REAL TIME IMPLEMENTATION OF A FACE TRACKING

N.Malasne, F.Yang, M.Paindavoine
Laboratory LE2I, University of Burgundy,
BP 47870 21078 Dijon Cedex, France
Tel: (33)3 80 39 63 29 ; fax: (33) 3 80 39 59 10
e-mail: nmalasne@u-bourgogne.fr

ABSTRACT

This paper describes a system capable of realizing a face detection and tracking in video sequences. In developing this system, we have used a RBF neural network to locate and categorize faces of different dimensions. The face tracker can be applied to a video communication system which allows the users to move freely in front of the camera while communicating. The system works at several stages. At first, we extract useful parameters by a low-pass filtering to compress data and we compose our codebook vectors. Then, the RBF neural network realizes a face detection and tracking on a specific board.

1 Introduction

A system capable of doing face localization and recognition in real time has many applications in intelligent man-machine interfaces and in other domains such as very low bandwidth video conferencing, virtual actor and video e-mail. We describe a system capable of detecting and to track faces in video sequences using a RBF neural network.

The Radial Basis Function (RBF) allows to make learning in neural networks. This function makes it possible to design a network with a good generalization ability and a minimum number of nodes to avoid unnecessary computational time. The RBF method is a technique for interpolation in a high dimensional space. RBF classifiers belong to the category of kernels classifiers. They use an overlapping formed by simple kernel functions to create complex decision regions. RBF networks are a recent addition to the face tracking and analysis model because their main advantages are computational simplicity and robust generalization. Mark Rosenblum and al.[1] have developed a system of human expressions recognition from motion based on a RBF network architecture. Howell and Buxton have performed a learning identification with RBF method[2].

Our aim is to elaborate a quite efficient algorithm using a RBF network which can track and recognize faces of different dimensions in real time in natural video sequences in any background. In the second section, we present the RBF network model. Learning process and node reduction are described. We show the system of face tracking developed in the third section. We exhibit the chosen method : extraction of useful parameters, RBF network architecture used. We display the results and performances we have obtained with a video sequence. Finally, we discuss about our hardware implementation.

2 Architecture of the network

The architecture of a RBF network[3] [4] is composed of 3 layers (see Figure 1). Each hidden node computes a kernel function on input data and the output layer achieves a weighted summation of the kernel functions. Each node is characterized by 2 important associated parameters : its center and the width of the radial function. A hidden node computes the highest output value when the input data is close to its center and this output decreases as the distance from the center increases. Several distances can be used to estimate the distance from a center but we usually use the Euclidian distance. A Gaussian function is taken as the kernel function[5]. The whole network configuration is achieved by calculating centers and widths associated with the hidden nodes and the weights of the connections from the hidden layer to the output layer.

Input data are fully connected to the hidden layer and this one is also fully connected to the output layer. Each input vector has M components. There are I RBF nodes in the hidden layer. In addition, each RBF node is characterized by 2 parameters, the center $\mathbf{c}_i$ and the width w_i of its associated Gaussian function. The output layer contains J nodes O_j representing J classes. The connections from hidden nodes to output nodes are weighted by multiplication values. Finally, each output node yields a weighted sum of its inputs.

2.1 Learning process

At first, the learning problem is to determine the appropriate Gaussian centers $\mathbf{c}_i$ and widths w_i. Then, the RBF learning process consists in defining some functions f_j to satisfy the condition :

$$f_j(\mathbf{x}_p) = \mathbf{y}_{pj} \quad p = 1, ..., P \quad j = 1, ..., J \qquad (1)$$

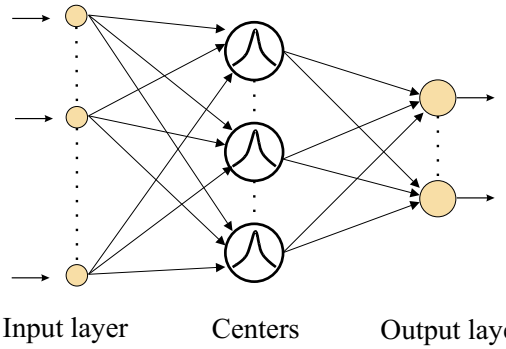

Figure 1: *Architecture of a typical RBF network.*

in which P and J are respectively the total number of training patterns and the number of classes. $(\mathbf{x}_p, \mathbf{y}_{pj})$ are the p-th training vectors, $\mathbf{y}_{pj}$ being the response that the output j have to yield when the $\mathbf{x}_p$ training pattern is presented to input. $f(\mathbf{x})$ can be written as :

$$f(\mathbf{x}) = \Sigma_{i=1}^{I} a_{ij} \Phi(\|\mathbf{x} - \mathbf{c}_i\|) \tag{2}$$

with $i = 1, ..., I, j = 1, ..., J$ and Φ is a Gaussian function

$$\Phi(\|\mathbf{x} - \mathbf{c}_i\|) = \frac{1}{\sqrt{2\pi {w_i}^2}} exp\{\frac{-\|\mathbf{x} - \mathbf{c}_i\|}{2{w_i}^2}\} \tag{3}$$

A pseudo inverse matrix technique yields the weight a_{ij} associating the hidden layer with the output layer. A way to design the network could be to associate a Gaussian function with each training point in the M-dimensional space. But in most of the applications, the number of training vectors is large and this technique becomes inefficient.

2.2 Node reduction

Initially, we have P training points in a M-dimensional space. If it is possible, our aim is to reduce the number of useful points. The algorithm which we use is inspired on a clustering algorithm proposed by Musavi[6].

1. Take any point C_k and its associated radius w_k (initially, $w_k = 0$).
2. Find the nearest point C_l of the same class by using the Euclidian distance.
3. Compute the mean of these 2 points. We obtain a new point with its associated width $\frac{\|C_k, C_l\|}{2} + w_k$.
4. Compute the distance D from the new mean to the nearest point of all other classes.
5. If $D > \lambda R$, then accept the merge of C_k and C_l and start again from step 2. If the condition is not satisfied, reject the merge and recover the 2 original points and their widths then restart from step 1.
6. Repeat step 1 to 5 until all points of each class be tested.

Finally, we obtain the Gaussian centers $\mathbf{c}_i$ and their widths $w_i (i = 1, ..., I \leq P)$ of the hidden nodes.

λ is the "clustering parameter" (λ is a positive number). When λ increases, node reduction is limited, but the accuracy increases. We use the value $\lambda = 2$ in our case. We can see (Figure 2) the result after using this clustering algorithm for 2 classes in a 2D feature space.

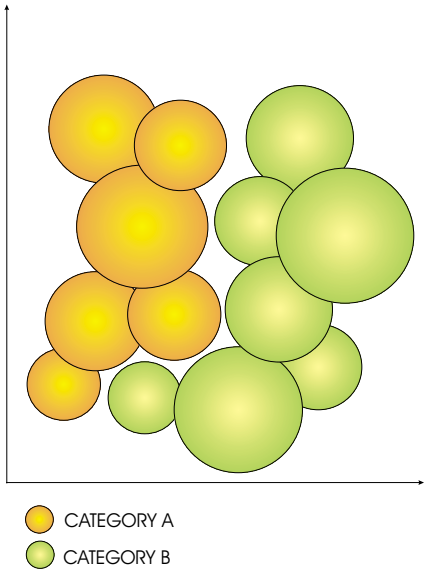

Figure 2: *Regions mapping in a 2D space.*

Notice that we obtain 2 non linear decision regions. In fact, RBF can map spaces of any shapes (non-linear, convex, disjoint spaces). In a M-dimensional space, the decisions regions are a set of M-dimensional hyperspheres.

3 Experiments and results

3.1 Image sequence

Our sequence contains 74 images of 2 persons moving in a room. This sequence was recorded in brightness format. The original resolution is 240×320. The frame rate is 10 Hz. Any special lighting was used. Notice that the size of the faces varies from 40×30 to 90×68. At first, we have to learn the training faces. In our example, we want to recognize 2 persons. We use 3 training faces for each of them (see Figure 3). Each training face is a window of size 40×30.

Our aim is to locate and recognize faces in a video sequence. So, it is very important to use a method as low time consuming as possible. It is the reason why we have to minimize the number of input data. Thus, a downsampling is realized. So, a 1-dimensional smoothing filtering is necessarily performed on the original images used. Only one brightness value out of six is taken

Figure 3: Learning faces

on each line of the images. So, the training vectors have only 200 (5×40) components. The clustering algorithm finds 2 centers and their widths for each class again. The location and recognition is realized by scanning (scan step=1) the image 240×320 with the 40×30 mask then downsampling it. Each mask yields 200 components to network input. The test results are presented in Tab.1.

Table 1: *Test results*

number of person in the sequence	141	
right detection and identification	132	93.6 %
no detection	4	2.8 %
wrong detection	5	3.6 %
wrong identification	0	0 %

The feature extraction yields input vectors with less components. So, we have reduced the computational time a lot. Therefore, the hardware implementation are feasible to realize a real-time recognition of faces. Here are some result images :

Figure 4: Results

4 Hardware implementation

4.1 Computing complexity

Our aim is to realize a specific board integrating our algorithm. We have chosen to use the *Neurosight* board from *General Vision*[8]. This board contains a CCD sensor (352×288 pixels), a FPGA Xilinx Spartan2-100, 2 memory banks (512Ko each one) and the 2 specific chips $ZISC$ (Zero Instruction Set Computer chip). One $ZISC$ chip contains 78 RBF-like neurons with a $M = 64$ components length vector, the distance computing

$$d_1(\mathbf{x}) = \sum_{1 \leq m \leq M} |x_m - c_m| \tag{4}$$

and the Heaviside decision function

$$f(\mathbf{x}) = \begin{cases} 1 & d(\mathbf{x}) \leq \sigma \\ 0 & d(\mathbf{x}) > \sigma \end{cases} \tag{5}$$

We have to adapt the complexity of the system to this board. We have done some choices. At first, we reduce the size of the original image by keeping only 1 line out of 4. This new image obtained(its size is 352×72) is now analysed with a window 32×8. On each line of each window, we compute the averages of the 8 consecutive 4 pixels blocks. Each window yields a 64 components length vector to be analysed with ZISC. The number of windows to analyse in an image is

$$N_v = \lfloor \frac{L - L_v}{P_c} + 1 \rfloor \lfloor \frac{C - C_v}{P_l} + 1 \rfloor \tag{6}$$

where L is the number of rows in the image, C is the number of columns, L_v is the number of rows in a window, C_v 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. The equations

$$A_1 = (\frac{C_v}{B} - 1) \times B \times L_v \tag{7}$$

$$D_1 = B \times L_v \tag{8}$$

represent the number A_1 of additions and is the number D_1 of divisions we need to extract the parameters of the first window of each column. B corresponds to the number of blocks per window line. The scan step on one column is $P_c = 1$, so we only need to compute 8 new parameters to have the next characteristic vector. The equations

$$A_2 = (\frac{C_v}{B} - 1) \times B \tag{9}$$

$$D_2 = B \tag{10}$$

represent respectively the number of additions and divisions we need to compute these new components and the equations

$$A_3 = (\frac{C_v}{B} - 1)B(L_v + \lfloor \frac{L - L_v}{P_c} \rfloor) \lfloor \frac{C - C_v}{P_l} + 1 \rfloor \tag{11}$$

$$D = B(L_v + \lfloor \frac{L - L_v}{P_c} \rfloor) \lfloor \frac{C - C_v}{P_l} + 1 \rfloor \tag{12}$$

correspond to the number of additions and divisions we need to extract all the vectors in a whole image.

We take $L = 72$, $C = 352$, $L_v = 8$, $C_v = 32$ and $P_c = 1, P_l = 2$, we obtain $N_v = 10465$ windows to test (65 windows per columns and 161 windows per line).

We need 278208 additions and 92736 divisions to extract from a image all the vectors to test with neurons.

Now, we are going to estimate the complexity of the ZISC neurons. The numbers A_3 of additions and S of subtractions we need to compute the distance $d_1(\mathbf{x})$ for all the windows are obtained with the equations

$$A_4 = (M-1) \times I \times N_v \quad (13)$$

$$S = M \times I \times N_v \quad (14)$$

where M represents the number of components in an input vector and I the number of neurons used. If all the neurons are used on both the $ZISC$, we take $I = 156$ and we have $A = A_3 + A_4 = 103.2$M additions et $S = 104.5$M soustractions. Then, each neuron compare the computing distance with the distance threshold determined during the learning step. We have such comparisons as neurons used. So, the equation

$$C = N_v \times (I-1) \quad (15)$$

represents the maximal number of comparison we need to compare all the vectors contained in a image. We obtain $C = 1622075$ comparisons with $I = 156$.

4.2 computing time

We have estimated the computing time we need to make our system working. We have determinated the number of clock periods (we note T_{Clk}) required for each step of the process to analyze one image.

Each image has 65 windows per columns to be tested. The first window requires $256T_{Clk}$ to compute and put the 64 first components in a FIFO then $64T_{Clk}$ to put this first vector in $ZISC$. All neurons of this one yield a response after $54T_{Clk}$ (neurons in parallel mode). So, we need $374T_{Clk}$ to test this first window. We only need to compute 8 new components to have a new vector to test. This one is obtained during $ZISC$ take a decision for the previous one. Next windows of this column need $64T_{Clk} + 54T_{Clk}) \times 64 = 7552T_{Clk}$ each of them to be tested. The first column is analyzed in $7926T_{Clk}$. Next columns requires $320T_{Clk} + (64T_{Clk} + 54T_{Clk}) \times 64 = 7872T_{Clk}$ to be checked. We only need $320T_{Clk}(374-54)$ to compute the first vector of next column. We have 161 columns to check. We need about $1.27MT_{Clk}$ to test all the windows in an image. The frequency of the oscillator frequency on the board is 33Mhz ($T_{Clk} = 30ns$). Test one image is realized in about 38ms. It's smaller than 40ms which is the threshold to respect the real time computing (25 frames per second).

5 Conclusion and perspectives

Our aim was to realize a first simple hardware implementation of a learned faces tracker. We used the specific chip $ZISC$ to recognize parameters extracted from a image. Each $ZISC$ chip contains 78 parallel neurons. This architecture requires a high volume computing. The system is capable of recognizing several persons in a video sequence in real time. At the moment, the results are not accurate enough because each face is represented with too low parameters. Our next hardware will use some bigger vectors (128 then 256 components) to increase the accuracy. We would like to implement a RBF-like neural network on a FPGA chip too.

References

[1] M. Rosenblum, Y. Yacoob and S.V. Larry, "Human expression recognition from motion using a radial basis function network architecture," *IEEE Transaction on neural networks,* Vol.7, No.5, pp. 1121-1138, 1996.

[2] A.J. Howell, Y. Buxton and S.V. Larry, "Learning identity with radial basis function networks," *Neurocomputing,* Elsevier, Vol.20, pp. 15-34, 1998.

[3] M.J.D. Powell, "Radial Basis Functions for multivariate interpolation: A review," *Algorithms for approximation,* Clarendon Press, pp. 143-167, Oxford, 1987.

[4] I. Park and I.W. Sandberg, "Universal approximation using radial basis function networks," *Neural Computations,* Vol.3, pp. 246-257, 1991.

[5] S. Lee and R.M. Kil, "A Gaussian potential function network with hierarchically self-organizing learning," *Neural Networks,* Vol.4, pp. 207-224, 1991.

[6] M.T. Musavi, W. Ahmed and al, "On the training of radial basis function classifiers," *Neural Networks,* Vol.5, pp. 595-603, 1992.

[7] H. A.Rowley, S. Baluja, T. Kanade, "Neural Network-Based Face Detection," *PAMI,* 1998.

[8] Web site : www.general-vision.com