1

Facial expression recognition using shape and texture information

I. Kotsia[1] and I. Pitas[1]

Aristotle University of Thessaloniki
pitas@aiia.csd.auth.gr
Department of Informatics
Box 451 54124
Thessaloniki, Greece

Summary. A novel method based on shape and texture information is proposed in this paper for facial expression recognition from video sequences. The Discriminant Non-negative Matrix Factorization (DNMF) algorithm is applied at the image corresponding to the greatest intensity of the facial expression (last frame of the video sequence), extracting that way the texture information. A Support Vector Machines (SVMs) system is used for the classification of the shape information derived from tracking the Candide grid over the video sequence. The shape information consists of the differences of the node coordinates between the first (neutral) and last (fully expressed facial expression) video frame. Subsequently, fusion of texture and shape information obtained is performed using Radial Basis Function (RBF) Neural Networks (NNs). The accuracy achieved is equal to 98,2% when recognizing the six basic facial expressions.

1.1 Introduction

During the past two decades, many studies regarding facial expression recognition, which plays a vital role in human centered interfaces, have been conducted. Psychologists have defined the following basic facial expressions: anger, disgust, fear, happiness, sadness and surprise [**?**]. A set of muscle movements, known as Action Units, was created. These movements form the so called *Facial Action Coding System* (*FACS*) [**?**]. A survey on automatic facial expression recognition can be found in [**?**].

In the current paper, a novel method for video based facial expression recognition by fusing texture and shape information is proposed. The texture information is obtained by applying the DNMF algorithm [**?**] on the last frame of the video sequence, i.e. the one that corresponds to the greatest intensity of the facial expression depicted. The shape information is calculated as the difference of Candide facial model grid node coordinates between the first and the last frame of a video sequence [**?**]. The decision made regarding the class the sample belongs to, is obtained using a SVM system. Both the DNMF and SVM algorithms have as an output the distances of the sample under examination from each of the six classes (facial expressions). Fusion of the distances obtained from DNMF and SVMs applications is attempted using a RBF NN system. The experiments performed using the Cohn-Kanade database indicate a recognition accuracy of 98,2% when recognizing the six basic facial expressions. The novelty of this method lies in the combination of both texture and geometrical information for facial expression recognition.

1.2 System description

The diagram of the proposed system is shown in Figure **??**.

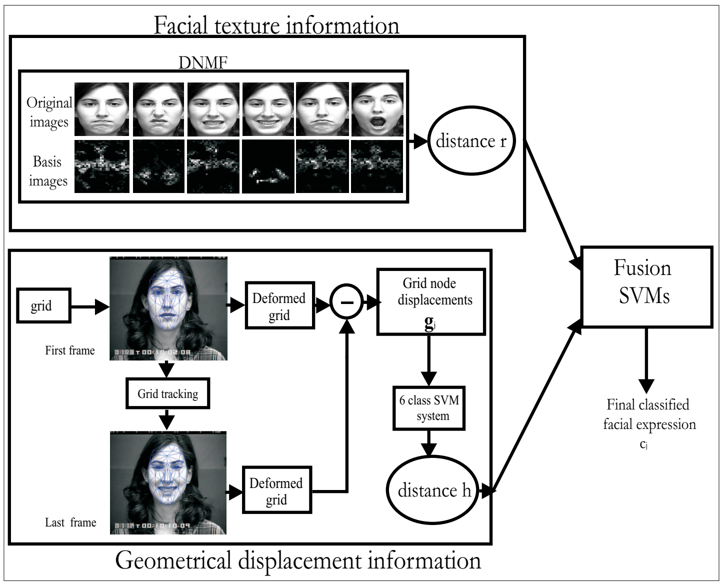

Fig. 1.1. System architecture for facial expression recognition in facial videos

The system is composed of three subsystems: two responsible for texture and shape information extraction and a third one responsible for the fusion of their results. Figure **??** shows the two sources of information (texture and shape) used by the system.

1.3 Texture information extraction

Let $\mathcal{U}$ be a database of facial videos. The facial expression depicted in each video sequence is dynamic, evolving through time as the video progresses. We

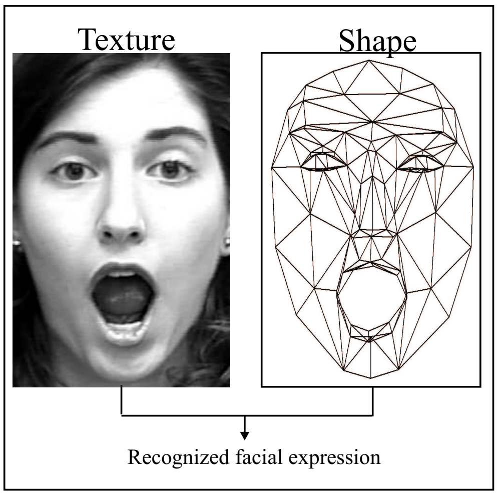

Fig. 1.2. Fusion of texture and shape.

take under consideration the frame that depicts the facial expression in its greatest intensity, i.e. the last frame, to create a facial image database Y. Thus, $\mathcal{Y}$ consists of images where the depicted facial expression obtains its greatest intensity . Each image $\mathbf{y} \in \mathcal{Y}$ belongs to one of the 6 basic facial expression classes$\{\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_6\}$ with $\mathcal{Y} = \bigcup_{r=1}^{6} \mathcal{Y}_r$. Each image $\mathbf{y} \in \Re_+^{K \times G}$ of dimension $F = K \times G$ forms a vector $\mathbf{x} \in \Re_+^F$. The vectors $\mathbf{x} \in \Re_+^F$ will be used in our algorithm.

The algorithm used was the DNMF algorithm, which is a extension of the Non-negative Matrix Factorization (NMF) algorithm. The NMF algorithm algorithm is an object decomposition algorithm that allows only additive combinations of non negative components. DNMF was the result of an attempt to introduce discriminant information to the NMF decomposition. Both NMF and DNMF algorithms will be presented analytically below.

1.3.1 The Non-negative Matrix Factorization Algorithm

A facial image $\mathbf{x}_j$ after the NMF decomposition can be written as $\mathbf{x}_j \approx \mathbf{Z}\mathbf{h}_j$, where $\mathbf{h}_j$ is the j-th column of $\mathbf{H}$. Thus, the columns of the matrix $\mathbf{Z}$ can be considered as basis images and the vector $\mathbf{h}_j$ as the corresponding weight vectors. Vectors $\mathbf{h}_j$ can also be considered as the projections vectors of the original facial vectors $\mathbf{x}_j$ on a lower dimensional feature space .

In order to apply NMF in the database $\mathcal{Y}$, the matrix $\mathbf{X} \in \Re_+^{F \times G} = [x_{i,j}]$ should be constructed, where $x_{i,j}$ is the i-th element of the j-th image, F is the number of pixels and G is the number of images in the database. In other words the j-th column of $\mathbf{X}$ is the $\mathbf{x}_j$ facial image in vector form (i.e. $\mathbf{x}_j \in \Re_+^F$).

NMF aims at finding two matrices $\mathbf{Z} \in \Re_{+}^{F\times M} = [z_{i,k}]$ and $\mathbf{H} \in \Re_{+}^{M\times L} = [h_{k,j}]$ such that :

$$\mathbf{X} \approx \mathbf{ZH}. \tag{1.1}$$

where M is the number of dimensions taken under consideration (usually $M \ll F$).

The NMF factorization is the outcome of the following optimization problem :

$$\min_{\mathbf{Z},\mathbf{H}} D_N(\mathbf{X}||\mathbf{ZH}) \text{ subject to} \tag{1.2}$$
$$z_{i,k} \geq 0,\ h_{k,j} \geq 0,\ \sum_i z_{i,j} = 1,\ \forall j.$$

The update rules for the weight matrix $\mathbf{H}$ and the bases matrix $\mathbf{Z}$ can be found in [**?**].

1.3.2 The Discriminant Non-negative Matrix Factorization Algorithm

In order to incorporate discriminants constraints inside the NMF cost function (**??**), we should use the information regarding the separation of the vectors $\mathbf{h}_j$ into different classes. Let us assume that the vector $\mathbf{h}_j$ that corresponds to the jth column of the matrix $\mathbf{H}$, is the coefficient vector for the ρth facial image of the rth class and will be denoted as $\boldsymbol{\eta}_{\rho}^{(r)} = [\eta_{\rho,1}^{(r)} \dots \eta_{\rho,M}^{(r)}]^T$. The mean vector of the vectors $\boldsymbol{\eta}_{\rho}^{(r)}$ for the class r is denoted as $\boldsymbol{\mu}^{(r)} = [\mu_1^{(r)} \dots \mu_M^{(r)}]^T$ and the mean of all classes as $\boldsymbol{\mu} = [\mu_1 \dots \mu_M]^T$. The cardinality of a facial class $\mathcal{Y}_r$ is denoted by N_r. Then, the within scatter matrix for the coefficient vectors $\mathbf{h}_j$ is defined as:

$$\mathbf{S}_w = \sum_{r=1}^{6} \sum_{\rho=1}^{N_r} (\boldsymbol{\eta}_{\rho}^{(r)} - \boldsymbol{\mu}^{(r)})(\boldsymbol{\eta}_{\rho}^{(r)} - \boldsymbol{\mu}^{(r)})^T \tag{1.3}$$

whereas the between scatter matrix is defined as:

$$\mathbf{S}_b = \sum_{r=1}^{6} N_r(\boldsymbol{\mu}^{(r)} - \boldsymbol{\mu})(\boldsymbol{\mu}^{(r)} - \boldsymbol{\mu})^T. \tag{1.4}$$

The discriminant constraints are incorporated by requiring $\text{tr}[\mathbf{S}_w]$ to be as small as possible while $\text{tr}[\mathbf{S}_b]$ is required to be as large as possible.

$$D_d(\mathbf{X}||\mathbf{Z}_D\mathbf{H}) = D_N(\mathbf{X}||\mathbf{Z}_D\mathbf{H}) + \gamma\text{tr}[\mathbf{S}_w] - \delta\text{tr}[\mathbf{S}_b]. \tag{1.5}$$

where γ and δ are constants and D is the measure of the cost for factoring $\mathbf{X}$ into $\mathbf{ZH}$ [**?**].

Following the same Expectation Maximization (EM) approach used by NMF techniques [**?**], the following update rules for the weight coefficients $h_{k,j}$ that belong to the r-th facial class become:

$$h_{k,j}^{(t)} = \frac{T_1 + \sqrt{T_1^2 + 4(2\gamma - (2\gamma + 2\delta)\frac{1}{N_r})h_{k,j}^{(t-1)}}}{2(2\gamma - (2\gamma + 2\delta)\frac{1}{N_r})}$$

$$\frac{\sum_i z_{i,k}^{(t-1)} \frac{x_{i,j}}{\sum_l z_{i,l}^{(t-1)} h_{l,j}^{(t-1)}}}{2(2\gamma - (2\gamma + 2\delta)\frac{1}{N_r})}. \tag{1.6}$$

where T_1 is given by:

$$T_1 = (2\gamma + 2\delta)(\frac{1}{N_r} \sum_{\lambda, \lambda \neq l} h_{k,\lambda}) - 2\delta\mu_k - 1. \tag{1.7}$$

The update rules for the bases $\mathbf{Z}_D$, are given by:

$$\acute{z}_{i,k}^{(t)} = z_{i,k}^{(t-1)} \frac{\sum_j h_{k,j}^{(t)} \frac{x_{i,j}}{\sum_l z_{i,l}^{(t-1)} h_{l,j}^{(t)}}}{\sum_j h_{k,j}^{(t)}} \tag{1.8}$$

and

$$z_{i,k}^{(t)} = \frac{\acute{z}_{i,k}^{(t)}}{\sum_l \acute{z}_{l,k}^{(t)}}. \tag{1.9}$$

The above decomposition is a supervised non-negative matrix factorization method that decomposes the facial images into parts while, enhancing the class separability. The matrix $\mathbf{Z}_D^{\dagger} = (\mathbf{Z}_D^T \mathbf{Z}_D)^{-1} \mathbf{Z}_D^T$, which is the pseudo-inverse of $\mathbf{Z}_D$, is then used for extracting the discriminant features as $\acute{\mathbf{x}} = \mathbf{Z}_D^{\dagger}\mathbf{x}$. The most interesting property of DNMF algorithm is that it decomposes the image to facial areas, i.e. mouth, eyebrows, eyes, and focuses on extracting the information hiding in them. Thus, the new representation of the image is a better one compared to the one acquired when the whole image was taken under consideration.

For testing, the facial image $\mathbf{x}_j$ is projected on the low dimensional feature space produced by the application of the DNMF algorithm:

$$\acute{\mathbf{x}}_j = \mathbf{Z}_D^{\dagger}\mathbf{x}_j \tag{1.10}$$

For the projection of the facial image $\acute{\mathbf{x}}_j$, one distance from each center class is calculated. The smallest distance defined as:

$$r_j = \min_{k=1,\ldots,6} \|\acute{\mathbf{x}}_j - \boldsymbol{\mu}^{(r)}\| \tag{1.11}$$

is the one that is taken as the output of the DNMF system.

1.4 Shape information extraction

The geometrical information extraction is done by a grid tracking system, based on deformable models [**?**]. The tracking is performed using a pyramidal implementation of the well-known Kanade-Lucas-Tomasi (KLT) algorithm. The user has to place manually a number of Candide grid nodes on the corresponding positions of the face depicted at the first frame of the image sequence. The algorithm automatically adjusts the grid to the face and then tracks it through the image sequence, as it evolves through time. At the end, the grid tracking algorithm produces the deformed Candide grid that corresponds to the last frame i.e. the one that depicts the greatest intensity of the facial expression.

The shape information used from the j video sequence is the displacements $\mathbf{d}_j^i$ of the nodes of the Candide grid, defined as the difference between coordinates of this node in the first and last frame [**?**]:

$$\mathbf{d}_j^i = [\Delta x_j^i \Delta y_j^i]^T \quad i \in \{1, \ldots, K\} \quad \text{and} \quad j \in \{1, \ldots, N\} \tag{1.12}$$

where i is an index that refers to the node under consideration. In our case $K = 104$ nodes were used.

For every facial video in the training set, a feature vector $\mathbf{g}_j$ of $F = 2{\cdot}104 = 208$ dimensions, containing the geometrical displacements of all grid nodes is created:

$$\mathbf{g}_j = [\mathbf{d}_j^1 \quad \mathbf{d}_j^2 \quad \ldots \quad \mathbf{d}_j^K]^T. \tag{1.13}$$

Let $\mathcal{U}$ be the video database that contains the facial videos, that are clustered into 6 different classes $\mathcal{U}_k,\ k = 1, \ldots, 6$, each one representing one of 6 basic facial expressions. The feature vectors $\mathbf{g}_j \in \Re^F$ labelled properly with the true corresponding facial expression are used as an input to a multi class SVM and will be described in the following section.

1.4.1 Support Vector Machines

Consider the training data:

$$(\mathbf{g}_1, l_1), \ldots, (\mathbf{g}_N, l_N) \tag{1.14}$$

where $\mathbf{g}_j \in \Re^F \quad j = 1, \ldots, N$ are the deformation feature vectors and $l_j \in \{1, \ldots, 6\} \quad j = 1, \ldots, N$ are the facial expression labels of the feature vector. The approach implemented for multiclass problems used for direct facial expression recognition is the one described in [**?**] that solves only one optimization problem for each class (facial expression). This approach constructs 6 two-class rules where the $k-$th function $\mathbf{w}_k^T \phi(\mathbf{g}_j) + b_k$ separates training vectors of the class k from the rest of the vectors. Here ϕ is the function that maps the deformation vectors to a higher dimensional space (where the data are supposed to be linearly or near linearly separable) and $\mathbf{b} = [b_1 \ldots b_6]^T$ is the bias vector. Hence, there are 6 decision functions, all obtained by solving a different SVM problem for each class. The formulation is as follows:

$$\min_{\mathbf{w},\mathbf{b},\boldsymbol{\xi}} \qquad \frac{1}{2}\sum_{k=1}^{6} \mathbf{w}_k^T \mathbf{w}_k + C \sum_{j=1}^{N} \sum_{k \neq l_j} \xi_j^k \tag{1.15}$$

subject to the constraints:

$$\begin{aligned} \mathbf{w}_{l_j}^T \phi(\mathbf{g}_j) + b_{l_j} &\geq \mathbf{w}_k^T \phi(\mathbf{g}_j) + b_k + 2 - \xi_j^k \\ \xi_j^k \geq 0, \quad j = 1, \dots, N, &\quad k \in \{1, \dots, 6\} \backslash l_j. \end{aligned} \tag{1.16}$$

where C is the penalty parameter for non linear separability and $\boldsymbol{\xi} = [\dots, \xi_i^m, \dots]^T$ is the slack variable vector. Then, the function used to calculate the distance of a sample from each center class is defined as:

$$s(\mathbf{g}) = \max_{k=1,\dots,6} (\mathbf{w}_k^T \phi(\mathbf{g}) + b_k). \tag{1.17}$$

That distance was considered as the output of the SVM based shape extraction procedure. A linear kernel was used for the SVM system in order to avoid search for appropriate kernels.

1.5 Fusion of texture and shape information

The application of the DNMF algorithm on the images of the database resulted in the extraction of the texture information of the facial expressions depicted. Similarly, the classification procedure performed using the SVM system on the grid following the facial expression through time resulted in the extraction of the shape information .

More specifically, the image $\mathbf{x}_j$ and the corresponding vector of geometrical displacements $\mathbf{g}_j$ were taken into consideration. The DNMF algorithm, applied to the $\mathbf{x}_j$ image, produces the distance r_j as a result, while SVMs applied to the vector of geometrical displacements $\mathbf{g}_j$, produces the distance s_j as the equivalent result. The distances r_j and s_j were normalized in $[0, 1]$ using Gaussian normalization. Thus, a new feature vector $\mathbf{c}_j$, defined as:

$$\mathbf{c}_j = [r_j \quad s_j]^T. \tag{1.18}$$

containing information from both sources was created.

1.5.1 Radial Basis Function (RBF) Neural Networks (NNs)

A RBF NN was used for the fusion of texture and shape results. The RBF function is approximated as a linear combination of a set of basis functions [**?**]:

$$p_k(\mathbf{c}_j) = \sum_{n=1}^{M} w_{k,n} \phi_n(\mathbf{c}_j) \tag{1.19}$$

where M is the number of kernel functions and $w_{k,n}$ are the weights of the hidden unit to output connection. Each hidden unit implements a Gaussian function:

$$\phi_n(\mathbf{c}_j) = exp[-(\mathbf{m}_n - \mathbf{c}_j)^T \mathbf{\Sigma}_n^{-1} (\mathbf{m}_n - \mathbf{c}_j)] \tag{1.20}$$

where $j = 1, \ldots M$, $\mathbf{m}_n$ is the mean vector and $\mathbf{\Sigma}_n$ is the covariance matrix [?].

Each pattern $\mathbf{c}_j$ is considered assigned only to one class l_j. The decision regarding the class l_j of $\mathbf{c}_j$ is taken as:

$$l_j = \underset{k=1,\ldots,6}{\operatorname{argmax}} \, p_k(\mathbf{c}_j) \tag{1.21}$$

The feature vector $\mathbf{c}_j$ was used as an input to the RBF NN that was created. The output of that system was the label l_j that classified the sample under examination (pair of texture and shape information) to one of the 6 classes (facial expressions).

1.6 Experimental results

In order to create the training set, the last frames of the video sequences used were extracted. By doing so, two databases were created, one for texture extraction using DNMF and another one for shape extraction using SVMs. The texture database consisted of images that corresponded to the last frame of every video sequence studied, while the shape database consisted of the grid displacements that were noticed between the first and the last frame of every video sequence.

The databases were created using a subset of the Cohn-Kanade database that consists of 222 image sequences, 37 samples per facial expression. The leave-one-out method was used for the experiments [?]. For the implementation of the RBF NN, 25 neurons were used for the output layer and 35 for the hidden layer.

The accuracy achieved when only DNMF was applied was equal to 86,5%, while the equivalent one when SVMs along with shape information were used was 93,5%. The obtained accuracy after performing fusion of the two information sources was equal to 98,2%. By fusing texture information into the shape results certain confusions are resolved. For example, some facial expressions involve subtle facial movements. That results in confusion with other facial expressions when only shape information is used. By introducing texture information, those confusions are eliminated. For example, in the case of anger, a subtle eyebrow movement is involved which cannot probably be identified as movement, but would most probably be noticed if texture is available. Therefore, the fusion of shape and texture information results in correctly classifying most of the confused cases, thus increasing the accuracy rate.

The confusion matrix [?] has been also computed. It is a $n \times n$ matrix containing information about the actual class label l_j, $j = 1, .., n$ (in its columns) and the label obtained through classification o_j, $j = 1, .., n$ (in its rows). The diagonal entries of the confusion matrix are the numbers of facial expressions that are correctly classified, while the off-diagonal entries correspond to misclassification. The confusions matrices obtained when using DNMF on texture information, SVM on shape information and when the proposed fusion is applied are presented in Table **??**.

Table 1.1. Confusion matrices for DNMF results, SVMs results and fusion results, respectively.

$lab_{cl} \backslash lab_{ac}$	anger	disgust	fear	happiness	sadness	surprise
anger	13	0	0	0	0	0
disgust	10	37	0	0	0	0
fear	4	0	37	0	0	1
happiness	2	0	0	37	0	0
sadness	7	0	0	0	37	5
surprise	1	0	0	0	0	31

$lab_{cl} \backslash lab_{ac}$	anger	disgust	fear	happiness	sadness	surprise
anger	24	0	0	0	0	0
disgust	5	37	0	0	0	0
fear	0	0	37	0	0	1
happiness	0	0	0	37	0	0
sadness	8	0	0	0	37	0
surprise	0	0	0	0	0	36

$lab_{cl} \backslash lab_{ac}$	anger	disgust	fear	happiness	sadness	surprise
anger	33	0	0	0	0	0
disgust	2	37	0	0	0	0
fear	0	0	37	0	0	0
happiness	0	0	0	37	0	0
sadness	2	0	0	0	37	0
surprise	0	0	0	0	0	37

1.7 Conclusions

A novel method for facial expression recognition is proposed in this paper. The recognition is performed by fusing the texture and the shape information extracted from a video sequence. The DNMF algorithm is applied at the last frames of every video sequence corresponding to the greatest intensity of the facial expression, extracting that way the texture information. Simultaneously, a SVM system classifies the shape information obtained by tracking the Candide grid between the first (neutral) and last (fully expressed facial expression) video frame. The results obtained from the above mentioned methods are then fused using RNF NNs. The system achieves an accuracy of 98,2% when recognizing the six basic facial expressions.

1.8 Acknowledgment

This work has been conducted in conjunction with the "SIMILAR" European Network of Excellence on Multimodal Interfaces of the IST Programme of the European Union (www.similar.cc).

References

1. P. Ekman, and W.V. Friesen, "Emotion in the Human Face," *Prentice Hall*, 1975.
2. T. Kanade, J. Cohn, and Y. Tian, "Comprehensive Database for Facial Expression Analysis," *Proceedings of IEEE International Conference on Face and Gesture Recognition*, 2000.
3. B. Fasel, and J. Luettin, "Automatic Facial Expression Analysis: A Survey," *Pattern Recognition*, 2003.
4. S. Zafeiriou, A. Tefas, I. Buciu and I. Pitas, "Exploiting Discriminant Information in Non-negative Matrix Factorization with application to Frontal Face Verification," *IEEE Transactions on Neural Networks*, accepted for publication, 2005.
5. D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," *Advances in Neural Information Processing Systems*, vol. 13pp. 556−562, 2001.
6. I. Kotsia, and I. Pitas, "Real time facial expression recognition from image sequences using Support Vector Machines," *IEEE International Conference on Image Processing (ICIP 2005)*, 11-14 September, 2005.
7. V. Vapnik, "Statistical learning theory," 1998.
8. A. G. Bors and I. Pitas, "Median Radial Basis Function Neural Network," *IEEE Transactions on Neural Networks*, vol. 7, pp. 1351-1364, November 1996.

Limited Receptive Area neural classifier for texture recognition of metal surfaces

Oleksandr Makeyev[1], Tatiana Baidyk[2] and Anabel Martín[2]
1 National Taras Shevchenko University of Kyiv,
64, Volodymyrska Str., 01033, Kiev, Ukraine
mckehev@hotmail.com
2 Center of Applied Sciences and Technological Development, National Autonomous University of Mexico, Cd. Universitaria, Circuito Exterior s/n, Coyoacán, 04510, México, D.F., Mexico
tbaidyk@aleph.cinstrum.unam.mx; anabelmartin@lycos.com

Abstract. The Limited Receptive Area (LIRA) neural classifier is proposed for texture recognition of mechanically treated metal surfaces. It can be used in systems that have to recognize position and orientation of complex work pieces in the task of assembly of micromechanical devices. The performance of the proposed classifier was tested on specially created image database in recognition of four texture types that correspond to metal surfaces after: milling, polishing with sandpaper, turning with lathe and polishing with file. The promising recognition rate of 99.7% was obtained.

1 Introduction

The main approaches to microdevice production are the technology of micro electromechanical systems (MEMS) [1, 2] and microequipment technology (MET) [3-6]. To get the best of these technologies it is important to have advanced image recognition systems.

Texture recognition systems are widely used in industrial inspection, for example, in textile industry for detection of fabric defects [7], in electronic industry for inspection of the surfaces of magnetic disks [8], in decorative and construction industry for inspection of polished granite and ceramic titles [9], etc.

Numerous approaches were developed to solve the texture recognition problem. Many statistical texture descriptors are based on a generation of co-occurrence matrices. In [8] the texture co-occurrence of *n*-th rank was proposed. The matrix contains statistics of the pixel under investigation and its surrounding. Another approach was proposed in [9]. The authors proposed the coordinated cluster representation (CCR) as a technique of texture feature extraction. The underlying principle of the CCR is to extract a spatial correlation between pixel intensities using the distribution function of the occurrence of texture units. Experiments with one-layer texture classifier in the CCR feature space prove this approach to be very promising. Leung et al. [10] proposed textons (representative texture elements) for texture description and recognition. The vocabulary of textons corresponds to the characteristic features of the image. There are many works on applying neural networks in texture recognition problem [11, 12].

In this paper we propose the LIRA neural classifier [4] for metal surface texture recognition. Four types of metal surfaces after mechanical treatment were used to test the proposed texture recognition system.

Different lighting conditions and viewing angles affect the grayscale properties of an image due to such effects as shading, shadowing, local occlusions, etc. The real metal surface images that it is necessary to recognize in industry have all these problems and what is more there are some problems specific for industrial environment, for example, metal surface can have dust on it.

The reason to choose a system based on neural network architecture for the current task was that such systems have already proved their efficacy in texture recognition due to significant properties of adaptability and robustness to texture variety [13].

We have chosen the LIRA neural classifier because we have already applied it in the flat image recognition problem in microdevice assembly and the results were very promising [4]. We have also tested it in handwritten digit recognition task and its recognition rate on the MNIST database was 0.55% [4] that is among the best results obtained on this database.

2 Metal surface texture recognition

The task of metal surface texture recognition is important to automate the assembly processes in micromechanics [3]. To assembly a device it is necessary to recognize the position and orientation of the work pieces to be assembled [4]. It is useful to identify the surface of a work piece to recognize its position and orientation. For example, let the shaft have two polished cylinder surfaces for bearings, one of them milled with grooves for dowel joint, and the other one turned with the lathe. It will be easier to obtain the orientation of the shaft if we can recognize both types of the surface textures.

There are works on fast detection and classification of defects on treated metal surfaces using a back propagation neural network [14], but we do not know any on texture recognition of metal surfaces after mechanical treatment.

To test our texture recognition system we created our own image database of metal surface images. Four texture classes correspond to metal surfaces after: milling, polishing with sandpaper, turning with lathe and polishing with file (Fig. 1). It can be seen that different lighting conditions affect greatly the grayscale properties of the images. The textures may also be arbitrarily oriented and not centered perfectly. Metal surfaces may have minor defects and dust on it. All this image properties correspond to the conditions of the real industrial environment and make the texture recognition task more complicated. Two out of four texture classes that correspond to polishing with sandpaper and to polishing with file sometimes can be hardly distinguished with the naked eye (Fig. 1, columns *b* and *d*).

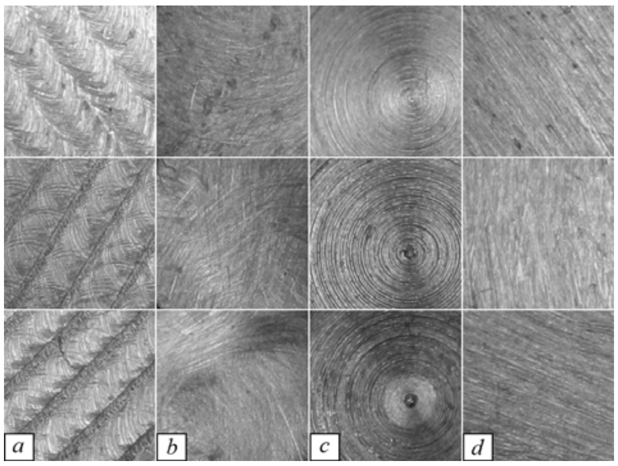

Fig. 1. Examples of metal surfaces after (columns): a) milling, b) polishing with sandpaper, c) turning with lathe, d) polishing with file

3 The LIRA neural classifier

The LIRA neural classifier [4] was developed on the basis of the Rosenblatt perceptron [15]. The three-layer Rosenblatt perceptron consists of the sensor *S*-layer, associative *A*-layer and the reaction *R*-layer. The first *S*-layer corresponds to the retina. In technical terms it corresponds to the input image. The second *A*-layer corresponds to the feature extraction subsystem. The third *R*-layer represents the system's output. Each neuron of this layer corresponds to one of the output classes.

The associative layer *A* is connected to the sensor layer *S* with the randomly selected, non-trainable connections. The weights of these connections can be equal either to 1 (positive connection) or to -1 (negative connection). The set of these connections can be considered as a feature extractor.

A-layer consists of 2-state neurons; their outputs can be equal either to 1 (active state) or to 0 (non-active state). Each neuron of the *A*-layer is connected to all the neurons of the *R*-layer. The weights of these connections are modified during the perceptron training.

We have made four major modifications in the original perceptron structure. These modifications concern random procedure of arrangement of the *S*-layer connections, the adaptation of the classifier to grayscale image recognition, the training procedure and the rule of winner selection.

We propose two variants of the LIRA neural classifier: LIRA_binary and LIRA_grayscale. The first one is meant for the recognition of binary (black and white) images and the second one for the recognition of grayscale images. The structure of the LIRA_grayscale neural classifier is presented in Fig. 2.

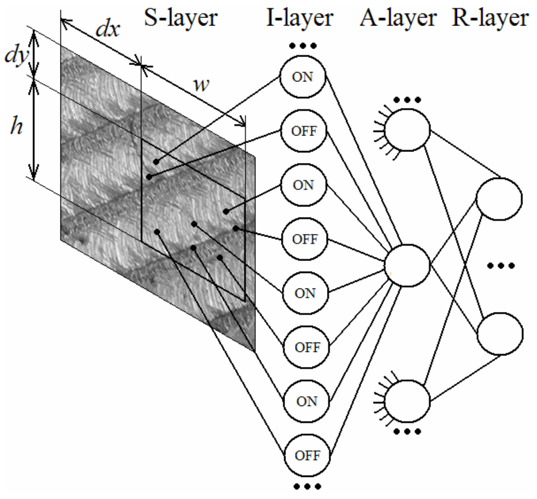

Fig. 2. The structure of the LIRA_grayscale neural classifier

The one-layer perceptron has very good convergence but it demands the linear separability of the classes in the parametric space. To obtain linear separability it is necessary to transform the initial parametric space represented by pixel brightness to the parametric space of larger dimension. In our case the connections between the *S*-layer and the *A*-layer transform initial ($W_S \cdot H_S$)-D space (W_S and H_S stand for width and height of the *S*-layer) into *N*-dimension space represented by binary code vector. In our experiments $W_S = H_S = 220$ and *N* varied from 64,000 to 512,000. Such transformation improves the linear separability. The coding procedure used in the LIRA classifier is the following.

3.1 Image coding

Each input image defines the activities of the *A*-layer neurons in one-to-one correspondence. The binary vector that corresponds to the associative neuron activities is termed the image binary code $A = (a_1, \ldots, a_N)$, where *N* is the number of the *A*-layer neurons. The procedure that transforms the input image into the binary vector *A* is termed the image coding.

We connect each *A*-layer neuron to *S*-layer neurons randomly selected not from the entire *S*-layer, but from the window $h \cdot w$ that is located in the *S*-layer (Fig. 2).

The distances *dx* and *dy* are random numbers selected from the ranges: *dx* from $[0,\ W_S - w)$ and *dy* from $[0,\ H_S - h)$. We create the associative neuron masks that represent the positions of connections of each *A*-layer neuron with neurons of the window $h \cdot w$. The procedure of random selection of connections is used to design the mask of *A*-layer neuron. This procedure starts with the selection of the upper left corner of the window $h \cdot w$ in which all connections of the associative neuron are located.

The following formulas are used:

$dx_i = random_i(W_S - w)$,

$dy_i = random_i(H_S - h)$,

where *i* is the position of a neuron in associative layer *A*, $random_i(z)$ is a random number that is uniformly distributed in the range [0, *z*). After that position of each connection within the window $h \cdot w$ is defined by the pair of numbers:

$x_{ij} = \mathrm{random}_{ij}(w)$,

$y_{ij} = random_{ij}(h)$,

where *j* is the number of the connection with the retina.

Absolute coordinates of the connection on the retina are defined by the pair of the numbers:

$X_{ij} = x_{ij} + dx_i$,

$Y_{ij} = y_{ij} + dy_i$.

To adapt the LIRA neural classifier for grayscale image recognition we have added the additional 2-state neuron layer between the *S*-layer and the *A*-layer. We term it the *I*-layer (intermediate layer, see Fig. 2).

The input of each *I*-layer neuron is connected to one neuron of the *S*-layer and the output is connected to the input of one neuron of the *A*-layer. All the *I*-layer neurons connected to one *A*-layer neuron form the group of this *A*-layer neuron. There are two types of *I*-layer neurons: ON-neurons and OFF-neurons. The output of the ON-neuron *i* is equal to 1 when its input value is larger than the threshold θ_i and it is equal to 0 in opposite case. The output of the OFF-neuron *j* is equal to 1 when its input value is smaller than the threshold θ_j and it is equal to 0 in opposite case. For example, in Fig. 2, the group of eight *I*-layer neurons, four ON-neurons and four OFF-neurons, corresponds to one *A*-layer neuron. The thresholds θ_i and θ_j are selected randomly from the range $[0, \eta \cdot b_{\max}]$, where $b_{\max}$ is maximal brightness of the image pixels, η is the parameter selected experimentally from the range [0, 1]. The *i*-th neuron of the *A*-layer is active ($a_i = 1$) only if outputs of all the neurons of its *I*-layer group are equal to 1 and is non-active ($a_i = 0$) in opposite case.

Taking into account the small number of active neurons it is convenient to represent the binary code vector not explicitly but as a list of numbers of active neurons. Let, for example, the vector *A* be:

A = 00010000100000010000.

The corresponding list of the numbers of active neurons will be 4, 9, and 16. Such compact representation of code vector permits faster calculations in training procedure. Thus, after execution of the coding procedure every image has a corresponding list of numbers of active neurons.

3.2 Training procedure

Before starting the training procedure the weights of all connections between neurons of the *A*-layer and the *R*-layer are set to 0. As distinct from the Rosenblatt perceptron our LIRA neural classifier has only non-negative connections between the *A*-layer and the *R*-layer.

The first stage. The training procedure starts with the presentation of the first image to the LIRA neural classifier. The image is coded and the *R*-layer neuron excitations E_i are computed. E_i is defined as:

$$E_i = \sum_{j=1}^{N} a_j \cdot w_{ji},$$

where E_i is the excitation of the *i*-th neuron of the *R*-layer, a_j is the output signal (0 or 1) of the *j*-th neuron of the *A*-layer, w_{ji} is the weight of the connection between the *j*-th neuron of the *A*-layer and the *i*-th neuron of the *R*-layer.

The second stage. Robustness of the recognition is one of the important requirements the classifier must satisfy. After calculation of the neuron excitations of the *R*-layer, the correct class *c* of the image under recognition is read. The excitation E_c of the corresponding neuron of the *R*-layer is recalculated according to the formula:

$$E_c* = E_c \cdot (1 - T_E),$$

where $0 \le T_E \le 1$ determines the reserve of excitation the neuron that corresponds to the correct class must have. In our experiments the value T_E varied from 0.1 to 0.5.

After that we select the neuron with the largest excitation. This winner neuron represents the recognized class.

The third stage. Let us denote the winner neuron number as *j* keeping the number of the neuron that corresponds to the correct class denoted as *c*. If *j* = *c* then nothing is to be done. If $j \neq c$ then following modification of weights is to be done:

$$w_{ic}(t+1) = w_{ic}(t) + a_i,$$

$$w_{ij}(t+1) = w_{ij}(t) - a_i, \text{ if } (w_{ij}(t+1) < 0) \text{ then } w_{ij}(t+1) = 0,$$

where $w_{ij}(t)$ and $w_{ij}(t + 1)$ are the weights of the connection between the *i*-th neuron of the *A*-layer and the *j*-th neuron of the *R*-layer before and after modification, a_i is the output signal (0 or 1) of the *i*-th neuron of the *A*-layer.

The training process is carried out iteratively. After all the images from the training set have been presented the total number of training errors is calculated. If this number is larger than one percent of the total number of images then the next training cycle is performed, otherwise training process is stopped. The training process is also stopped if the number of performed training cycles is more than a predetermined value.

It is obvious that in every new training cycle the image coding procedure is repeated and gives the same results as in previous cycles. Therefore in our experiments we performed the coding procedure only once and saved the lists of active neuron numbers for each image on the hard drive. Later, during the training procedure, we used not the images, but the corresponding lists of active neurons. Due to this approach, the training process was accelerated approximately by an order of magnitude.

It is known [16] that the performance of the recognition systems can be improved with implementation of distortions of the input image during the training process. In our experiments we used different combinations of horizontal, vertical and bias image shifts, skewing and rotation.

3.3 Recognition procedure

In our LIRA neural classifier we use image distortions not only in training but also in recognition process. There is an essential difference between implementation of distortions for training and recognition. In the training process each distortion of the initial image is considered as an independent new image. In the recognition process it is necessary to introduce a rule of decision-making in order to be able to make a decision about a class of the image under recognition based on the mutual information about this image and all its distortions. The rule of decision-making that we have used consists in calculation of the *R*-layer neuron excitations for all the distortions sequentially:

$$E_i = \sum_{k=0}^{d} \sum_{j=1}^{N} a_{kj} \cdot w_{ji},$$

where E_i is the excitation of the *i*-th neuron of the *R*-layer, a_{kj} is the output signal (0 or 1) of the *j*-th neuron of the *A*-layer for the *k*-th distortion of the initial image, w_{ji} is the weight of the connection between the *j*-th neuron of the *A*-layer and the *i*-th neuron of the *R*-layer, *d* is the number of applied distortions (case $k = 0$ corresponds to the initial image).

After that we select the neuron with the largest excitation. This winner neuron represents the recognized class.

4 Results

To test our texture recognition system we created our own image database of mechanically treated metal surfaces (see Section 2 for details). We work with four texture classes that correspond to metal surfaces after: milling, polishing with sandpaper, turning with lathe and polishing with file. 20 grayscale images of 220x220 pixels were taken for each class. We randomly divide these 20 images into the training and test sets for the LIRA_grayscale neural classifier. The number of images in training set varied from 2 to 10 images for each class.

All experiments were performed on a Pentium 4, 3.06 GHz computer with 1.00 GB RAM.

We carried out a large amount of preliminary experiments first to estimate the performance of our classifier and to tune the parameter values. On the basis of these preliminary experiments we selected the best set of parameter values and carried out final experiments to obtain the maximal recognition rate. In preliminary experiments the following parameter values were set: window $h \cdot w$ width $w = 10$, height $h = 10$, parameter that determines the reserve of excitation the neuron that corresponds to the correct class must have $T_E = 0.3$. The following distortions were chosen for the final experiments: 8 distortions for training including 1 pixel horizontal, vertical and bias image shifts and 4 distortions for recognition including 1 pixel horizontal and vertical image shifts. The number of training cycles was equal to 30.

The numbers of ON-neurons and OFF-neurons in the *I*-layer neuron group that corresponded to one *A*-layer neuron were chosen in order to keep the ratio between the number of active neurons *K* and the total number of associative neurons *N* within the limits of $K = c \cdot \sqrt{N}$, where *c* is the constant selected experimentally from the range [1, 5]. This ratio corresponds to neurophysiological data. The number of active neurons in the cerebral cortex is hundreds times less than the total number of neurons. For example, for the total number of associative neurons *N* = 512,000 we selected three ON-neurons and five OFF-neurons.

In each experiment we performed 50 runs to obtain statistically reliable results. That is, the total number of recognized images was calculated as number of images in test set for one run multiplied by 50. New mask of connections between the *S*-layer and the *A*-layer and new division into the training and test sets were created for the each run.

In the first stage of final experiments we changed the total number of associative neurons *N* from 64,000 to 512,000. The results are presented in Table 1. Taking into account that the amount of time needed for 50 runs of coding and classifier's training and recognition with *N* = 512,000 is approximately 3 h and 20 min we can conclude that such computational time is justified by the increase in the recognition rate. That is why we used *N* = 512,000 in all the posterior experiments.

Table 1. Dependency of the recognition rate on the total number of associative neurons

Total number of associative neurons	Number of errors / Total number of recognized images	% of correct recognition
64,000	20 / 2000	99
128,000	13 / 2000	99.35
256,000	8 / 2000	99.6
512,000	6 / 2000	99.7

In the second stage of final experiments we performed experiments with different combinations of distortions for training and recognition. The results are presented in Table 2. It can be seen that distortions used in training process have great impact on the recognition rate that is no wonder if to take into account that the use of 8 distortions for training allows to increase the size of training set 9 times. Distortions used in recognition process also have significant positive impact on the recognition rate.

Table 2. Dependency of the recognition rate on the distortions

Distortions		Number of errors / Total number of recognized images	% of correct recognition
Training	Recognition		
-	-	1299 / 2000	35.05
-	+	1273 / 2000	36.35
+	-	14 / 2000	99.3
+	+	6 / 2000	99.7

In the third stage of final experiments we performed experiments with different numbers of images in the training and test sets. The results are presented in Table 3. The note tr./t. reflects how many images were used for training (tr.) and how many for testing (t.). It can be seen that even in case of using only 2 images for training and 18 for recognition the LIRA_grayscale neural classifier gives a good recognition rate of 83.39%.

Table 3. Dependency of the recognition rate on the number of images in training set

tr./t.	Number of errors / Total number of recognized images	% of correct recognition
2/18	598 / 3600	83.39
4/16	174 / 3200	94.56
6/14	34 / 2800	98.78
8/12	8 / 2400	99.67
10/10	6 / 2000	99.7

5 Discussion

The LIRA neural classifier was tested in the task of texture recognition of mechanically treated metal surfaces. This classifier does not use floating point or multiplication operations. This property combined with the classifier’s parallel structure allows its implementation in low cost, high speed electronic devices. Sufficiently fast convergence of the training process and very promising recognition rate of 99.7% were obtained on the specially created image database (see Section 2 for details). There are quite a few methods that perform well when the features used for the recognition are obtained from a training set image that has the same orientation, position and lighting conditions as the test image; but as soon as orientation or position or lighting conditions of the test image is changed with respect to the one in the training set the same methods will perform poorly. The usefulness of methods that are not robust to such changes is very limited and that is the reason for developing of our texture classification system that works well independently of the particular orientation, position and lighting conditions. In this regard the results obtained in experiments are very promising.

6 Conclusion

This paper continues the series of works on automation of micro assembly processes [3, 4].

The LIRA neural classifier is proposed for texture recognition of mechanically treated metal surfaces. It can be used in systems that have to recognize position and orientation of complex work pieces in the task of assembly of micromechanical devices as well as in surface quality inspection systems. The performance of the proposed classifier was tested on specially created image database in recognition of four texture types that correspond to metal surfaces after: milling, polishing with sandpaper, turning with lathe and polishing with file. The promising recognition rate of 99.7% was obtained.

Acknowledgment

The authors gratefully acknowledge Dr. Ernst Kussul, National Autonomous University of Mexico, for constructive discussions and helpful comments.

This work was supported by projects PAPIIT 1112102, PAPIIT IN116306-3, PAPIIT IN108606-3, NSF-CONACYT 39395-A.

References

1. W.S. Trimmer (ed.), Micromechanics and MEMS. Classical and Seminal Papers to 1990 (IEEE Press, New York, 1997).
2. A.M. Madni, L.A. Wan, Micro Electro Mechanical Systems (MEMS): an Overview of Current State-of-the Art, Aerospace Conference **1**, 421-427 (1998).
3. E. Kussul, T. Baidyk, L. Ruiz-Huerta, A. Caballero, G. Velasco, L. Kasatkina, Development of Micromachine Tool Prototypes for Microfactories, J. Micromech. Microeng. **12**, 795-813 (2002).
4. T. Baidyk, E. Kussul, O. Makeyev, A. Caballero, L. Ruiz, G. Carrera, G. Velasco, Flat Image Recognition in the Process of Microdevice Assembly, Pattern Recogn. Lett. **25**(1), 107-118 (2004).
5. C.R. Friedrich, M.J. Vasile, Development of the Micromilling Process for High- aspect-ratio Micro Structures, J. Microelectromech. S. **5**, 33-38 (1996).
6. Naotake Ooyama, Shigeru Kokaji, Makoto Tanaka, et al, Desktop Machining Microfactory, Proc. of the 2-nd International Workshop on Microfactories, 14-17 (2000).
7. Chi-ho Chan, Grantham K.H. Pang, Fabric Defect Detection by Fourier Analysis, IEEE T. Ind. Appl. **36**(5), 1267-1276 (2000).
8. L. Hepplewhite, T.J. Stonham, Surface Inspection Using Texture Recognition, Proc. of the 12th IAPR International Conference on Pattern Recognition **1**, 589-591 (1994).
9. R. Sanchez-Yanez, E. Kurmyshev, A. Fernandez, One-class Texture Classifier in the CCR Feature Space, Pattern Recogn. Lett. **24**, 1503-1511 (2003).
10. T. Leung, J. Malik, Representing and Recognizing the Visual Appearance of Materials Using Three-dimensional Textons, Int. J. Comput. Vision **43**(1), 29-44 (2001).
11. M.A. Mayorga, L.C. Ludeman, Shift and Rotation Invariant Texture Recognition with Neural Nets, Proc. of the IEEE International Conference on Neural Networks **6**, 4078-4083 (1994).
12. Woobeom Lee, Wookhyun Kim, Self-organization Neural Networks for Multiple Texture Image Segmentation, Proc. of the IEEE Region 10 Conference TENCON 99 **1**, 730-733 (1999).
13. M.A. Kraaijveld, An Experimental Comparison of Nonparametric Classifiers for Time-constrained Classification Tasks, Proc. of the Fourteenth International Conference on Pattern Recognition **1**, 428-435 (1998).
14. C. Neubauer, Fast Detection and Classification of Defects on Treated Metal Surfaces Using a Back Propagation Neural Network, Proc. of the IEEE International Joint Conference on Neural Networks **2**, 1148-1153 (1991).
15. F. Rosenblatt, Principles of Neurodynamics (Spartan books, New York, 1962).
16. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based Learning Applied to Document Recognition, P. IEEE **86**(11), 2278-2344 (1998).

A Tracking Framework for Accurate Face Localization

Ines Cherif, Vassilios Solachidis and Ioannis Pitas

Department of Informatics, Aristotle University of Thessaloniki
Thessaloniki 54124, Greece.
Tel: +30-2310996304
{ines,vasilis,pitas}@aiia.csd.auth.gr

Abstract. This paper proposes a complete framework for accurate face localization on video frames. Detection and forward tracking are first combined according to predefined rules to get a first set of face candidates. Backward tracking is then applied to provide another set of possible localizations. Finally a dynamic programming algorithm is used to select the candidates that minimize a specific cost function. This method was designed to handle different scale, pose and lighting conditions. The experiments show that it improves the face detection rate compared to a frame-based detector and provides a higher precision than a forward information-based tracker.

1 Introduction

Achieving a good localization of faces on video frames is of high importance for an application such as video indexing and thus, multiple approaches were proposed to increase the face detection rate. In this paper, we introduce a new method making full use of the information provided by a backward tracking process and merging the latter with the detection and forward tracking results using a Dynamic Programming (DP) algorithm. Detection and forward tracking were associated in several research works to improve the detection rate [1]. Combining forward and backward tracking, on the other hand is a rather new idea. It is suitable for analyzing movie or prerecorded content, since in such cases, we have access to the entire video. An extension to particle filtering is described in [2]. In this probabilistic framework, the preliminary detected faces are propagated by sequential forward tracking. A backward propagation is then performed to refine the previous results. As for Dynamic Programming techniques, they are widely used to tackle various issues, among them motion estimation [3], feature extraction and object segmentation [4]. They were also used to perform the face detection and tracking, searching for the best matching region for a given face template [5]. In [6], a multiple object tracking is presented, where the Viterbi Algorithm is used to find the best path between candidates selected according to skin color criteria.
In this paper, a new deterministic approach is presented. It applies face detection, forward tracking and backward tracking, using some predefined rules.

From all the possible extracted candidates, a Dynamic Programming algorithm selects those that minimize a cost function.
The paper is organized as follows: Section 2 presents the new framework for the extraction and labelling of the candidates for the face localizations. Section 3 describes how the trellis structure is applied to select the trajectory with the lowest cost. Section 4 provides the results obtained on several video sequences and section 5 concludes the paper.

2 Tracking Framework

In order to achieve a high detection rate on each frame of a video sequence, detection and tracking algorithms were combined and some rules were defined to form a complete tracking framework.

2.1 Detection

The implemented face detector is based on Haar-like features [7]. The algorithm provides good detection results in case the orientation of the face is almost frontal. But it also produces some false alarms. Therefore, a postprocessing step is added for rejecting detected faces, if the number of skin-like pixels present in the detected bounding box is below a threshold. The region of the image containing the detected face is converted into the HSV color space and two morphological operations, erosion and dilation are performed, in order to remove the sparse pixels. The detection bounding box is then replaced by the smallest bounding box containing all the skin-like pixels. This operation helps removing a part of the background and thus better defining the tracked region.
The skin-like pixels are identified as those that fulfill the three following conditions:

$$0 < h < 0.1 \tag{1}$$

$$0.23 < s < 0.68 \tag{2}$$

$$0.27 < v \tag{3}$$

where h, s and v are the coordinates of the HSV color space. This approach is similar to the one used in [8].
The detection process is applied on the first and last frame of a shot and every five frames within the shot. This detection frequency appears to provide satisfactory results. Ideally, if a person is once correctly located in each shot, then the forthcoming processes will provide the missing localizations in the other frames.

2.2 Forward Tracking

To be able to localize faces on every video frame, a forward tracking process is performed on each frame, starting from frames where faces have been detected. The tracking algorithm used is the one described in [9], based on the so-called morphological elastic graph matching (EGM) algorithm. It is initialized by the output of the face detection algorithm and the faces can then be tracked until the next detection of the same face or until the end of the shot, if the faces are not detected again.
In fact, one face can be detected several times in a shot, this can lead to multiple tracking of a same actor, which is time consuming. To overcome this problem, a tracking rule is used in order to identify if newly detected faces correspond to previously tracked faces. This rule is based on the percentage of overlap P_{over} between the detected bounding boxes (D_i) and the ones resulting from the forward tracking (F) in the same frame. We define P_{over} as follows:

$$P_{over}(F) = \max_i \frac{A_{(F \bigcap D_i)}}{\min(A_{D_i}, A_F)} \tag{4}$$

where A_{D_i} is the area of the i^{th} detection bounding box and A_F is the area of the forward tracking bounding box. As for $A_{(F \bigcap D_i)}$, it corresponds to the area recovered by both bounding boxes. If P_{over} is higher than 70%, the two bounding boxes correspond to the same actor and the new detection is used to re-initialize the tracker.
This rule is illustrated on Fig 1. On the first frame of the shot, D_1 represents a detected face and is associated to a first actor. The forward tracking of the detected face is performed until the next detection frame and the bounding boxes are assigned the same label (Actor 1). On the next detection frame, D_2 and D_3 are compared to the tracking bounding box on the same frame. The face that fulfills the overlap condition (D_3) is assigned the same label (Actor 1) while the other (D_2) is associated to a new actor (Actor 2). This rule is applied to the other detections D_4 and D_5 as well.

2.3 Backward Tracking

In order to provide a new set of face candidates, a backward tracking process is performed on each frame. The tracker is initialized by the face detection results as shown in Fig 1. This backward process is very useful in case a face is not detected at the beginning but in the middle of a shot. The forward tracking provides the bounding box localizations from the detection frame to the end of the shot. As for the backward tracking, it will provide the missing results from the first frame of the shot to the frame where the last face detection has been performed.
A more interesting contribution of the backward tracking is obtained when the forward tracking or the detection process fails to accurately locate the face of

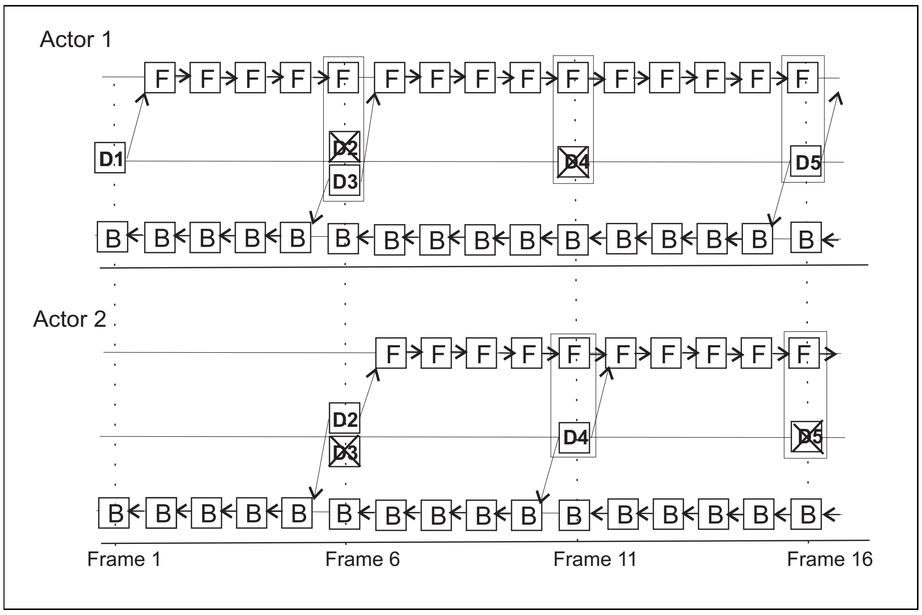

Fig. 1. Illustration of the tracking rule. (D): Detection bounding boxes, (F): Forward tracking bounding boxes and (B): Backward tracking bounding boxes

an actor on a frame i, due for instance to an occlusion, bad illumination or if the tracker sticks to the background. If the next detection of this same actor on the frame $(i + 5n,\ n \in \mathrm{N}^{*})$ is more precise, then this information will be propagated back and might generate, on i, a new face candidate with a higher accuracy.
Proceeding this way, we will get one, two or three candidates per frame for the face localization, corresponding to respectively the face detection, forward tracking and backward tracking results.

3 A trellis structure for optimal face detection

Now in order to improve face localization, Dynamic Programming is used as a postprocessing. In Section 2, each bounding box was assigned a label. Therefore a trellis can be defined for each actor as represented in Fig 2. The labels D, F and B define the states of the trellis diagram. The frames, where face detection took place can have states D, F and B, while the other frames can have states F and B only.
The complexity of the trellis is considerably reduced in comparison with other approaches that draw the trellis using all the bounding boxes provided by the detector or the tracker [6]. In fact, the number of possible paths in the trellis grows exponentially with the number of nodes. Therefore, limiting the number

of candidates to three is a major advantage of this method.

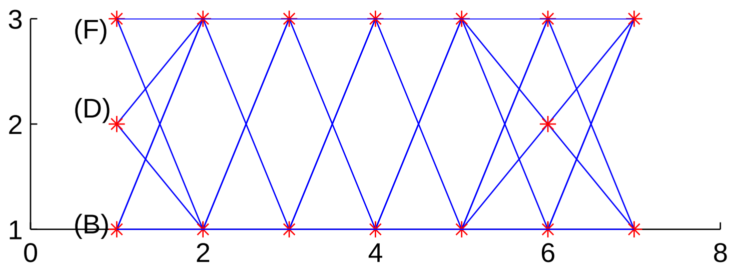

Fig. 2. Model of trellis with 7 frames ($N = 7$). (D): Detection results,(F): Forward tracking results and (B): Backward tracking results

3.1 Cost

Finding the optimal face detection/tracking is equivalent to a best path extraction from a trellis. For each frame of the video sequence we have one, two or three states representing the face candidates provided by the face detection/tracking framework. The cost of a path until the frame l can be expressed as follows:

$$C(l) = -\sum_{i=1}^{l} C(s_i) - \sum_{i=2}^{l} C(s_{i-1}, s_i) \tag{5}$$

For each edge connecting a state s_{i-1}(corresponding to a bounding box B_{i-1} in the previous frame) to another state s_i(corresponding to a bounding box B_i in the current frame) we define the transition cost $C(s_{i-1}, s_i)$ as a combination of two metrics $C_1(s_{i-1}, s_i)$ and $C_2(s_{i-1}, s_i)$:

1. The first cost C_1 takes into account the overlap between the bounding boxes referenced B_i and B_{i-1}.

$$\mathcal{O}(B_{i-1}, B_i) = \frac{A_{(B_{i-1} \bigcap B_i)}}{\min(A_{B_{i-1}}, A_{B_i})} \tag{6}$$

where A_{B_i} is the area of the bounding box B_i. $A_{(B_{i-1} \bigcap B_i)}$ represents the area of the intersection of the bounding boxes B_i and B_{i-1}. We will assume that the bounding boxes of two consecutive frames must have a non-zero overlap. C_1 will take a $-\infty$ value in order to forbid the transition between non-overlapping bounding boxes.

$$C_1(s_{i-1}, s_i) = \begin{cases} \mathcal{O}(B_i, B_{i+1}), \text{ if } \mathcal{O}(B_i, B_{i+1}) > 0 \\ -\infty, \text{ otherwise} \end{cases} \tag{7}$$

Practically, a very small negative value will suffice.

2. The cost C_2 is equal to the ratio between the areas of the bounding boxes as specified by Eq.8. This metric penalizes big changes of the bounding box area during tracking.

$$C_2(s_{i-1}, s_i) = \frac{min(A_{B_{i-1}}, A_{B_i})}{max(A_{B_{i-1}}, A_{B_i})} \tag{8}$$

The transition cost $C(s_{i-1}, s_i)$ is then deduced from $C_1(s_{i-1}, s_i)$ and $C_2(s_{i-1}, s_i)$ e.g. by simple multiplication.
To obtain now the node cost $C(s_i)$, we compute the distance between the center of the bounding box (x_{c_i}, y_{c_i}) and the centroid $(\overline{x}, \overline{y})$ of the skin-like pixels.

$$C(s_i) = \exp\left(-\frac{\sqrt{(\overline{x} - x_{c_i})^2 + (\overline{y} - y_{c_i})^2}}{\sqrt{H^2 + W^2}}\right) \tag{9}$$

with H and W being the height and width of the frame.
The position of the centroid is defined as follows:

$$\overline{x} = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} jA(i,j) \tag{10}$$

$$\overline{y} = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} iA(i,j) \tag{11}$$

where A is an $n \times m$ matrix, whose elements take the value 1 when the corresponding pixel in the bounding box B_i is skin-like and 0 otherwise.

Once both node and transition costs are defined, the optimal path will be extracted as follows. For each node on the frame l, the accumulate cost $C(l)$ from the first frame to l is calculated using the accumulate cost $C(l-1)$ to the different states in the frame $l-1$. The lowest cost provides the shortest path to the current node and the sequence of nodes leading to this cost are memorized. This process is iterated until the last frame. The shortest path is then retrieved by backtracking the path to the first frame. An example of optimal path is presented on Fig 3 for 30 video frames.

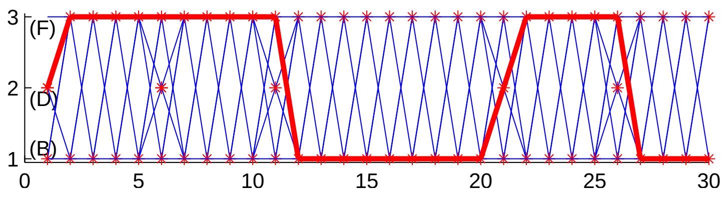

Fig. 3. Shortest path extracted from a 30-frame trellis.

4 Experiments and results

4.1 Metrics for performance evaluation

Three metrics are used to evaluate the performance of the algorithm described above:

– Detection Rate (DR)

$$DR = \frac{N_{GD}}{N_{GT}} \tag{12}$$

where N_{GD} is the number of good detections within the set of detected bounding boxes. N_{GT} is the number of ground-truth bounding boxes. A detected bounding box is considered as good detection if $\frac{A_{(GT \bigcap D)}}{A_{GT}} > 0.3$, where $A_{(GT \bigcap D_i)}$ is the overlapping area between the ground-truth bounding box and the detected bounding box associated to it.

– False Alarm rate (FA)

$$FA = \frac{N_{FA}}{N_D} \tag{13}$$

where N_{FA} refers to the number of false alarms within the set of detected bounding boxes. N_D is the number of bounding boxes detected. A bounding box is counted as false alarm if $\frac{A_{(GT \bigcap D)}}{A_{GT}} < 0.3$.

– Overlap precision measure (P)

$$P = \frac{1}{N_{GD}} \sum_{i=1}^{N_{GD}} \frac{A_{(GT \bigcap D_i)}}{\sqrt{A_{GT} A_{D_i}}} \tag{14}$$

This metric evaluates the overlap between the ground-truth and the correctly detected bounding boxes [10]. This measure not only favors the bounding boxes presenting a high overlap with the ground-truth bounding boxes, but also penalizes those that contain a lot of non-ground-truth pixels

4.2 Results

Ground-truth has been generated manually for a series of video sequences in order to evaluate the performance of the algorithm.
The metrics were calculated for three sets of results. The first set (A) corresponds to the detections performed on each frame, the second set (B) contains the results of the detection (with a five-frame period) combined with a forward tracking process, while the third set (C) represents the detection, forward and backward results merged by the proposed algorithm as shown in the previous sections. The results obtained on three video sequences are presented in Table1.

In the three cases, we notice that the Detection Rate (DR) increases when forward tracking is used. In fact, the face detector fails to determine the position

	Sequence 1			Sequence 2			Sequence 3		
	setA	setB	setC	setA	setB	setC	setA	setB	setC
Detection Rate(DR)	0.6923	1	1	0.7345	1	1	0.6281	0.9587	1
False Alarm (FA)	0	0	0	0	0	0	0.4685	0.5105	0.5105
Precision (P)	0.7911	0.7183	0.7595	0.7971	0.7985	0.8044	0.8262	0.8242	0.8515

Table 1. Performance results.

of some faces due to the pose or the poor illumination. The missed faces can be recovered by the forward tracking process. The Detection Rate (DR) also further increases when both forward and backward tracking have been used, since the face was not detected at the beginning of the shot but after several frames. For each of these frames, the trellis contained only one candidate resulting from backward tracking.
Once candidates were provided by the detector, forward and backward trackers, the trellis performed a selection that always improved the overlap precision (P), i.e. the face localization on the video frame.
We can also notice that one drawback of the tracking approach is that when a face is erroneously detected, then it is tracked on the whole shot thus increasing the False Alarm rate (FA), as can be seen in the case of the sequence 3.

5 Conclusion

In this paper, we proposed a forward/backward tracking process providing an accurate face localization in digital videos. It can also be applied for tracking any object for which we process an object detector. The described process combines detection, forward and backward tracking algorithms in order to extract possible faces. These candidates are used as nodes in a trellis diagram. The extraction of the optimal path from this trellis provided us the optimal choice of the facial bounding boxes. Our approach was mainly oriented towards face localization improvement and we noticed in fact that the precision rate was increased, while realizing a good detection rate. In our future work we will go further into exploiting the trellis structure and work towards decreasing the false alarm rate by merging distinctive trajectories.

6 Acknowledgement

The work presented was developed within NM2 (New Media for a New Millenium), a European Integrated Project (http://www.ist-nm2.org), funded under the European Commission IST FP6 programme.

References

1. R.C. Verma, C. Schmid, and K. Mikolajczyk, “Face detection and tracking in a video by propagating detection probabilities,” *IEEE Transactions PAMI*, vol. 25, no. 10, pp. 1215–1228, Oct. 2003.
2. Ji Tao and Yap-Peng Tan, “Accurate face localization in videos using effective information propagation,” in *Proc. of the IEEE International Conference on Image Processing (ICIP 2005)*, Genoa, September 2005.
3. M. Gong, “Motion estimation using dynamic programming with selective path search,” in *International Conference on Pattern Recognition*, Cambridge, United Kingdom, august 2004, vol. 4, pp. 203–206.
4. Changming Sun and Ben Appleton, “Multiple paths extraction in images using a constrained expanded trellis,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1923–1933, 2005.
5. Z. Liu and Y. Wang, “Face detection and tracking in video using dynamic programming,” in *Proc. of the IEEE Int. Conf. on Image Processing (ICIP 2000)*, 2000, pp. 53–56.
6. F. Pitié, S-A. Berrani, R. Dahyot, and A. Kokaram, “Off-line multiple object tracking using candidate selection and the viterbi algorithm,” in *Proc. of the IEEE Int. Conf. on Image Processing (ICIP 2005)*, Genoa, September 2005.
7. Paul Viola and Michael J. Jones, “Robust real-time face detection,” in *IEEE ICCV Workshop on Statistical and Computational Theories of Vision*, Vancouver, Canada, 2001.
8. N. Nikolaidis G. Stamou, M. Krinidis and I. Pitas, “A monocular system for automatic face detection and tracking,” in *Proc. of Visual Communications and Image Processing (VCIP 2005)*, Beijing, China, July 2005.
9. N. Nikolaidis G. Stamou and I. Pitas, “Object tracking based on morphological elastic graph matching,” in *Proc. of the IEEE Int. Conf. on Image Processing (ICIP 2005)*, Genova, Italy, September 2005.
10. S. Huovinen B. Martinkauppi, M. Soriano and M. Laaksonen, “Face video database,” in *Proc. First European Conference on Color in Graphics, Imaging and Vision (CGIV 2002)*, Poitiers, France, April 2002.

A Combination of Spatiotemporal ICA and Euclidean Features for Face Recognition

Jiajin Lei, Tim Lay, Chris Weiland and Chao Lu
Department of Computer and Information Sciences, Towson University
8000 York Road Towson, MD 21252, USA
Email: clu@towson.edu

Abstract

ICA decomposes a set of features into a basis whose components are statistically independent. It minimizes the statistical dependence between basis functions and searches for a linear transformation to express a set of features as a linear combination of statistically independent basis functions. Though ICA has found its application in face recognition, mostly spatial ICA was employed. Recently, we studied a joint spatial and temporal ICA method, and compared the performance of different ICA approaches by using our special face database collected by AcSys FRS Discovery system. In our study, we have found that spatiotemporal ICA apparently outperforms spatial ICA, and it can be much more robust with better performance than spatial ICA. These findings justify the promise of spatiotemporal ICA for face recognition. In this paper we report our progress and explore the possible combination of the Euclidean distance features and the ICA features to maximize the success rate of face recognition.

Keywords: Machine vision, Face recognition, Spatiotemporal ICA.

1. INTRUDUCTION

Face recognition is one of the most successful applications of image processing and analysis, and it has become one of the major topics in the research areas of machine vision and pattern recognition in the recent years. The applications can be seen in, but not limited to, the following areas: access control, advanced human-computer interaction, video surveillance, automatic indexing of images, video database and etc. In reality the process of face recognition is performed in two steps: (1) feature extraction and selection; and (2) classification of objects. These two steps are mutually related. Although the performance of classifier is crucial, a successful face recognition methodology may also depend heavily on the particular choice of features used by the classifier. So as far as face recognition is concerned, much effort has been put on how to extract and select the representative features [1]. Feature extraction and selection involve the derivation of salient features from the raw input data for classification and provide enhanced discriminatory power. Various kinds of methods have been proposed in the literatures [1]. Among them statistical techniques, such as principle component analysis (PCA), independent component analysis (ICA), have been widely used for face recognition. These techniques represent a face as a linear combination of low rank basis images. They employ feature vectors consisting of coefficients that are obtained by simply projecting facial images onto a set of basis images [2]. The practice proved that statistical method offers much more robustness and flexibility in terms of handling variations in image intensity and feature shapes. PCA uses eigenvectors with the largest eigenvalues to obtain a set of basis functions such that the original function can be represented by a linear combination of these basis functions [3]. The basis functions found by the PCA are uncorrelated, i.e. they cannot be linearly predicted from each other. However, higher order dependencies still exist in the PCA and, therefore, the basis functions are not properly separated [4]. ICA is a method that is sensitive to high-order relationship [5, 6]. By using ICA we can explore the important information hidden in high-order relationship among the basis functions. On the other hand Euclidean features are extracted from distances between certain important points on the face. This technique takes the advantage of the fact that different people have different face shape. But how to precisely locate the face organs is a big challenge.

Recently, Chen [7] proposed a spatiotemporal ICA algorithm to identify dynamic micro-Doppler motions. Continuing his work, we have applied spatiotemporal idea to face recognition. All experiments were performed on our special face database collected by AcSys FRS Discovery system. Two face datasets have been set up. One face dataset has less variation, while the other encompasses much more changes in terms of face expression and head side movements. In this study a comparison of performances among different face recognition approaches has been made. And we also explore the possible combination of the Euclidean distance features and the spatiotemporal ICA features to maximize the success rate of face recognition.

2. THE FACE ORGAN LOCALIZATION AND EUCLIDEAN FEATURE COMPUTATION

Feature based face recognition seeks to extract, from a face image, a set of numerical characteristics that can uniquely identify that face. Our proposed feature set is based upon the physical distances between common points of the face. In our study two features were chosen. The first feature was calculated by obtaining the distance between the centers of the eyes and the distance from the center of the left eye to the center of the mouth. These two values were then used to form a ratio in order to normalize for variance in the scaling of each image. The second feature was determined symmetrically with the center of the right eye. The third feature was extracted by making a ratio of the distance between two eyes and distance from the mouth to the middle point of two eyes.
To obtain the centers of the eyes and the mouth, a number of image processing methods were employed. In order to find the eyes, pattern matching was used to locally identify possible eyes. Specifically, the light to dark to light contrast of the pupils and eyelashes was looked for in the original grayscale image. All areas exuding this appearance were highlighted for closer scrutiny in a more global pattern-matching scheme after all potential eyes within a certain area were highlighted.

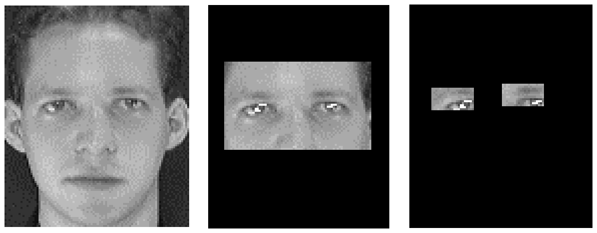

Figure 1. Identify eyes on a face

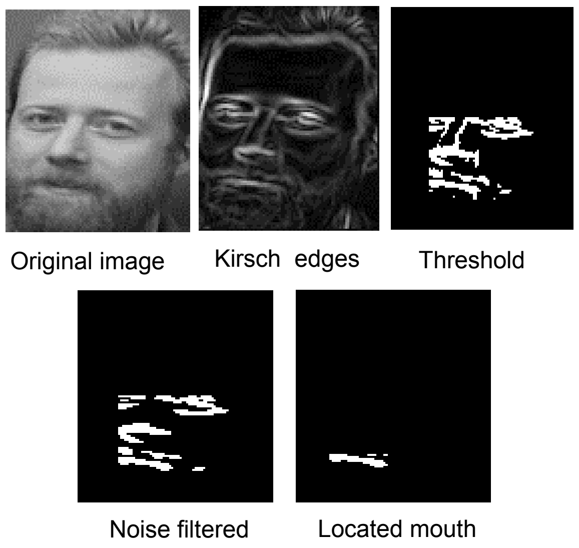

Figure 2. Finding the month on a face

After all areas of interest were highlighted, a simple symmetry detection scheme was implemented to identify possible pairs of eyes. The two pairs closest in size and shape to a predefined notion of an eye were chosen. More specifically, for each pixel, the number of highlighted pixels within a short distance of that pixel was stored at that pixel's location in a two dimensional array. Then two local maxima were searched for which were on approximately the same level. These two points were chosen as the approximate center of the eye, and everything not within a short distance of these areas was erased from the image. The procedures can be seen from Figure 1.

To find the mouth, a kirsch edge detection [8] filter was applied to the preprocessed grayscale image. The image had its edges removed and was then threshed based on the distribution of its histogram. The threshold value was set at the 80th percentile of the gray distribution. The mouth is one of the most contrasting features of the face, and thus with kirsch edge detection, it is featured more brightly than most parts of the face. The image was divided into small blocks in order to search for thin vertical lines and remove them as noise. The next step was to search through the binary image and obtain the characteristics of each cohesive group of remaining pixels. Objects were matched to a predefined notion of what a mouth could be, based on height verse width and area. The best matching group of pixels was taken as the mouth. The procedures of finding the month on a face are illustrated in Figure 2.

Obviously the above algorithms have their own problems and weaknesses. These revolve around alterations to the face and large variances in lighting. Dark framed glasses or glasses with any significant glare resulted in erroneous measurements. Asymmetrical facial expressions also resulted in off measurements, especially when the center of the mouth was shifted. Mustaches extending over the mouth also resulted in errors finding and reading the whole of the mouth. When part of the face was cast in a heavy shadow, unsatisfactory features were obtained. Because of these problems it is hard to use these features exclusively for classification. Combination with other features is necessary.

3. SPATIAL ICA AND SPATIOTEMPORAL ICA FOR FACE RECOGNITION

ICA is a statistical data processing technique to de-correlate the high order relationship of input. It was originally used for blind source separation (BSS). The basic ideal behind is to represent a set of random variables using basic functions, where the components (basic functions) are statistically as independent as possible. The observed random data (signal) $\mathbf{X}= (x_1, x_2, ..., x_m)^T$ can be linear combination of independent components (signals) $\mathbf{S} = (s_1, s_2, ..., s_n)^T$. We may express the model as

$$\mathbf{X} = \mathbf{AS}, \tag{1}$$

where **A** is an unknown constant matrix, called the mixing matrix. In feature extraction the columns of **A** represent features, and s_i is the coefficient of the *i*th feature in the data vector **X.** Several methods for estimation of this model have been proposed [9, 10]. Here we used fixed-point fast ICA algorithm for independent components (ICs) estimation [11].

3.1 Spatial ICA

If we concatenate a 2-D face image column-wisely, it can be represented as a 1-D signal (space-varying signal) as shown in Figure 3. Thus, a single face image becomes one entry in matrix **X** of (1). In face recognition, the first step is to find the ICs as well as **A** or its inverse **W** as in (2) from **X** by using an ICA algorithm. Each IC component can also be represented by an image. Figure 4 illustrates the procedure, and Figure 5 shows some samples of ICs.

$$I\widetilde{C} = W * X, \tag{2}$$

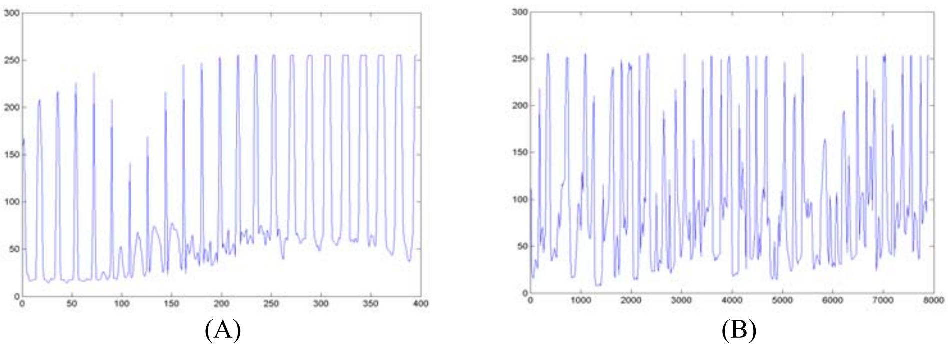

Figure 3. Face image signals created by concatenating rows of the image: (A) One face image; (B) A sequence of images

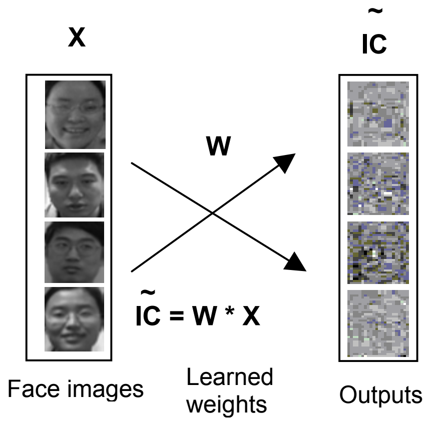

Figure 4. Estimate a set of ICs using ICA algorithm

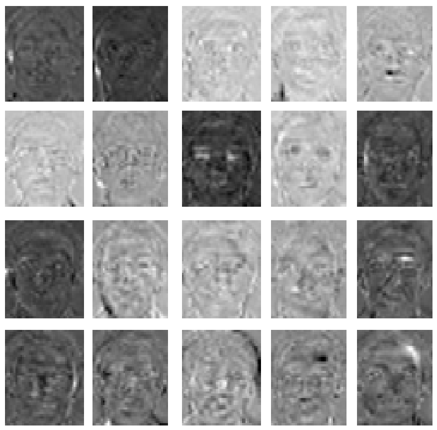

Figure 5. Some examples of spatial ICs

After ICs have been obtained, any observed new face image can be represented by linear combination of these ICs with a coefficient vector **A** as illustrated in Figure 6, and expressed in (3):

$$X_k(x, y) = \sum_{n=1}^{N} A_n IC_n(x, y), \tag{3}$$

where ($A_{1,}$ $A_{2,}$ ..., A_n)= **A**. The vector **A** is the desirable feature set of the observed image and will be used for classification.

3.2 Spatiotemporal ICA

Basically, spatiotemporal ICA shares the similar ideal with spatial ICA, but using an image sequence instead of a single image as operating unit. The face image sequence contains the features in both the space-domain and the time-domain. The goal of the spatiotemporal approach is to add time-domain feature into spatial 2-D feature set. So an entry in the **X** contains multiple images. A typical temporal image sequence is presented in Figure 7.

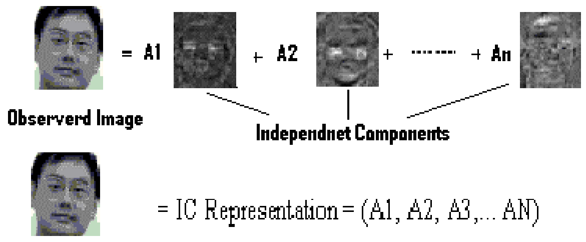

Figure 6. Representation of observed image with ICs

Figure 7. A sample of a face image sequence

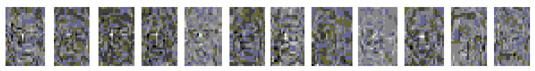

Figure 8. An examples of spatiotemporal ICs

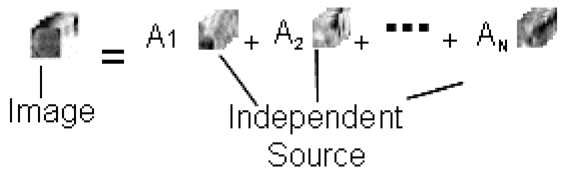

Figure 9. Representation of observed image sequence with spatiotemporal ICs

Similar to the spatial ICA, spatiotemporal ICs can be obtained by using joint spatial and temporal algorithm. That is the entries of **X** in (1) are image sequences. Spatiotemporal ICs are also sequences. Figure 8 illustrates an example of spatiotemporal ICs, where each sequence consists of 12 images.

Observed face image sequences can be represented by the spatiotemporal ICs as illustrated in Figure 9 and expressed with

$$X_k(x,y,t) = \sum_{n=1}^{N} A_n IC_n(x,y,t), \tag{4}$$

where $(A_1, A_2, ..., A_n)$= **A**. Notice in (4) that the time feature has been included, which means more information is added in this model with respect to spatial ICA.

4. LOCALIED ICA

With respect to PCA, ICA is spatially more localized [2, 6]. But it does not display perfectly the local characteristics and still uses the whole face information for operation if the input is with entire face images. Actually to recognize a person, ICA only bases on the important and valuable part of face information, such as eyes, mouth, and nose. If the whole face information is used, it may not add any more help. On the other hand, it may "dilute" the essential ones and makes performance deteriorated. So additional localization constraints should be imposed on ICA for better performance. For this purpose we take the advantage of the fact that eyes and mouth can be localized by the algorithm established in section 2. After positions of eyes and mouth have been found, a certain size of patches around eyes and mouth are respectively dug out. These two patches are concatenated together into a vector as the operation unit for matrix **X** of (1).

5. AcSys FRS DISCOVERY SYSTEM AND FACIAL DATABASE PREPARATION

Face database used in this work was produced by AcSys FRS Discovery System, which is powered by HNet technology and developed by AcSys Biometrics Corp., Canada. The System, which is not just a video camera, can track precisely the human face and store a sequence of face images in real time. The purpose of our study is to consider complicated situations, such as different face expressions, face side movements, and other variations (such as with glasses) in the image sequences. The AcSys FRS system can help us to achieve this goal, while other commercially available database cannot. Figure 10 shows the main display screen of the system. Using the functions provided by the system, we can customize and take the sequential face images for different purposes. In this study, two facial datasets have been collected, one with less variation (dataset 1), and the other one with more changes in terms of face expressions and head side movements (dataset 2). For each person 200 face images were sequentially recorded for each dataset. Every face image was manually cropped to 112-by-92 pixel size.

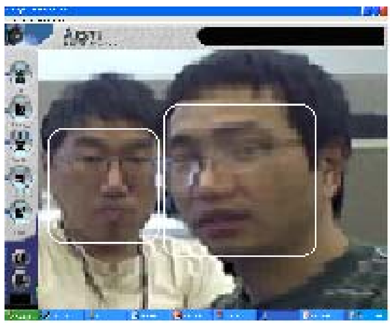

Figure 10. FRS main screen

6. THE EXPERIMENTS

Face recognition experiments respectively using spatial ICA, spatiotemporal ICA, localized ICA, and Euclidean feature were conducted. Instead of one single image used for input data unit as with spatial ICA and localized ICA, spatiotemporal ICA employs a sequence of images (12 images used in our experiments). As a result the dimension of image signal vector can become 12x112x92=123648, which is impracticable in terms of computational speed. To reduce the dimension we resized all the face images to 31-by-21 pixels. For each person 12 image sequences were produced in the following way. In the 200 image long sequence, we randomly choose a starting point, and took the following 12 images as an image sequence like given in Figure 7. For experiments of spatial ICA, localized ICA, and Euclidean features, 20 facial images were randomly picked from 200 images for each individual. In localized ICA, the patch sizes are 20-by-40 for eyes and 20-by-20 for month. For each experiment, we used half of the dataset (6 sequences for spatiotemporal ICA, 10 images for the others) for training and the remaining half for testing.

As mentioned earlier, in order to apply ICA algorithms to 2-D images, we concatenate rows of a 2-D image into a vector. The concatenated face image (space-varying) shares the same syntactic characteristics to regular time-varying signal (see Figure 3). This ensures that ICA can be applied to face image data [12]. After matrix **X** has been constructed with multi-image vectors, we also apply data normalization to eliminate the variation of images. Independent components (ICs) were estimated using training dataset. With the estimated ICs each observed new face image (sequence) can be represented by variant linear combination of ICs as building blocks. The variation is reflected in the amplitudes of coefficients of ICs (that is rows of matrix **A**), which can be found by

$$\boldsymbol{A} = \boldsymbol{X} * ICs^{-1} \quad , \tag{5}$$

where **X** is the new image (sequence) matrix (multi-images or image sequences). This matrix **A** contains representing features of the images (or image sequence). We used it as input data set for classification. For the purpose of performance evaluation, the numbers of ICs (features) from 2 to 200 with 10 as steps were respectively estimated. Classification was done on all of these numbers of features respectively. We calculated 3 Euclidean features for each image. Classification was conducted only once for this experiment.

We also explored performance of feature set combined from ICA features and Euclidean features. For this purpose we just appended Euclidean features to ICA feature space and repeated the above procedures. It must be noted all the experiments were conducted on both dataset1 and dataset2 parallelly.

In our experiments, linear Bayes normal classifier (LDC) and k-nearest neighbor (KNN) classifier [13] were used.

7. RESULTS AND DISCUSSION

The face recognition rates with respect to different numbers of features for different approaches are shown in Figure 11. The highest values are listed in the Table 1. The results show that spatiotemporal ICA outperforms any other approaches. This gracefully conforms to our expectation. In addition, all approaches perform better using dataset 1 than using dataset 2. This is not surprising since dataset 1 represents more stable condition. What worth noticing at this point are the disparities of performances between using different datasets within the same approach. Even though in Figure 11 (D) we see two performance curves apart in the middle part of the figure, they tend to converge at the end. Especially in Figure 11 (A) two curves get very close. For the other methods the two performance curves are consistently separated. This observation proofs that spatiotemporal ICA is less affected by variations of face expression and other factors. That is spatiotemporal ICA should be more robust than other methods. These findings justify the promise of spatiotemporal ICA for face recognition.

Table 1 The highest recognition rate for each experiment

	Classifier	**Spatial ICA**		**Spatiotemporal ICA**		**Localized ICA**		**Euclidean**	
		Dataset 1	Dataset 2	Dataset 1	Dataset 2	Dataset 1	Dataset 2	Dataset 1	Dataset 2
Highest Correct Rate	LDC	0.6823	0.6212	0.9724	0.9615	0.7616	0.7043	0.4016	0.3143
	K-NN	0.7002	0.6389	0.8954	0.7979	0.7530	0.7028	0.4021	0.3087

The recognition rate of spatial ICA itself is not good. But after the features were localized the performance was apparently improved (see Figure 11 (B), (C), (E) and (F)). So localization of face images before conducting ICA is a choice for improvement. It is worth for further investigation.

Though Euclidean features can be used in face recognition, it shows very poor performance with 40% recognition rate. In the hope that Euclidean distance features may give help for other approaches, we explored the combination of the Euclidean distance features with ICA features. Figure 12 displays the changes of recognition rates after Euclidean distance features have been added to ICA feature spaces. It seems that when the size of ICA feature space is small, Euclidean distance features put great weight for the performance improvement. But when the number of ICA features gets large the weight of Euclidean features in the total feature space dies away dramatically. This means Euclidean features only helps when the ICA performance is not good enough.

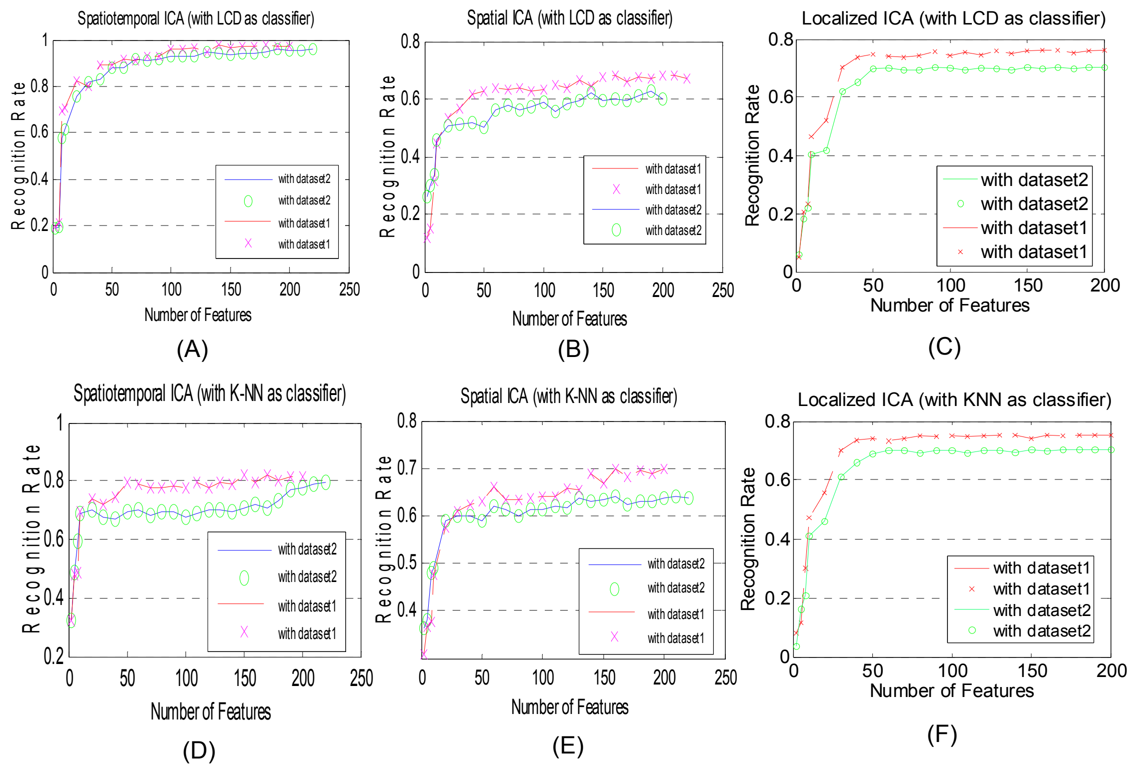

Figure 11. Face recognition rate against different feature numbers

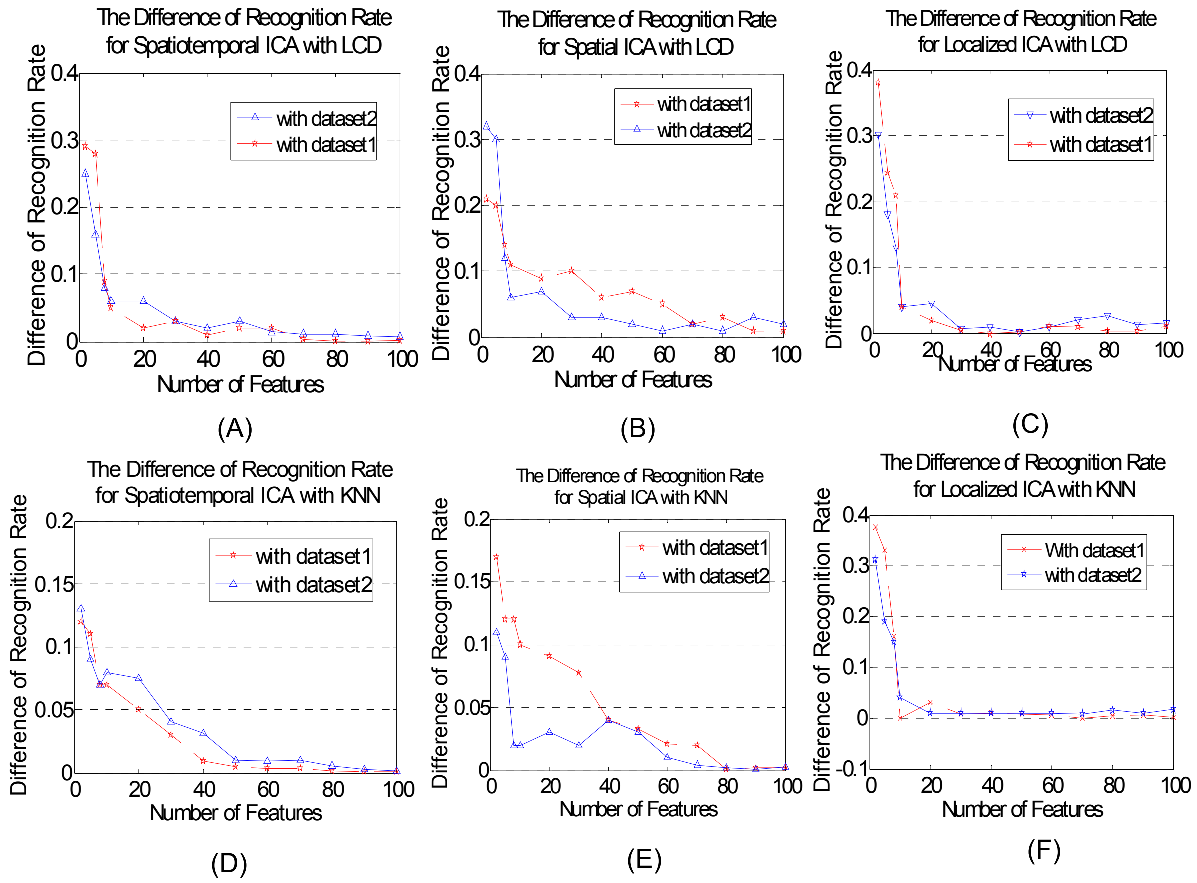

Figure 12. The Differences of recognition rates between before and after combining Euclidean features

Acknowledgements

We would like to thank Dr. Victor C. Chen of NRL, who has made many suggestions and provided support and advice.

References

[1] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld, Face Recognition: A Literature Survey, ACM Computing Surveys, Vol.35, No. 4, 399-458, December, 2003.

[2] Jongsun Kim, Jongmoo Choi, Juneho Yi, and Matthew Turk, Effective Representation Using ICA for Face Recognition Robust to Local Distortion and partial Occlusion, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 27, No. 12, 2005, 1977-1981.

[3] M. Turk, A. Pentland, Eigenfaces for Recognition, Journal of Cognitive Neuroscience 3(1), 71-86, 1991.

[4] R. Brunell and T. Poggio, Face Recognition: Features vs. Templates, IEEE Trans. Pattern Analysis and Machine Intelligence, 15(10):1042-1053, 1993.

[5] Chengjun Liu and Harry Wechsler, Comparative Assessment of Independent Component Analysis (ICA) for Face Recognition, In: the 2nd International Conference on Audio- and Video-Based Biometric Person Authentication, AVBPA'99, Washington D.C. USA, March 22-24,1999.

[6] M. Stewart Bartlett, J. R. Movellan, and T. J. Sejnowski, Face Recognition by Independent Component Analysis, IEEE Transactions on Neural Network, Vol.13, Nov., 1450-1464, 2002.

[7] Victor C. Chen, “Spatial and Temporal Independent Component Analysis of Micro-Doppler Features” In: 2005 IEEE International Radar Conference Record, 348 – 353, 9 – 12 May 2005, Arlington, VA, USA.

[8] Umbaugh, Scott E. Computer Imaging: Digital Image Analysis and Processing. New York, Taylor & Francis,2005.

[9] Bruce A. Draper, Kyungim Baek, Marian S. Bartlett, and J. Ross Beveridge, Recognizing Faces with PCA and ICA, http://www.face-rec.org/algorithms/Comparisons/draper_cviu.pdf

[10] Andreas Jung, An Introduction to a New Data Analysis Tool: Independent Component Analysis, http://andreas.welcomes-you.com/research/paper/Jung_Intro_ICA_2002.pdf.

[11] FastICA MATLAB package: http://www.cis.hut.fi/projects/ica/fastica

[12] James V. Stone, *Independent Component Analysis: A Tutorial Introduction*, Bradford Book, 2004.

[13] *R.P.W. Duin, P. Juszczak, P. Paclik,E. Pekalska, D. de Ridder, D.M.J. Tax, Prtools, http://www.prtools.org/.*