# RBF NEURAL NETWORKS APPLIED TO FACE TRACKING AND RECOGNITION

Laboratory LE2I, University of Burgundy, 21000 DIJON, France

## Abstract

**This paper describes a trainable system capable of detecting and tracking faces 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 £rst, we extract useful parameters by a low-pass £ltering to compress data and we compose our codebook vectors. Then, the RBF neural network realizes detection and tracking of faces.**

## 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 trainable system capable of detecting and tracking 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 classi£ers belong to the category of kernels classi£ers. 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 identi£cation with RBF method[2].

Our aim is to elaborate a quite ef£cient algorithm using a RBF network which can track and recognize faces of different dimensions in natural video sequences in any background. In the future, we want to implant this algorithm on an FPGA (Field Programmable Gate Array) device associated with an arti£cial retina. 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.

## 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 con£guration 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 $c_i$ 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.

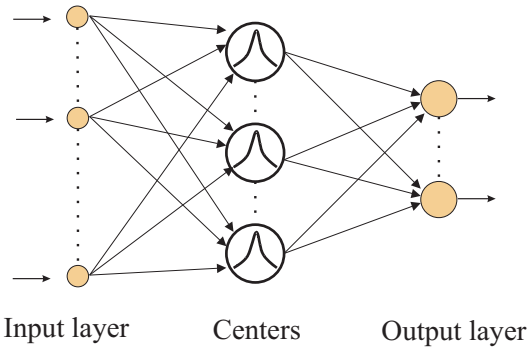



Figure 1: *Architecture of a typical RBF network.*

### 2.1 Learning process

At £rst, the learning problem is to determine the appropriate Gaussian centers $c_i$ and widths $w_i$. Then, the RBF learning process consists in de£ning some functions $f_j$ to satisfy the condition :

$$f_j(x_p) = y_{pj} \qquad p = 1, ..., P \qquad j = 1, ..., J \qquad (1)$$

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

$$f(x) = \Sigma_{i=1}^{I} a_{ij} \Phi(\|x - c_i\|) \qquad (2)$$

with $i = 1, ..., I, j = 1, ..., J$ and $\Phi$ is a Gaussian function :

$$\Phi(\|x - c_i\|) = \frac{1}{\sqrt{2\pi {w_i}^2}} exp\{\frac{-\|x - c_i\|}{2{w_i}^2}\} \qquad (3)$$

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 inef£cient. Moreover, even if the number of training vectors is short, it's better to design a network as simple as possible.

### 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].
Initially, each training point is a cluster.

1. Take any cluster $C_k$.
2. Find the nearest point $C_l$ of the same class by using the Euclidian distance.
3. Compute the mean of this 2 clusters. The width of this new cluster is $\frac{\|C_k, C_l\|}{2} + w_k$. The new cluster is the new mean associated with its width.
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 satis-£ed, reject the merge and recover the 2 original clusters then restart from step 1.
6. Repeat step 1 to 5 until all clusters of each class be used.

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

$\lambda$ is the "clustering parameter" ($\lambda$ is a positive number). When $\lambda$ 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.

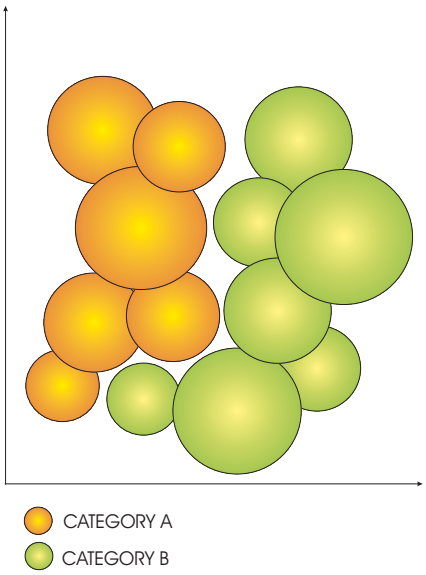



Figure 2: *Regions mapping in a 2D space.*

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 N-dimensional space, the decisions regions are a set of N-dimensional hyperspheres.

## 3 Experiments : Faces location and recognition

### 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 $240 \times 320$. The frame rate is 10

Hz. Any special lighting was used. Notice that the size of the faces varies from $40 \times 30$ to $90 \times 68$.

At £rst, 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 $40 \times 30$.

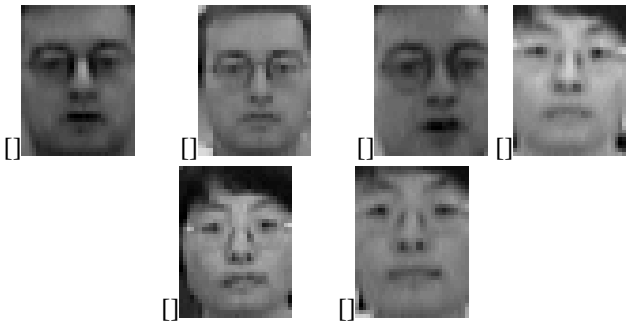


Figure 3: *Learning faces.*

## 3.2 First simulation : original image

The components of the training vectors are at £rst simply the brightness values (1 byte) of the 1200 pixels in each training face. If the lighting conditions are extreme, we can to perform a preprocessing. This preprocessing attempts to equalize the intensity values in across the window. We £t a function which varies linearly across the window to the intensity values in an oval region inside the window[7]. Pixels outside the oval may represent the background, so those intensity values are ignored in computing the lighting variation across the face. The linear function will approximate the brightness of each part of the window, and can be subtracted from the window to compensate for the variation of lighting conditions. We have applied this technique to our sequence. You can see results with some faces from the Face Yale Database (see Figure 4).

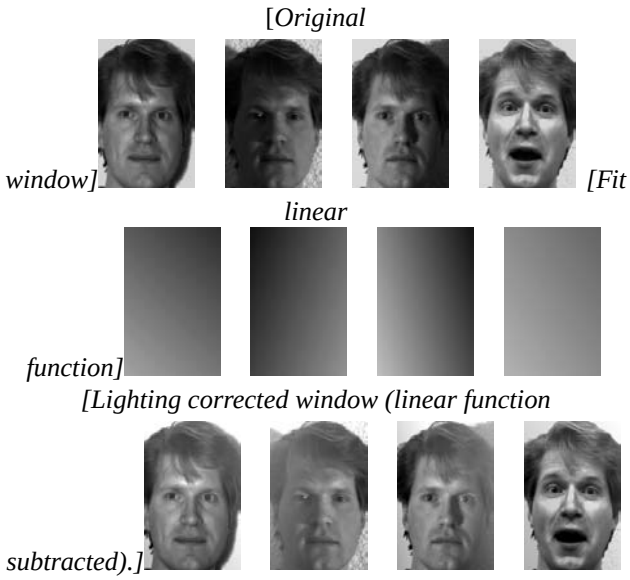



*Figure 4: Lighting correction.*

Then, the clustering algorithm £nds 2 centers and their widths for each class (person). The RBF network is achieved with these parameters. However, the hidden nodes are partially connected to the output layer (see Figure 5). In fact, the hidden nodes associated with one person are only connected to the output node representing this one.

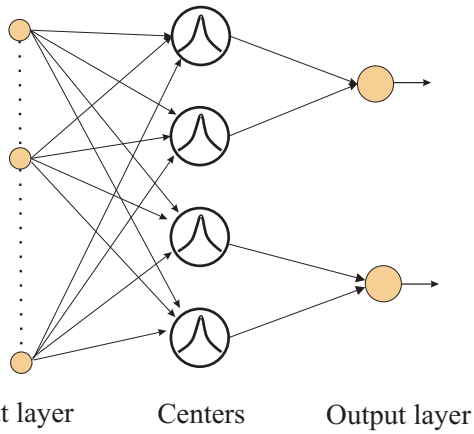



Figure 5: *Architecture of the RBF network used.*

The location and recognition is performed by extracting all the $40 \times 30$ samples (scan step=1) in the scene of size $240 \times 320$. Each $40 \times 30$ mask yields 1200 components to network input.

The performance of the system is displayed in Tab.1. Four different results for each face are distinguished :

- Right detection and identi£cation : the person's face is correctly located and identi£ed.
- No detection : a face is not detected.
- Wrong detection : a face is not correctly located.
- Wrong identi£cation : a face is correctly located but not correctly identi£ed.

Table 1: *Test results*

| | | |
|---|---|---|
| number of person in the sequence | 141 | 100% |
| right detection and identi£cation | 131 | 92.9 % |
| no detection | 4 | 2.8 % |
| wrong detection | 6 | 4.1 % |
| wrong identi£cation | 0 | 0 % |

## 3.3 Second simulation : features extraction

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 £ltering is

necessarily performed on the original images used. Only one brightness value out of six is taken on each line of the images. So, the training vectors have now only 200 ($5 \times 40$) components. The clustering algorithm £nds 2 centers and their widths for each class again. The location and recognition is realized by scanning (scan step=1) the image $240 \times 320$ with the $40 \times 30$ mask then downsampling it. Each mask yields 200 components to network input. The test results are presented in Tab.2.

Table 2: *Test results*

| | | |
|---|---|---|
| number of person in the sequence | 141 | 100% |
| right detection and identi£cation | 132 | 93.6 % |
| no detection | 4 | 2.8 % |
| wrong detection | 5 | 3.6 % |
| wrong identi£cation | 0 | 0 % |

We can see that the results are very close for both simulations. However, the feature extraction yields input vectors with less components. So, we have reduced the computational time a lot. Therefore, the hardware implementations are feasible to realize a real-time recognition of faces. Here are some result images :

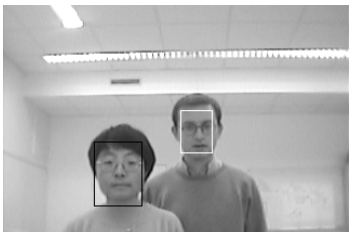
[*result00.]* 

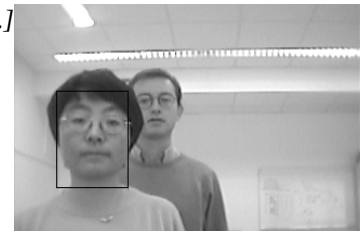
 *[result10.]*

# 4 Conclusion and perspectives

Our next aim is the implementation of a face tracker and recognizer on a chip as FPGA, DSP or ASIC. Thus, we need an ef£cient method as simple as possible. That is the reason why we use a RBF network whose architecture is quite simple. We have proposed an algorithm to simplify the network architecture (hidden nodes reduction). In addition, we have to use a fast feature extraction to reduce the size of the input vectors. Finally, we will ratify our method using other image sequences tests then we will realize the hardware implementation.

[*result25.]* 

 *[result32.]*

 *[result50.]*

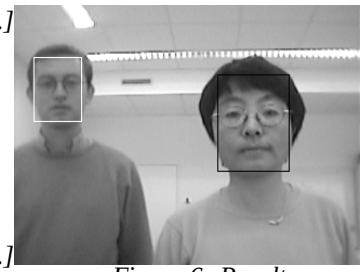
 *[result63.]*

*Figure 6: Results.*

# 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 classi£ers," *Neural Networks*, Vol.5, pp. 595-603, 1992.

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