 1. The input layer of this network has N units for an N-dimensional input vector. The input units are fully connected to the I hidden layer units, which are in turn connected to the J output layer units, where J is the number of output classes. RBF networks belong to the category of kernel networks. Each hidden node (unit) computes a kernel function on input data, and the output layer achieves a weighted summation of the kernel functions. Each node is characterized by two important associated parameters: 1), its center and 2) the width of the radial function. A hidden node provides the highest output value when the input vector is close to its center and this output value decreases as the distance from the center increases. Several distances can be used to estimate the distance from a center but the most common is the Euclidean distance

$$d(\mathbf{x}) = \|\mathbf{x} - \mathbf{c}_i\|. \tag{1}$$

The activation function usually chosen of the hidden node is a Gaussian function

$$f_i(\boldsymbol{x}) = \exp^{(-d(\boldsymbol{x})^2/\sigma_i^2)} \tag{2}$$

such that each hidden node is defined by two parameters: its center $\boldsymbol{c_i}$ and the width of the radial function σ_i.

B. Training Procedure of RBF Neural Networks

The training procedure undergoes a two-step decomposition: estimating $\boldsymbol{c_i}$ and σ_i and estimating the weights between the hidden layer and output layer.

Estimate $\boldsymbol{c_i}$ and $\boldsymbol{\sigma_i}$: Conventionally, the unsupervised k-means clustering algorithm can be applied to find I clusters from the set of training vectors. However, the difference in the facial appearance of one person due to a change of pose are greater than that person's difference from another person. In

Fig. 5. Some results of face tracking and identity verification.

TABLE I
RESULTS: VARIATION OF THE INPUT VECTOR LENGTHS USING THE FIRST FOUR SEQUENCES (SEE FIG. 5)

	Number components	Number of faces to be verified	No detect.	Incorrect detection	Correct results(%)
original window	1280	1796	140	0	92.2
subsampling 1 pixel/4 on each row	320	1796	76	0	95.8
subsampling 1 pixel/8 on each row	160	1796	120	0	93.3
subsampling 1 pixel/16 on each row	80	1796	170	60	87.1
subsampling 1 pixel/8 on each row 1 pixel/2 on each column	80	1796	170	55	87.5

TABLE II
RESULTS: VARIATION OF USED MEASURED DISTANCES WITH A SUBSAMPLING 1 PIXEL/4 ON EACH ROW

	Number components	Number of faces to be verified	No detect.	Incorrect detection	Correct results(%)
Distance $d_2(\boldsymbol{x})$	320	1796	76	0	95.8
Distance $d_1(\boldsymbol{x})$	320	1796	64	0	96.4
Distance $d_0(\boldsymbol{x})$	320	1796	32	0	98.2

order to only obtain clusters according to identity, the k-means clustering algorithm was used in a supervised manner. It is respectively applied to training feature vectors belonging to each person. This was inspired by an algorithm proposed by

TABLE III
RESULTS: VARIATION OF RBF KERNEL ACTIVATION FUNCTIONS USING MEASURED DISTANCE $d_0(\boldsymbol{x})$ WITH A SUBSAMPLING 1 PIXEL/4 ON EACH ROW

	Number components	Number of faces to be verified	No detect.	Incorrect detection	Correct results(%)
Gaussian	320	1796	32	0	98.2
Heaviside	320	1796	118	0	93.4

Musavi *et al.* [23]. Initially, we have P training points (vectors) in an N-dimensional space and each training point is a cluster as follows:

1) take any point $\boldsymbol{c}_k$ and its associated width σ_k (initially $\sigma_k = 0$);
2) find the nearest point $\boldsymbol{c}_l$ of the same class by using the Euclidean distance;
3) compute the mean of these two points; we obtain a new point with its associated width $\sigma = (\|\boldsymbol{c}_k, \boldsymbol{c}_l\|)/2 + \sigma_k$;
4) compute the distance D from the new mean to the nearest point of all other classes;
5) if $D > \lambda\sigma$, then accept the merge of $\boldsymbol{c}_k$ and $\boldsymbol{c}_l$ and start again from step 2; if the condition is not satisfied, reject the merge and recover the two original points and their widths, then restart from step 1);
6) repeat steps 1)–5) until all clusters of each class be used.

λ is the "clustering parameter" with $1 \leq \lambda \leq 3$ [23]. We use the value $\lambda = 2$ in our case. Finally, we obtain the Gaussian centers $\boldsymbol{c}_i$ and the widths $\sigma_i (i = 1, \ldots, I \leq P)$ of the hidden nodes. We can see (Fig. 2) the result after using this clustering algorithm for

Fig. 6. Some images from the ORL database.

two classes in a two-dimensional (2-D) feature space. Note that we obtain two nonlinear decision regions. In fact, RBF neural networks can map spaces of any shape (nonlinear, convex, disjoint spaces). They use overlapping simple functions to cover complex decision regions.

Estimate the weights between the hidden and output layer : Given that the Gaussian function centers and widths are computed from P training vectors, the connection weights between the hidden and output layers can be calculated using the pseudo-inverse matrix method.

C. Model Description: Face Tracking and Identity Verification

Many face recognition algorithms require segmenting the face from the background, and subsequently extracting features such as eyes, nose, and mouth for further processing. We propose an integrated automatic face localization and identification model only using a classifier which responds to the question, "Does the input vector correspond or not the person to be verified?" The idea behind this is to simplify the model and reduce computation complexity in order to facilitate hardware implementations.

Fig. 3 represents the structure of our model. The size of faces in the scene varies from 40×32 pixels to 135×108 pixels with four scales. The ratio between any two scales is fixed to 1.5 [19], [24]. We first subsample the original scene and extract only the 40×32 windows in the 4 subsampled images. Each pre-processed 40×32 window (see Section III-A) is then fed to RBF network as an input vector. After the training procedure, the hidden nodes obtained are partially connected to the output layer. In fact, the hidden nodes associated with one person are only connected to the output node representing this class. This technique reduces data dependencies and is computationally more efficient [18]. The decision stage yields the presence, position, identity and scale of the faces using the maximal output values of the RBF neural network.

III. Experiments and Results

In this section, we present results obtained by using several configurations with our model. Experiments are based on eightvideo sequences of 256 images. In all sequences, the scene size is 288×352 pixels. In the first four sequences (see Fig. 5), there are either zero, one, two, or three different faces presented. We have decided to verify two persons (Soph and Seb) in these sequences. The 12 same training faces (see Fig. 4) are used for each person in order to compare the different configurations of the model.

A. Variation of the Input Vector Lengths

In this section, we try to reduce the input vectors length of the RBF network in order to simplify hardware implementations. In the preprocessing stage, we use first all pixels of each 40×32 window to compose the feature vectors. Each pixel represents one component of the vector. So, the input vectors of RBF neural network have 40×32 components. Second, we minimize the number of components in order to reduce the computing time. We realize a subsampling preprocessing: sample one pixel out of 4, 8, and 16 on each row of each window. We display some tested images (see Fig. 5). Results of face tracking and identity verification are presented in Table I. Efficiency of the model is defined as follows:

- correct detection and identification: the face is correctly located and identified;
- no detection: a presented face is not detected;
- incorrect detection: a face is not correctly located.

Performance decreases quickly when the input vectors have 80 components. In fact, incorrect detection regularly, appears when we use only one pixel out of 16 on each row of a window. The best results are obtained with one pixel out of four.

TABLE IV
Results of Face Recognition With ORL Database: These Results Have Been Obtained With the Distance $d_0(\boldsymbol{x})$ and the Gaussian Activation Function. Here, Number of Recognized Represents the Number of Faces to be Recognized and Number of Intruder Indicates the Number of Faces to be Rejected

Number of Examples/individual	$p = 1$	$p = 3$	$p = 5$
Number of Total Example	40	120	200
Number of Test	14400	11200	8000
Number of Recognized	360	280	200
Number of Intruder	14040	10920	7800
Err-No-Recognized	68	47	21
Err-Confusion	1026	682	226
Correct rate	92.4%	93.5%	96.9%

TABLE V
Comparison of Recognition Rates Against Other Methods Using ORL Database

Methods	$p = 1$	$p = 3$	$p = 5$
Sim & al.	75.1%	92.2%	97.1%
Howell & Buxton	84%	91%	95%
Slimane & al.	80%	89%	96%

B. Variation of Used Measured Distances

Previous results were obtained using the Euclidean distance $d_2(\boldsymbol{x})$ to compute the difference between an input vector $\boldsymbol{x}$ and the centers (kernels) $\boldsymbol{c}$ for each hidden node of the RBF neural network,

$$d_2(\boldsymbol{x}) = \sqrt{\sum_{1 \le n \le N} (x_n - c_n)^2} \tag{3}$$

where N is the number of components in an input vector.

Fig. 7. Multiscaled face tracking and identity verification.

The distance $d_1(\boldsymbol{x})$ is usually better when we use some noisy images [25]

$$d_1(\boldsymbol{x}) = \sum_{1 \le n \le N} |x_n - c_n|. \tag{4}$$

Another distance considers only the components whose difference between x_n and c_n is greater than a threshold δ

$$d_0(\boldsymbol{x}) = \sum_{1 \le n \le N} 1 \quad \forall\, |x_n - c_n| > \delta. \tag{5}$$

Here, the threshold δ has been regulated to 10 [25]. Table II shows that we have the best result with the $d_0(\boldsymbol{x})$ distance.

C. Variation of RBF Kernel Activation Functions

The Gaussian function is usually taken as the kernel activation function

$$f(\boldsymbol{x}) = \exp^{(-d(\boldsymbol{x})^2/\sigma^2)} \tag{6}$$

where $d(\boldsymbol{x})$ is the measured distance between the input vector $\boldsymbol{x}$ and the center $\boldsymbol{c}$.

Another approach is the use of a simplified activation, for example the replacement of the Gaussian function in the RBF network by a Heaviside function leading to a simplified hardware implementation. The width of this function is the width σ associated to the corresponding center

$$f(\boldsymbol{x}) = \begin{cases} 1 & d(\boldsymbol{x}) \le \sigma \\ 0 & d(\boldsymbol{x}) > \sigma \end{cases}. \tag{7}$$

If one of the centers (hidden nodes) of the same class produces an active response, the output unit linked to this class gives a positive response. Comparative results are shown in the Table III. The number of no-detections has increased with the Heaviside function. The rate of correct results decreases from 98.2% to 93.4%. In fact, the RBF neural network using the Heaviside function restrains the capacity of generalization by lack of interactions between centers of a same class: the model only detects faces that are sufficiently close to training examples.

Among all configurations of the model, the best performance has been obtained with 320 components of input vectors (subsampling 1 pixel/4 on each row of a window), using measured distance $d_0(\boldsymbol{x})$ and the Gaussian activation function : the success rate is 98.2%. Almost all the faces are well detected, localized, and identified in these four sequences of images (1024 scenes). Note that we have succeeded in regrouping 12 example training vectors for each person in six or eight centers (hidden nodes) using the clustering algorithm.

D. Face Recognition With ORL Database

We have adapted our model of face tracking and identity verification for an application of simple face recognition that we use for ORL database of faces [20]. This contains 400 grayscale images of 40 persons, each image has a resolution of 92×112. Some individuals have images taken at different times. Variations allowed in the image included lighting, facial expressions and facial details. All the images were taken against a plain background, with tilt and rotation of up to 20° and scale

TABLE VI
RESULTS: VARIATION OF FACE SCALES WITH A SUBSAMPLING 1 PIXEL/4 ON EACH ROW

	Nb.images	Number components	Number of faces to be verified	No detect.	Incorrect detection	Correct results(%)
$d_0(\boldsymbol{x})$+ Gaussian	4×256	320	1492	39	0	97.4
$d_1(\boldsymbol{x})$+ Heaviside	4×256	320	1492	121	0	91.9

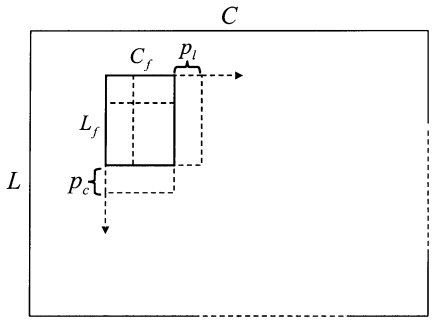


Fig. 8. Image scanning with a slippery window.

variation of up to 10%. Fig. 6 displays some images from ORL database. In order to use ORL data with the RBF network, we subsampled each image in order to obtain a resolution of 16×16 pixels [25] and the recognition was directly applied on these faces. The number of persons to verify is fixed at 40. One, three, or five arbitrarily chosen images of each individual makeup the training examples (number of patterns for learning by individual $p = 1, 3, 5$), and the rest of the faces are used to test the model. Several executions have been realized with each value of p distributing arbitrarily the training set and the test set. Then their means must be calculated and presented in Table IV. Efficiency is defined as follows:

correct—correct recognition of a face;
nonrecognition—a face has not been recognized;
confusion—a face is confused with an intruder.

For example, if we take $p = 1$, then each output unit (associated to a class) would have to recognize nine faces of its class and to reject the $9 \times 39 = 351$ faces corresponding to the other classes (intruders). With $p = 5$, we have a success rate of 97%, which is very close to or superior to the performances announced in the literature[9], [19], [25] (see Table V). Our system presents the advantages of moderate computational requirements and limited memory requirements. For example, generally when a method uses $p = 5$ training vectors by individual, our clustering algorithm allows us to use only 2 or 3 centers by individual.

E. Variation of Face Scales

We have applied our algorithm to the last four video sequences in which we can observe a great variation of the size of the faces and various lighting conditions. The analysis with four scales allows us to track and identify faces whose sizes vary from 40×32 to 135×108 pixels in the original scene (see Fig. 7). The results are shown in Table VI, and the success rates are, respectively, 97.4% using measured distance $d_0(\boldsymbol{x})$ and the Gaussian activation function, and 91.9% using distance $d_1(\boldsymbol{x})$ and the Heaviside function.

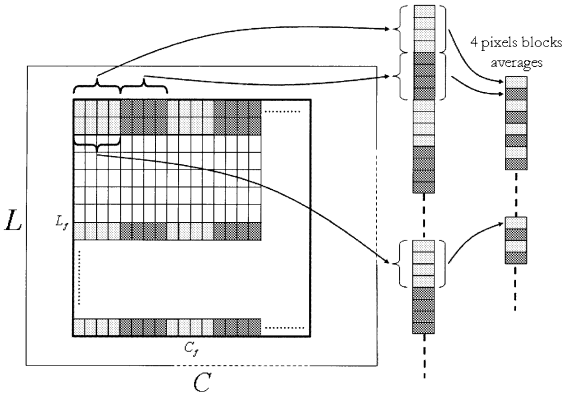


Fig. 9. Input vectors extraction.

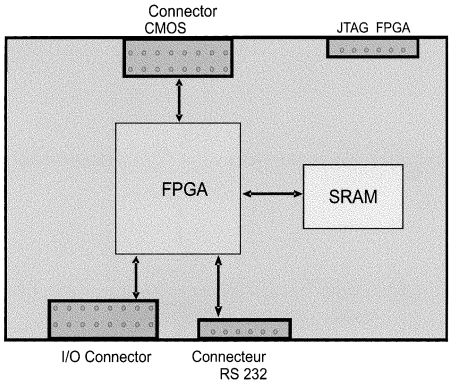


Fig. 10. Architecture of the MEMEC board.

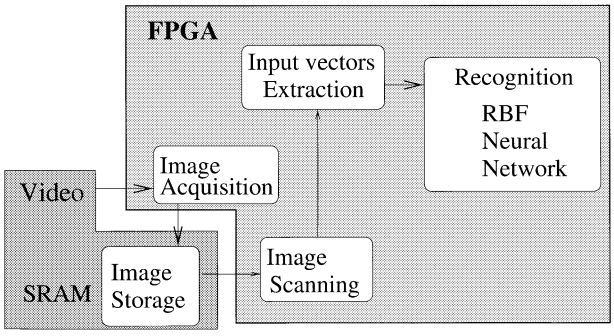


Fig. 11. Organization's tasks on the MEMEC board.

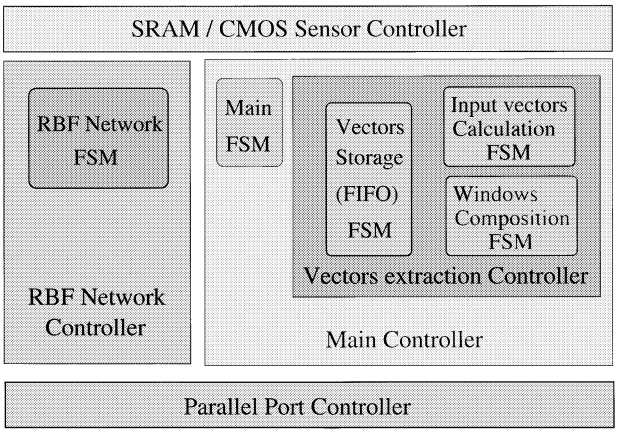


Fig. 12. Different controllers coded in VHDL [finite state machine (FSM)].

TABLE VII
RESULTS OF THE FIRST IMPLEMENTATION ON THE MEMEC BOARD

	extraction	15 centers	interfaces & controls	Total
Total Number of Slices	3072	3072	3072	3072
Number of Slices used	57	435	335	827
Slices used rate	2%	14.1%	10.9%	27%
Total Number of Blocks RAM	16	16	16	16
Number of Blocks RAM used	1	15	0	16
Blocks RAM used rate	6%	94%	0%	100%

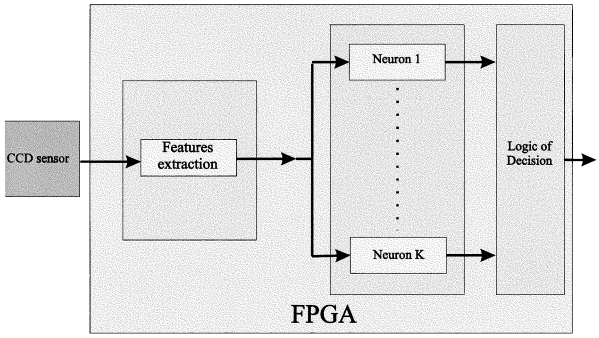


Fig. 13. Pipeline technique applied to each processing stage.

TABLE VIII
NECESSARY OPERATION NUMBERS FOR EACH STAGE OF FIRST IMPLEMENTATION: V REPRESENTS THE NUMBER OF WINDOWS TO BE ANALYZED, A_1 AND D CORRESPOND TO THE NUMBERS OF ADDITIONS AND DIVISIONS FOR INPUT VECTORS EXTRACTION. A_2, S, AND O ARE THE NUMBERS OF ADDITIONS, SUBTRACTIONS AND COMPARISONS FOR DISTANCE AND DECISION CALCULATION. THESE RESULTS HAVE BE OBTAINED USING (8)–(16)

	V	A_1	D	A_2	S	O
Image : 288×352						
Window : 40×32						
Scan steps : $p_l = 2$, $p_c = 4$	10143	1112832	370944	48534255	48838545	142002
Nb. hidden nodes =15						
Scale =1						

Our aim is to realize face tracking and identity verification in real time on embedded systems. This is why we consider the "mapping algorithm architecture" in order to optimize the implementation. For example, we have reduced the complexity and the volume of calculation using the partial connection between the hidden layer and the output layer of RBF neural network. With the clustering algorithm, we have succeeded in reducing the number of hidden nodes. We have also evaluated our model with the distance $d_1(\boldsymbol{x})$ and the Heaviside activation function which are easier to realize hardware implementations and give an acceptable performance (close to 92% success rate). In Section IV, we analyze each stage of the complexity of our model in order to prepare hardware implementations.

IV. FROM ALGORITHM TO ARCHITECTURE: COMPLEXITY OF THE MODEL

A. Number of Windows to be Analyzed

We use a slippery window of $L_f \times C_f$ dimension applied to a $L \times C$ image (see Fig. 8). The dimension of this window as well as the displacement scan step along the row and column determine the number of windows to be analyzed V that we calculate by

$$V = \left\lfloor \frac{L - L_f}{p_c} + 1 \right\rfloor \left\lfloor \frac{C - C_f}{p_l} + 1 \right\rfloor \tag{8}$$

where L is the number of rows in the image, C is the number of columns, L_f is the number of rows in a window, C_f the number of its columns, p_c the scan step along the columns of the image, and p_l the scan step among the rows.

For E scales (scale $e = 1, \ldots, E$)

$$V_E = \sum_{e=1}^{E} \left(\left\lfloor \frac{L_e - L_{f_e}}{p_{c_e}} + 1 \right\rfloor \left\lfloor \frac{C_e - C_{f_e}}{p_{l_e}} + 1 \right\rfloor \right) \tag{9}$$

we use the same dimension of slippery window for each scale as well as the same displacement scan step. We take L_f, C_f, p_l, and p_c constant to each scale. The previous equation becomes

$$V_E = \sum_{e=1}^{E} \left(\left\lfloor \frac{L_e - L_f}{p_c} + 1 \right\rfloor \left\lfloor \frac{C_e - C_f}{p_l} + 1 \right\rfloor \right). \tag{10}$$

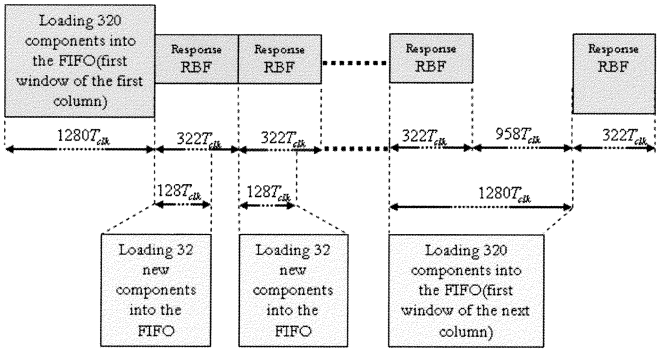


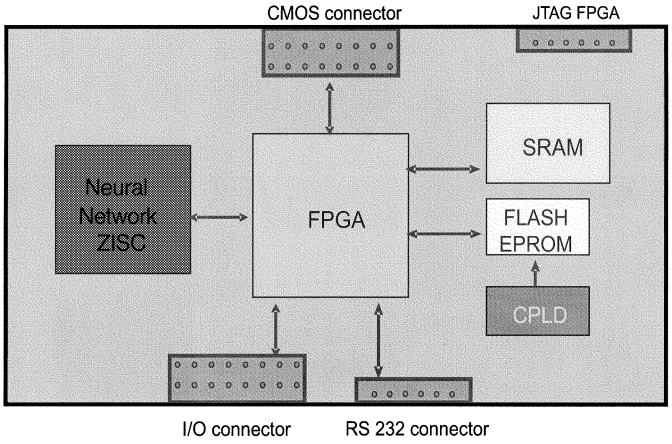
Fig. 14. Computing time diagram for the first implementation.

TABLE IX
RESULTS: REALIZATION ON THE MEMEC BOARD

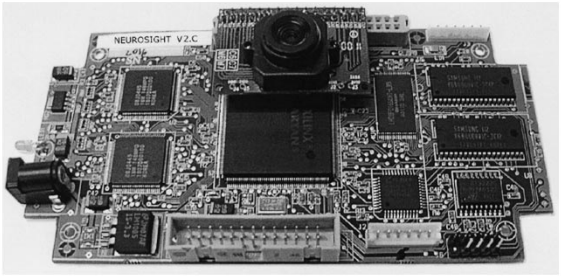
Nb.components	Number of faces to be verified	No detect.	Incorrect detection	Correct results (%)
320	1796	123	20	92

B. Input Vectors Extraction

In order to simplify the hardware implementations of our model we reduce its complexity. Originally, input vectors of RBF network are obtained by subsampling one pixel out of four of each row in slippery windows of 40×32 pixels. Thus, we have 320 components by vector. Here, we approximate this stage of subsampling using the average of four successive pixels. On each row of each window, we compute the sum of eight consecutive 4–pixel blocks (see Figs. 8 and 9). We vertically scan the image to be analyzed so as to use the redundancy of components of a vector with its following. We thus reduce the calculation linked to input vectors extraction.



(a)



(b)

Fig. 15. (a) Neurosight block diagram and (b) the board picture.

The number of necessary additions for the calculation of the vector linked to the first window of each column is

$$A_1' = \left(\frac{C_f}{8} - 1\right) \times 8 \times L_f. \tag{11}$$

In order to form each new vector from following windows of each column, it suffices to add to the preceding $8 \times p_c$ new components. In order to analyze an image with only one scale, the number of necessary additions for the extraction of all vectors is

$$A_1 = \left(\frac{C_f}{8} - 1\right) \times 8 \times L \times \left\lfloor \frac{C - C_f}{p_l} + 1 \right\rfloor. \tag{12}$$

In the same manner, we obtain the number of necessary divisions for the extraction of all vectors

$$D = 8 \times L \times \left\lfloor \frac{C - C_f}{p_l} + 1 \right\rfloor. \tag{13}$$

C. Distance and Decision Calculation

Equations (4) and (5) show that the complexity of the distance $d_1(\boldsymbol{x})$ is almost identical to that of the distance $d_0(\boldsymbol{x})$. By using the distance $d_0(\boldsymbol{x})$, we obtain performances slightly superior to those of the distance $d_1(\boldsymbol{x})$ (see Tables II and VI), however controlling the threshold δ of this distance is difficult. Our group has therefore focused on the distance $d_1(\boldsymbol{x})$. This distance calculation demands N subtractions and $N-1$ additions by center [see (4)]. We have

$$A_2 = (N-1) \times I \times V \tag{14}$$

$$S = N \times I \times V \tag{15}$$

where A_2 and S are respectively the number of necessary additions and subtractions for an image linked to distance calculation. N corresponds to the number of components in each input vector and I is the total number of hidden nodes. The output

TABLE X
NECESSARY OPERATION NUMBERS FOR EACH STAGE OF SECOND IMPLEMENTATION: V REPRESENTS THE NUMBER OF WINDOWS TO BE ANALYZED, A_1 AND D CORRESPOND TO THE NUMBERS OF ADDITIONS AND DIVISIONS FOR INPUT VECTORS EXTRACTION. A_2, S, AND O ARE THE NUMBERS OF ADDITIONS, SUBTRACTIONS AND COMPARISONS FOR DISTANCE AND DECISION CALCULATION. THESE RESULTS HAVE BE OBTAINED USING (8)–(16)

	V	A_1	D	A_2	S	O
Image : 72×352						
Window : 8×32						
Scan steps : $p_l = 2$, $p_c = 1$	10465	278208	92736	9889425	10046400	146510
Nb. hidden nodes =15						
Scale =1						

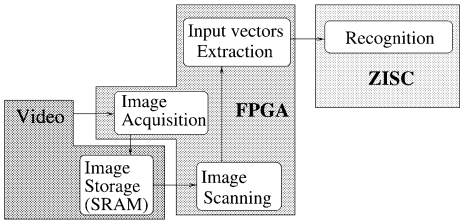


Fig. 16. Distribution's tasks on the neurosight board.

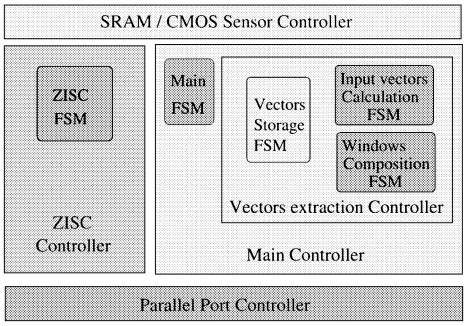


Fig. 17. Different controllers coded in VHDL for the second hardware implementation.

TABLE XI
RESULTS OF THE SECOND IMPLEMENTATION ON THE NEUROSIGHT BOARD

	extraction	interfaces & Controls	Total
Total Number of Slices	768	768	768
Number of Slices used	57	235	292
Slices used rate	7.4%	30.6%	38%
Total Number of Blocks RAM	8	8	8
Number of Blocks RAM used	1	0	1
Blocks RAM used rate	12.5%	0%	12.5%

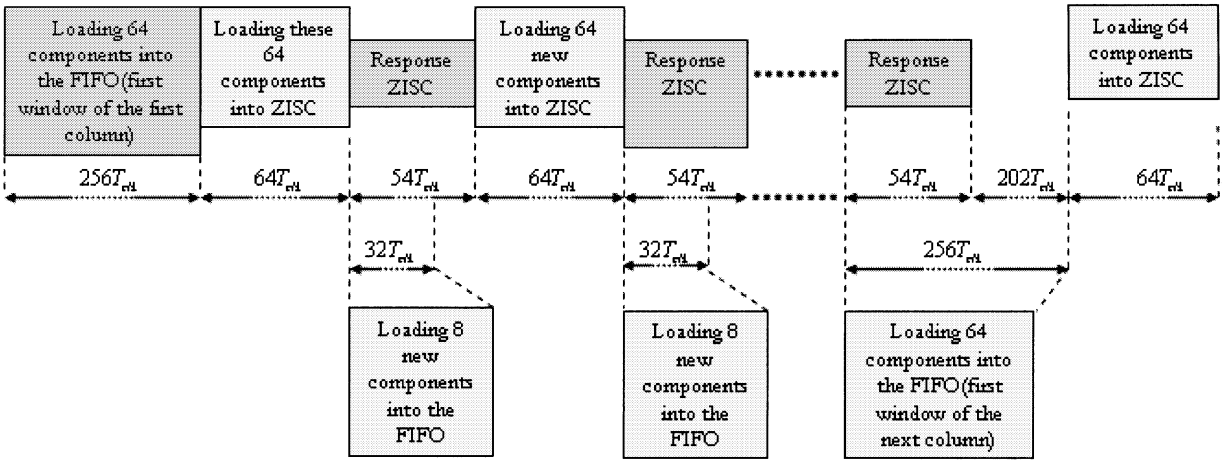


Fig. 18. Computing time diagram for the second implementation.

layer of the neural network uses *binary logic function "or"* with Heaviside activation function in hidden layer: each center compares the calculated distance with the distance threshold determined during the training step. We have used such comparisons as centers. So

$$O = V \times (I - 1) \tag{16}$$

represents the maximum number of comparisons to compare all responses of the RBF network contained in an image.

V. Hardware Implementations

Hardware implementations of the RBF approach have be realized for different applications, on either FPGA [26], or neurochip [27]. Commercial RBF products include the IBM ZISC chip and the Nestor Ni 1000 chip [28]. Here, our aim is to elaborate in real time an efficient model of unconstrained face tracking and identity verification in arbitrary scenes. Thus, hardware implementations have been realized on three embedded systems based on FPGA, ZISC chip, and DSP. We use industrial electronic systems: a MEMEC board [29], a general vision neurosight board [30], and a board based on DSP TMS320c6x developed in our laboratory. We discuss first for each case the architecture of the system. Then results are presented in terms of hardware resources used and processing speed, and we compare the three implementations and propose the possible improvements. Note that these implementations are used for face tracking and identity verification with only one scale $(\text{scale} = 1)$. A more complete future realization will take into account different scales.

A. First Realization: Implementation Based on FPGA

1) Hardware Resources: This first implementation is realized on a *MEMEC* industrial board [29] comprising a FPGA *Xilinx SpartanII-300* [31], which contains 3072 "slices" (slice = basic cell logic of FPGA) and 16 memory blocks of 512 bytes each (Fig. 10). We have implanted on the FPGA our model of face tracking and identity verification with the VHDL description using Xilinx ISE tool [31].

This implementation creates an RBF neural network with 15 hidden nodes. Each hidden node stores a center vector of 320 components. The used measured distance is the distance $d_1(\boldsymbol{x})$. The activation function of each center is a Heaviside function whose associated width delimits the influence area of the center. Figs. 11 and 12 show the organization's tasks and the coding of these tasks using VHDL description for this first hardware implementation. The original video image is stored in an image memory bank with each pixel coded on a byte, the input vector extraction consists of calculating averages of four successive pixels on rows of the image. Each vector is fed to the 15 hidden nodes of the RBF network which gives their respective responses in parallel. In Table VII, we present information on FPGA resources used. The input vectors extraction needs 57 "slices" in order to define the image memory access and the interaction logic with centers. A memory block (FIFO) is necessary to store input vectors to be tested. Each trained center needs one memory block and 29 slices for calculation (distance, activation function, decision). This implementation uses 827 "slices" (27% of total resources). Note that the number of centers is limited by the number of independent internal memory blocks.

TABLE XII
Results: Realization on the Neurosight Board

Nb.components	Number of faces to be verified	No detect.	Incorrect detection	Correct results (%)
64	1796	210	53	85.3

2) Processing Speed: The complete implementation is realized in parallel using the pipeline technique for each stage of the processing (see Fig. 13). The images size is 288×352 and contains $161 \times 63 = 10\,143$ windows of 40×32 pixels each. The displacement scan step along the row is $p_l = 2$ and along the column is $p_c = 4$ [see (10)]. We realized, respectively, 49.65M additions, 48.8M subtractions, 370 944 divisions, and 142 002 comparisons (see Table VIII).

We determined the number of clock periods, T_{clk}, required for each stage of the process in order to analyze one image (see Fig. 14). The first windows of the first column needs $1280T_{\text{clk}}$ (320 components $\times 4T_{\text{clk}}$) to calculate 320 components of its input vector and then $322T_{\text{clk}}$ to obtain the decision of all centers. This first window is tested using $1602T_{\text{clk}}$. We require only

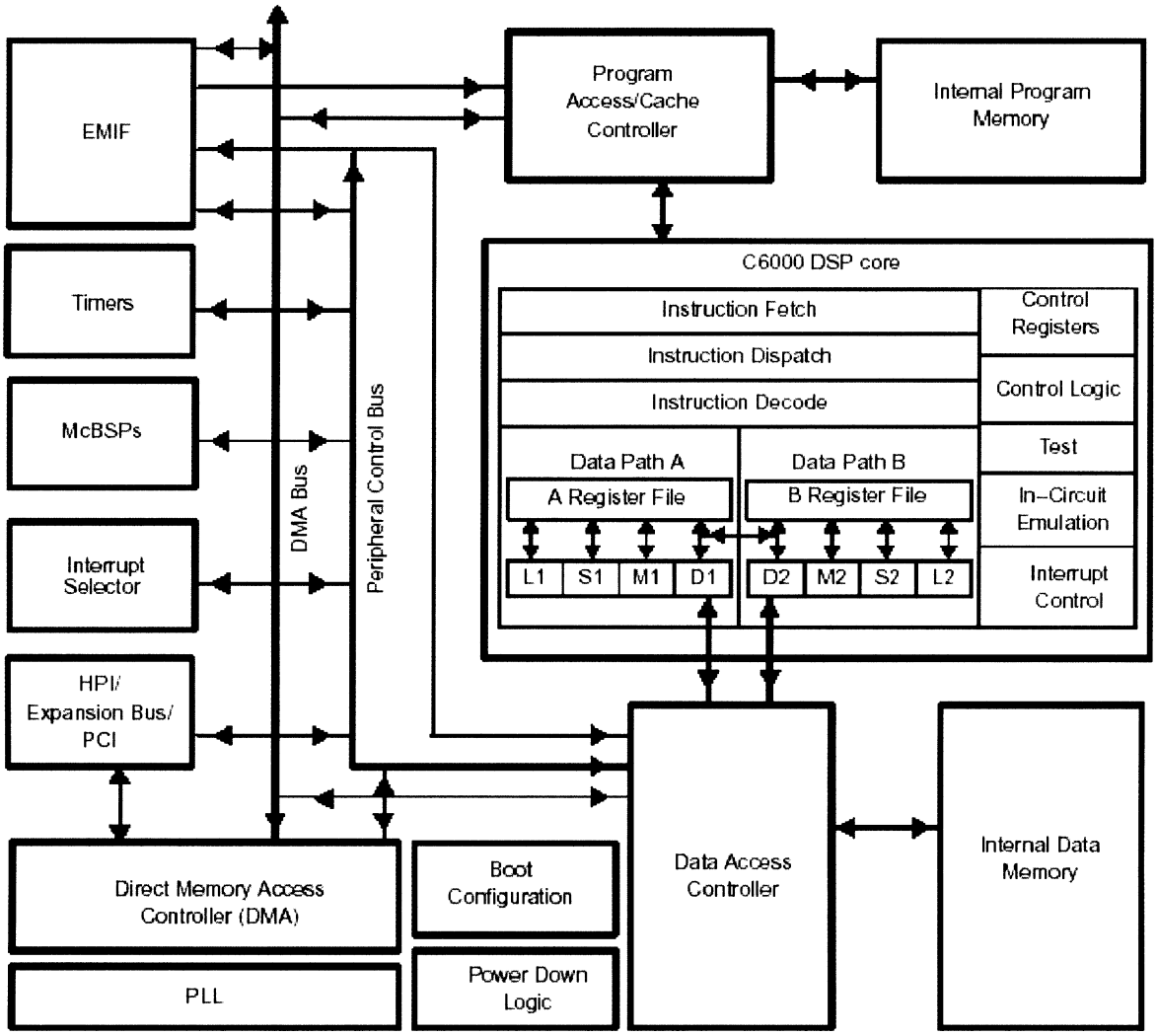


Fig. 19. Internal architecture of DSP TMS320 C6201.

32 new components (eight components by row of the window $\times$ four new rows) to form the following vector. This input vector extraction is defined while centers determine the responses of RBF network linked to the previous window. So, 62 following windows of this column use $322 \times 62 = 19\,964 T_{\mathrm{clk}}$ to be tested. This first column of the image is analyzed in $21\,566 T_{\mathrm{clk}}$. The first window of each next column requires only $1280 T_{\mathrm{clk}}$ in order to be processed. Sixty-two other windows of these columns need $322 T_{\mathrm{clk}} \times 62$ in order to be tested. This implementation uses $3.42 M T_{\mathrm{clk}}$ for one image of 288×352 with one scale. The oscillator frequency on the board is 50 Mhz ($T_{\mathrm{clk}} = 20$ ns). The processing speed of this first implementation is 14 images per second with a success rate of 92% for face tracking and identity verification (see Table IX).

B. Second Realization: Implementation Based on ZISC Chip

1) Hardware Resources: We also made hardware implementation of our model using a commercial board linked to pattern recognition applications. This *General Vision Neurosight* board [30] contains a CMOS sensor (288×352 pixels), an FPGA Xilinx SpartanII-50, two memory banks of 512KB each, as well as two specific ZISC chips [28] (see Fig. 15).

One ZISC [30] chip contains 78 RBF-link nodes with a maximal length of input vectors $N = 64$. The used measured distance and the activation function of each node are, respectively, the distance $d_1(\boldsymbol{x})$ and the Heaviside function. We adapt the complexity of the model to this embedded system. At first, we reduce the size of the original image by keeping only one line out four. This new image obtained (size 72×352) is then analyzed with a slippery window of 8×32. On each row of each window, we compute averages of eight consecutive four pixels blocks. Each window yields an input vector of 64 components to be analyzed by the ZISC chip. We have $V = 10\,465$ windows (65 windows per row and 161 windows per column) to be tested with $L = 72$, $C = 352$, $L_f = 8$, $C_f = 32$, and $p_c = 1$, $p_l = 2$ (10). Necessary operation numbers to be realized, are respectively, 10.16 M additions, 10.05 M subtractions, 92 736 divisions, and 146 510 comparisons (see Table X).

We implement the input vectors extraction and all interfaces (memory access, ZISC access) on the FPGA Xilinx SpartanII-50. Figs. 16 and 17 show the distribution's tasks on the Neurosight board and the different levels of control coded in VHDL.

Table XI presents information on hardware resources used for this second implementation. The input vectors extraction imple-

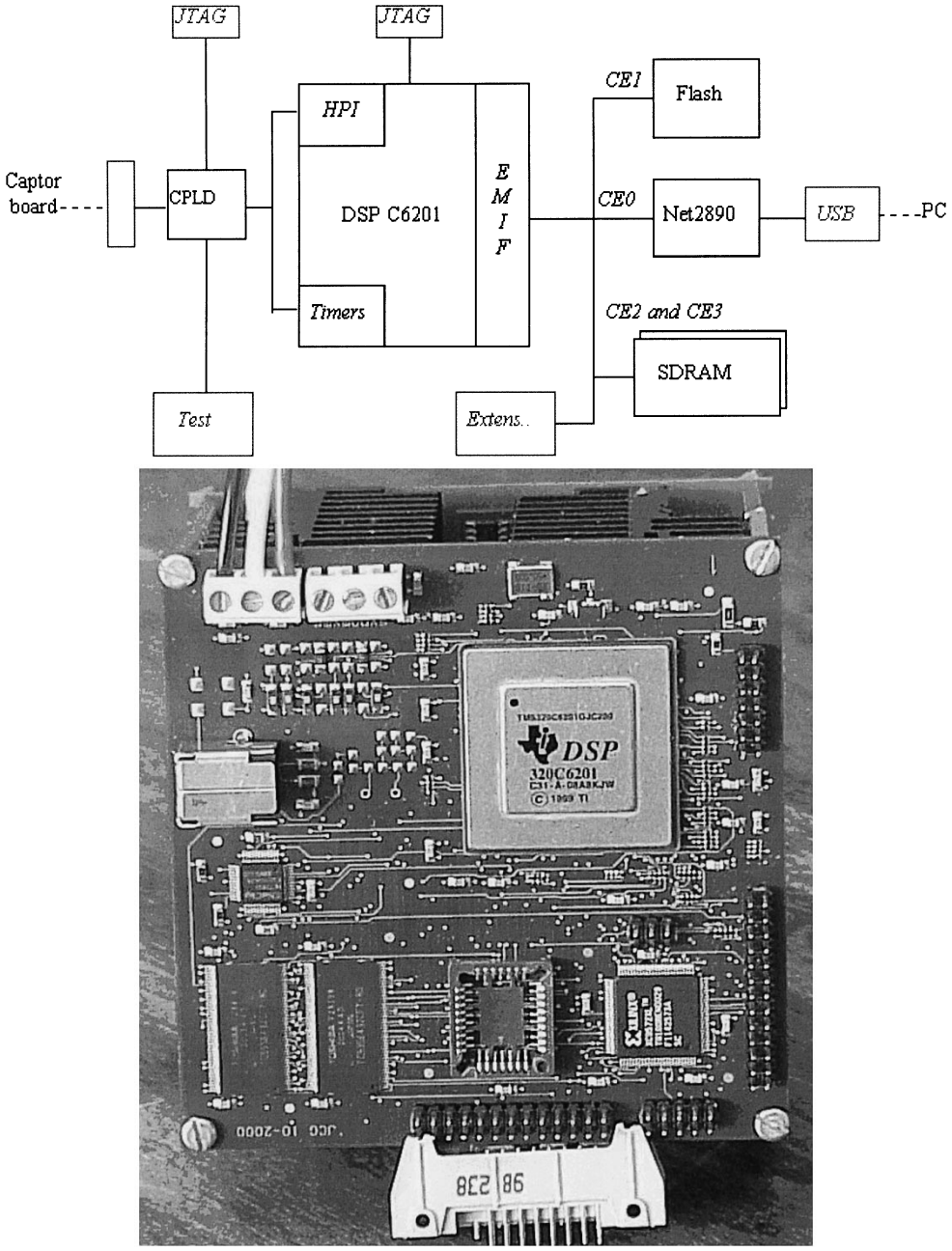


Fig. 20. Block diagram and the board picture of the third embedded system.

mentation requires the same resources as those used with the MEMEC board. Here, we use only one ZISC chip (78 nodes maximum).

2) Processing Speed: We have determined the number of clock periods, $T_{\rm clk}$, required for each stage of the process in order to analyze one image (see Fig. 18) like for the first realization. This second implementation uses $1.27MT_{\rm clk}$ for one 288×352 image with one scale. The oscillator frequency on the board is 33 Mhz ($T_{\rm clk} = 30$ ns). The processing speed of this second implementation is 25 images/s with a success rate of 85.3% for face tracking and identity verification (see Table XII).

C. Third Realization: Implementation Based on DSP

1) Hardware Resources: DSPs are specific processors destined for signal and image processing [32]. The C6x family is the last generation Texas Instruments DSP. They are available in fixed point (C62x and C64x) and floating point (C67x) versions, and CPU frequencies range from 150 MHz to 600 MHz. It is possible to do eight operations per clock-cycle with these DSPs. The program fetches, dispatches instruction, and decodes instruction units can deliver up to eight 32-bit instructions to functional units every CPU clock cycle. The instruction processing occurs in each of the two data-paths (A and B), each of which contains four functional units, one multiplier and three arithmetic logic unit (ALU), and 16 32-bit register files (see Fig. 19).

Our laboratory has developed a system based on a DSP TMS320 C6201B (see Fig. 20). A CCD sensor sends 16-bit data to the DSP via a complex programmable logic devices (CPLD). The DSP performs different processing and sends the

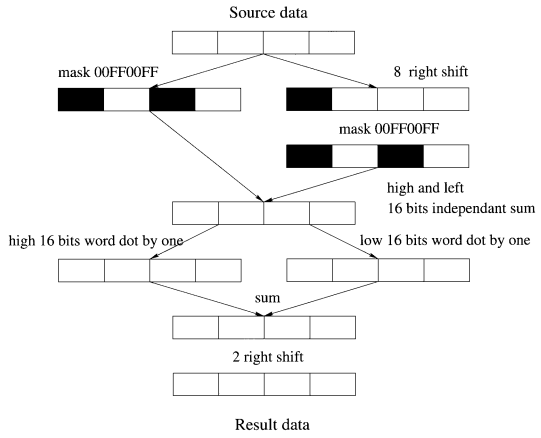


Fig. 21. Sequential diagram of input vectors extraction.

resulting images to a PC via a universal serial bus (USB). We have chosen a ROM boot process from a flash memory for the DSP (embedded board) [33]. Two SDRAM memories of $2M \times 32$ bits are available to store images between the different processings [34].

The third hardware implementation of our model for face tracking and identity verification is realized on this embedded system. The goal is to optimize in Assembler each stage of processing using, in parallel, eight functional units of TMS320 DSP C6201. We give implementation details for the input vectors extraction stage as an example. The corresponding calculation of this stage is represented by the following C function: For $(i = 0; i < 1280; i+ = 4)\ dst[i/4] = (src[i] + src[i + 1] + src[i + 2] + src[i + 3])/4$, where dst and src are vectors which contain, respectively, 1280 and 320 byte elements. So, in order to minimize the memory access, we access data 4 $\times$ 4 by loading a 32-bit word in each cycle. The sequential diagram below (Fig. 21), shows all processing steps applied to data during this stage of input vectors extraction.

Details of optimized implementation in parallel on 8 functional units are shown in the Table XIII. The goal of the optimized implantation is to use in parallel the maximum number of DSP functional units. Note that the detail is given only for data-path A [33]. The same progress concerns the data-path B, and allows us to divide halve the processing time. The A1 ... A9 and B0 ... B2 correspond to registers of the DSP TMS320C6201. Italic lines correspond to loop control and line skip instructions. The A2 register is used for the loop index and registers B0, B1, and B2 correspond to the flags that control the line skip at the appropriate instant. The A1 register points to source data and A10 points to result data. Efficiency indicates number of units used divided by eight units of DSP.

2) Processing Speed: Tables XIV and XV show, respectively, experimental implementation results obtained using the DSP C6201 and simulation results obtained using the DSP C64x with the development tool, Code Composer Studio (Texas Instruments). Theses results are in terms of processing speed. We have applied a vertical window (40×32 pixels) displacement to use the calculation redundancy for the input vectors extraction stage. We give some parameters for the following results:

TABLE XIII
PARALLEL IMPLEMENTATION OF INPUT VECTORS EXTRACTION FOR ith LOOP STEP

Step	Instructions	Units	Comment	Efficiency
	Load data pointed by A1 → A6(i+1)	D		
1	*// if (A2<0) A2-1 → A2*	S1	loop counter	0.5
	// if (A2>0) Jump loop	S2	loop jump	
	A6(i) and 00ff00ff → A7(i)	L		
2	// A6(i)>>8 → A8(i)	S		0.75
	// A7(i-1)+A8(i-1) → A9(i-1)	D		
3	A8(i) and 00ff00ff → A8(i)	L		
	// A9(i-1)/4 → A5(i-1)	S		0.5
	if (B0!=0) A5(i-1) → save data pointed by A10	S		
4	// A7(i)+A8(i) → A7(i)	D		0.63
	//if (B0!=0) B2-1 → B2	L2	line skip	
5	high (A7(n)× 1) → A8(i+1)	M		
	//if (B0!=0) 1 → B0	S2		0.38
6	low (A7(n)× 1) → A7(i+1)	M		
	//if (B0!=0) A1+72 → A1	S	line skip	0.63
	//if (B0!=0) B2+4 → B2	L2	line skip	

TABLE XIV
IMPLEMENTATION RESULTS USING TMS320 C6201

Language	C	Assembler
Input vectors extraction	4.14 ms	1.8 ms
Distance calculation	211 ms	144 ms
Gaussian function + Decision	67 ms	
Processing speed	3.5 images/second	4.8 images/second

TABLE XV
SIMULATION RESULTS USING TMS320 C64X

Language	C	Assembler
Input vectors extraction	1.2 ms	0.14 ms
Distance calculation	58.8 ms	13.3 ms
Gaussian function + Decision	22.2 ms	
Processing speed	12.1 images/second	28.6 images/second

- only one scale $(\text{scale} = 1)$ of initial image (288×352 pixels) is implemented which requires the most calculations in comparison with the other scales;
- $p_l = 2$, $p_c = 4$, and number of hidden nodes $= 15$;
- used measured distance and the activation function of each node are, respectively, the distance $d_0(\boldsymbol{x})$ and a Gaussian function; the number of components of input vector $N = 320$;
- number of windows to be analyzed V and the numbers of additions and divisions for input vectors extraction A_1, D are identical that the first implementation;
- correct rate of 98.2% is obtained for face tracking and identity verification.

D. Discussion on Three Hardware Implementations

Table XVI compares hardware complexity of three embedded systems. Note that some components (*CPLD, Flash Memory, Connector JTAG...*) can elaborate embedded systems,

TABLE XVI
HARDWARE COMPLEXITY COMPARISON OF THREE EMBEDDED SYSTEMS

	FPGA	ZISC	DSP
Components	CMOS sensor SRAM FPGA SpartanII-300	CMOS sensor SRAM FPGA SpartanII-50 2 ZISC chips	CCD sensor SDRAM DSP TMS320C6201B
Image Memory Size	2Mbytes	1Mbytes	16Mbytes
Interface Peripherals	Parallel port	Parallel port	USB port
Development Tool	Xilinx ISE	Xilinx ISE	Code Composer Studio

TABLE XVII
HARDWARE IMPLEMENTATIONS COMPARISON

	FPGA	ZISC	DSP
Components used	CMOS sensor SRAM SpartanII-300 (FPGA)	CMOS sensor SRAM SpartanII-50 (FPGA) 1 ZISC chip	CCD sensor SDRAM DSP C6201B
Image Memory used	99Kbytes	99Kbytes	99Kbytes
Number of Slices used	292 of 768	827 of 3072	
Number of Blocks RAM used	16 of 16	1 of 8	
Processing Speed	14 images/s	25 images/s	4.8 images/s
Power consumption	967 mw	883 mw	684 mw
Number of operations realized per second	1.39 Gops	0.53 Gops	1.63 Gops
Performances	85.3%	92%	98.2%

however, they are not used in our hardware implementations of face tracking and identity verification. The comparisons among results of three hardware implementations are shown in Table XVII.

Hardware resources used of the FPGA system are three times those used of the ZISC system, in terms of slices (827 versus 292). On the other hand, only an internal memory block is used on the ZISC system while all blocks (16 memory blocks) are used on the FPGA system. Occupation rate in terms of slices and memory blocks are, respectively, 38% and 12.5% for the ZISC system against 27% and 100% for the FPGA system. The great advantage of the ZISC is that it has 156 totally parallel hidden nodes while we have only 15 nodes on the FPGA system.

The ZISC processing speed is superior (25 images per second versus 14 images per second for the FPGA) while the performance associated with this system is the weakest (success rate of 85% against 92% for the FPGA system). The performance difference is essentially linked to the fact that the input vectors coded on the ZISC have a maximum length of 64 components.

A greater input vectors length of the ZISC specific chip would allow to performance increase of face tracking and identity verification. On the other hand, the pippeline placement of registers that stock the vector to be tested in each node would allow a reduction of the loading time from $64T_{\text{clk}}$ to $8T_{\text{clk}}$ (see Fig. 18). This is linked to the redundancy of two successive vectors component. For the FPGA system, three improvements are foreseeable, we can double the number of nodes if input vectors length is limited to a maximum of 256 components, because memory blocks of the FPGA possess a double access port (division of a block in two blocks of 256 bytes each). On the other hand, we can double the processing speed on this system. Indeed, we can divide the inputs vectors in two half-vectors (the input vector here has 512 components in maximum) stocked each in a half-block memory. To test an input vector, we input the distance calculation of this vector and a node and this can be realized in parallel on each half-vector. We can also extract two half-vectors to be tested in parallel because the image memory equally possesses a double access port.

For the third implementation, thanks to the performances by the DSP TMS320 C6201, we optimized our model in its ideal configuration and we have obtained a success rate of 98.2%. On the other hand the processing speed is inferior to the two preceding implementations (4.8 images/s versus 14 images/s for the FPGA and 25 images/s for the ZISC). The processing speed would increase to 28.6 images/s using the DSP TMS320 C64.

VI. CONCLUSION

We created a model that allows us to detect the presence of faces, to follow them, and to verify their identities in video sequences using a RBF neural network. The model's robustness has been tested using eight video sequences and the ORL face database. The best performance has been obtained with the following configuration: one subsampling of a pixel/4 for each row, the measured distance $d_0(\boldsymbol{x})$ and the Gaussian activation function. In fact, the subsampling preprocessing and the application of the $d_0(\boldsymbol{x})$ distance render the model less sensitive to face details and to the small differences between training examples and test windows, thus, we have the better generalization.

In relation to the lighting problem, we have tested the method proposed by Rowley [4] that consists of approximating the variation of lighting inside a oval mask (face) by a linear function in order to balance brightness distribution of a window. This improves the success rate in video sequences where there is a great variation in lighting [35]. The important calculation required for this method would considerably reduce hardware implementation performances. We envisaged adding in the examples training set synthesis faces which are formed by simulating the different conditions of lighting. Another solution based on the strategy of evaluative neural networks is to train the model with the last detected faces.

We have demonstrated the feasibility of face tracking and identity verification in real time using existing commercial boards. We have implanted our model on three embedded systems. The success rate of face tracking and identity verification is, respectively, 92% (FPGA), 85% (ZISC), and 98.2% (DSP). Processing speeds obtained for images of size 288×352 are, respectively, 14 images/s, 25 images/s, and 4.8 images/s. The discussion of these implementations allows us to propose possible improvements for the architecture of each embedded system.

Our model integrating 15 hidden nodes allows us to distinguish two faces with a good performance ($>90\%$ of success rate). Extending this model to recognition of more faces (>10) necessitates a calculation power superior to 10 Giga flops and,

thus, new architectures must be developed. They can be developed using more effective components, for example, *FPGA VirtexII* (integrating many multipliers and memories) or *DSP TMS320C64*, thus allowing a very rapid processing speed and better performance of face tracking and identity verification.

Our short-term perspective is to complete implementations using all four scales of images. A long term perspective is to extract input vectors of RBF neural network using an artificial retina, unloading a consequent calculation part of FPGA. Finally, we also would link to test our model on other applications of pattern recognition such as vehicle detection and tracking in real time.

REFERENCES

[1] M. S. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in images: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, pp. 42–53, Jan. 2001.

[2] R. Férand *et al.*, "A fast and accurate face detector based on neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, pp. 42–53, Jan. 2001.

[3] Intrator, D. Reisfeld, and Y. Yeshurnn, "Face recognition using a hybrid supervised/unsupervised neural network," *Pattern Recogn. Lett.*, no. 17, pp. 67–76, Jan. 2001.

[4] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, pp. 23–38, Jan. 1998.

[5] D. Valentin, H. Abdi, and B. Edelman, "What represents a face: A computational approach for the integration of physiological and psychological data," *Percept.*, vol. 26, pp. 1271–1288, 1997.

[6] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cogn. Neurosci.*, vol. 3, pp. 71–86, 1991.

[7] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, pp. 696–710, July 1997.

[8] H. Abdi, D. Valentin, and A. O'Toole, "A generalized auto-associator model for face semantic process," in *Optimization and Neural Network*, D. Levine, Ed. Hillsdale, NJ: Erlbaum, 1997.

[9] M. Slimane *et al.*, "Apprentissage nonsupervisé d'images par hybridation génétique d'une chaîne markov cachée," *Traitement du Signal*, vol. 16, no. 6, pp. 461–475, 1999.

[10] F. Yang and M. Paindavoine, "Face detection and localization in a scene: Parallel implementation on multi-DSP," *Traitement du Signal*, vol. 17, no. 2, pp. 179–188, 2000.

[11] K. Kobayashi, "Mobile terminals and devices technology for the 21st century," *Spec. Rev. Paper—New Start 21st Century: NEC Res. Develop.*, vol. 42, no. 1, 2001.

[12] Y. H. Yeh and C. Y. Lee, "Cost effective vlsi architectures and buffer size optimization for full search block matching algorithms," *IEEE Trans. VLSI Syst.*, vol. 7, pp. 345–358, June 1999.

[13] H. Liu *et al.*, "An automatic human face recognition system," *Opt. Lasers Eng.*, vol. 30, pp. 305–314, 1998.

[14] J. Buhmann, M. Lades, and F. Eeckmann, "Illumination-invariant face recognition with a contrast sensitive silicon retina," in *Advances in Neural Information Processing Systems (NIPS)*. New York: Morgan Kaufmann, 1994, vol. 6, pp. 769–776.

[15] Y. Matsumoto and A. Zelinsky, "An algorithm for real-time stereo vision implementation of head pose and gaze direction mesasurement," in *Proc. 4th IEEE Int. Conf. Automatic Face and Gesture Recognition*, Grenoble, France, Mar. 26–30, 2000, pp. 499–504.

[16] M. Rosenblum, Y. Yacoob, and L. S. Davis, "Human expression recognition from motion using a radial basis function network architecture," *IEEE Trans. Neural Networks*, vol. 7, pp. 1121–1138, Sept. 1996.

[17] S. Ranganath and K. Arun, "Face recognition using transform features and neural networks," *Pattern Recogn.*, vol. 30, no. 10, pp. 1615–1622, 1997.

[18] L. H. Koh, S. Ranganath, and Y. V. Venkatesh, "An integrated automatic face detection and recognition system," *Pattern Recogn.*, vol. 35, pp. 1259–1273, 2002.

[19] A. J. Howell and H. Buxton, "Learning identity with radial basis function networks," *Neurocomput.*, vol. 20, pp. 15–34, 1998.

[20] [Online]. Available: http://www.research.att.com/facedatabase.html

[21] J. Moody and C. Darken, "Learning with localized receptive fields," in *Proc. Connectionist Models Summer School*, San Mateo, CA, 1988.

[22] I. Park and I. W. Sandberg, "Universal approximation using radial basis function networks," *Neural Computat.*, vol. 3, pp. 246–257, 1991.

[23] M. T. Musavial *et al.*, "On the training of radial basis function classifiers," *Neural Networks*, vol. 5, pp. 595–603, 1992.

[24] E. Viennet, "Architecture Connexionniste Multimodulaire: Applicationà l'analyze de scène," Doctoral dissertation, Univ. de Paris Sud, 1993.

[25] T. Sim *et al.*, "Memory-based face recognition for visitor identification," in *Proc. 4th IEEE Int. Conf. Automatic Face and Gesture Recognition*, Grenoble, France, Mar. 2000, pp. 26–30.

[26] A. Pérez-Uribe and E. Sanchez, "FPGA implementation of an adaptable-size neural network," in *Proc. VI Int. Conf. Artificial Neural Networks ICANN'96*, Bochum, Germany, July 1996.

[27] M. Skrbek, "Fast neural network implementation," *Neural Network World*, vol. 5, pp. 375–391, 1999.

[28] T. Lindblad *et al.*, "Implementating of the new zero instruction set computer (ZISC036) from IBM for a Higgs Search," *Nucl. Instrum. Methods*, vol. A357, 1995.

[29] [Online]. Available: www.memec.com

[30] [Online]. Available: www.general-vision.com

[31] The Programmable Logic Data Book. Xilinx 1998. [Online]. Available: www.xilinx.com

[32] "TMS320C6000 Technical Brief SPRU197D (02/99),".

[33] "TMS320C6000 EMIF to External Flash Memory SPRA568,".

[34] "TMS320C6000 EMIF to External SDRAM Interface SPRA433B (12/01),".

[35] N. Malasné, F. Yang, M. Paindavoine, and J. Mitéran, "Suivi dynamique et vérification de visages en temps réel: Algorithme et architecture," in *Proc. 13ème RFIA Conf. Francophone*, Angers, France, Jan. 2002.

Fan Yang received the B.S. degree in electrical engineering from the University of Lanzhou, China, in 1982 and the M.S. (D.E.A.) (computer science) and Ph.D. degrees (image processing) from the University of Burgundy, France, in 1994 and 1998, respectively.

She was a Scientific Assistant at the Department of Electronics, University of Lanzhon, from 1982 to 1992. She is currently a full "Maitre de Conferences" and member of LE2I CNRS-UMR, Laboratory of Electronic, Computing, and Imaging Sciences at the University of Burgundy, France. Her research interests are in the areas of patterns recognition, neural network, motion estimation based on spatio-temporal Cabor filters, parallelism and real-time implementation, and, more specifically, automatic face image processing algorithms and architectures.

Michel Paindavoine received the Ph.D. degree in electronics and signal processing from Montpellier University, Montpellier, France, in 1982.

He joined Burgundy University, France, in 1985 as "Maitre de Conferences" and is currently a Full Professor and Director of LE2I UMR-CNRS, Laboratory of Electronic, Computing, and Imaging Sciences, Burgundy University. His main research topics are image acquisition and real time image processing.

Dr. Paindavoine is a member of ISIS (a research group in signal and image processing) of the French National Scientific Research Committee.