ACADEMIC PRESS

Available at www.ComputerScienceWeb.com POWERED BY SCIENCE DIRECT®

Computer Vision and Image Understanding 91 (2003) 22–52

Computer Vision and Image Understanding

www.elsevier.com/locate/cviu

Unsupervised recognition of multi-view face sequences based on pairwise clustering with attraction and repulsion

Bisser Raytchev* and Hiroshi Murase[1]

NTT Communication Science Laboratories, 3-1, Morinosato Wakamiya, Atsugi, Kanagawa 243-0198, Japan

Received 15 February 2002; accepted 11 February 2003

Abstract

In this paper we propose and investigate the possibilities inherent in a new, *unsupervised* approach to multi-view face recognition, which can be formulated mathematically as a problem of partitioning of proximity data, obtained from multi-view face image sequences. The proposed approach is implemented in two novel pairwise clustering algorithms, *CAR1* and *CAR2*, which partition the input data into identity clusters by performing combinatorial optimization guided by two types of interaction forces, attraction and repulsion, imposed on the original proximity matrices. Several experiments were conducted in order to test the performance of the proposed algorithms on real-world datasets including both frontal and side-view faces, which have been gathered over a period of several months. The obtained results can be considered encouraging for the general approach proposed here, and the new algorithms compared favorably to two other pairwise clustering algorithms, recently proposed in the image segmentation literature.
© 2003 Elsevier Inc. All rights reserved.

Keywords: Multi-view face recognition; Unsupervised learning; Pairwise clustering; Combinatorial optimization; Image sequences

* Corresponding author. Present address: Intelligent Systems Institute, National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba Central 2, Umezono 1-1-1, Tsukuba City, Ibaraki Prefecture 305-8568, Japan. Fax: +81-29-861-5930.

E-mail addresses: b.raytchev@aist.go.jp (B. Raytchev), murase@is.nagoya-u.ac.jp (H. Murase).

[1] Present address: Graduate School of Information Science, Nagoya University, Furo-Cho, Chikusa-Ku, Nagoya 464-8603, Japan.

1. Introduction

Being an area of both theoretical and practical interest, face recognition has attracted a lot of attention and research effort. However, in spite of the extensive research conducted in this area during the last several decades (see [1–4] for surveys), face recognition still remains a domain in which humans significantly outperform computers, especially in unconstrained, dynamically changing and unpredictable environments, for which still there are many unresolved problems. As new approaches for dealing with these problems are constantly being sought, here we will try to investigate one promising direction for research, which seems to have not received enough attention from the face recognition community. We consider the possibility that the ability of biological systems to interact directly with the sensory input from their environment and learn by self-organizing it into meaningful categories (*unsupervised learning*) could be a hint to one such promising direction. Perceptual self-organization, the ability to discover structural organization in the sensory data in unsupervised way, then should be important also in the design of artificial face recognition systems, which have to function efficiently and reliably in the ever-changing and unpredictable conditions of the real-world environments. Probably some of the reasons why the unsupervised approach has not received the attention it deserves can be attributed to the fact that it has been considered a "too difficult" problem, which is also computationally expensive. However, the recent increase in computational power and the development of better algorithms (one example being the recent development and successful application of more sophisticated graph-theoretic concepts), might cause a reevaluation of the role of unsupervised methods, leading to their broader application in many areas, like face recognition, in which traditionally the supervised methods have defined the dominating approach (see [5,6] for some previous attempts to use unsupervised methods).

Unsupervised face recognition can be considered important not only from theoretical point of view, but also because of the numerous potential practical applications which it can find like, for example, for online identification in video surveillance systems and man–machine interfaces, for content-based image retrieval/annotation in multimedia applications, among many others, when supervised strategies might be either impossible (category information is simply not available and category patterns have to be discovered, or "self-organized" from the input data stream) or impractical (when the manual segmentation/labeling of huge datasets into category groups can be overwhelming and costly).

Another difficult problem in face recognition, which already has started to attract more attention, is the problem of *multi-view* (or view-independent) face recognition—recognizing faces across different views. Achieving multi-view recognition in practical applications is difficult, because different people's faces observed in the same conditions (illumination, view angle, size and so on) look more similar to each other than the same person's face observed in different conditions—in frontal and side-view, under extreme illumination conditions, occluded, and so on. Different approaches have been proposed to solve this problem. While initially most of the research work was concentrated predominantly on *static* face images like for example, the modular eigenspaces of [7], elastic graph matching [8], or learning the correspondence transformations between views [9,10], just to mention a few representative approaches, more recently, the fact that the information contained in *dynamic* video sequences with continuously changing face views can be used advantageously, has started to attract more research effort [11–14].

Here we further expand the dynamic approach by considering an unsupervised alternative to the predominantly supervised algorithms for multi-view face recognition from video sequences (see also [15] for some other work on clustering sequences of tracked human figures). In the framework we propose, unsupervised multi-view face recognition is formulated mathematically as a problem of clustering of proximity data, obtained from multi-view face image sequences. The proposed approach is implemented in two novel pairwise clustering algorithms, *CAR1* and *CAR2*, which partition the input data into identity clusters by performing combinatorial optimization guided by two types of interaction forces, attraction and repulsion, imposed on the original proximity matrices. A detailed explanation of the motivation for proposing this approach is given in Section 1.1.

1.1. Motivation for the proposed general approach

The purpose of the learning algorithms introduced in this paper is to group a set of unlabelled face image sequences into their corresponding identity categories in unsupervised manner, without using any category information provided in advance. Although, as we already mentioned above, such formulation of the general problem of face recognition has not attracted much attention in the face recognition community, the basic underlying problem of "discovering structure hidden in unlabelled datasets" is considered important in research areas like image segmentation, perceptual grouping, content-based image retrieval, data mining and many others. In these areas, significant efforts have been aimed at developing algorithms for segmentation, grouping, self-organization, automatic inference (and so on, names varying among areas), which can be considered as instances of *clustering*, to use the standard pattern recognition term. Good overviews of clustering can be found in [16–18] and more details in [19,20]). A similar approach can be pursued also in face recognition, and we will try to show in this paper that certain clustering algorithms (especially *pairwise clustering*, defined below) can be successfully applied in unsupervised face recognition. We will also compare the clustering algorithms we propose here to some other algorithms which have been developed and successfully applied to similar problems in other areas, like image segmentation for example. It should be noted that although the algorithms proposed here have been used in this work specifically for face recognition, there is nothing domain-specific about them. Their application to other similar problems is straightforward.

Clustering methods try to partition the available data into clusters according to the "natural" categories present in the data, in the absence of explicit category information. Usually "natural" is assumed to mean that the patterns within the same cluster are more similar to each other than to patterns in different clusters. When a priori knowledge of the distribution of the observed data is not available, finding the proper balance between the two conflicting requirements—not to superimpose structure which does not exist in the data, and at the same time not to overlook existing structure, constitutes the major difficulty. Clustering methods can be further subdivided into *central* and *pairwise* clustering. In central clustering (for example see [21–24]) usually the data patterns can be represented as samples in a D-dimensional metric space, and it is assumed that each cluster can be parameterized by a center (also called centroid or prototype) around which the samples are spread according to some (usually assumed Gaussian) distribution. These assumptions are used to form a global criterion whose minimization determines the final partitioning. There are many practical cases, however, in which the representation of the input data with explicit coordinates in metric space is either not possible, or inconvenient to handle in such form, and instead the data are specified by their pairwise relations only, in the form of proximity (similarity or dissimilarity) values ordered in a proximity matrix. This is equivalent to representing each sample as a node in a graph in which the edges correspond to the proximity values. The task of pairwise clustering (see for example [25]) then is to partition the data based solely on the available pairwise relations, which can be formulated mathematically as a combinatorial optimization problem [26].

In our case, there are several peculiarities characterizing the data with which we deal, which motivate us to use pairwise rather than central clustering. As people move in unconstrained dynamic scenes, exposing different views of their faces to the camera (things being further complicated by factors like variation in scale and illumination, changes in facial expressions, and so on), the resultant face sequences for the different identity categories form complex non-linear manifolds in face image space. Although it is possible to treat each face as a separate sample represented by its coordinates in a certain metric face-space, centroids may be meaningless or difficult to define and handle there. Also, clustering which operates directly on individual face samples might be problematic, because for individual face samples, generally the within-class distance will be greater than the between-class distance, as illustrated in Fig. 1. If clustering were attempted with them, instead of grouping together different views of the same person's face (that is, grouping along the within-class, or view-variation axis in Fig. 1), most likely clusters of similar views of different people along the inter-class variation axis would be detected. Also, usually the number of the clusters (or identity categories) is not known in advance, while central clustering methods need it to be known a priori for the definition of their criterion functions, and the partitions they find are sensitive to this parameter. Finally, in the available datasets (especially if they are obtained under real-world surveillance conditions) the different categories might not be represented uniformly, some may be under-represented and some over-represented, and the same thing would be true for the face-views representation. Usually not all possible (or sufficiently many) views will be available, and typically the range of view change will vary widely among sequences. All these factors might render unjustifiable the modeling assumptions necessary for the central clustering approach, while pairwise clustering, being essentially a non-parametric, or model-free approach, needs fewer assumptions, thus being able to handle a larger variety of problems.

The rest of the paper is organized as follows. Section 2 briefly considers the problem of defining suitable measures of distance (or dissimilarity), both between

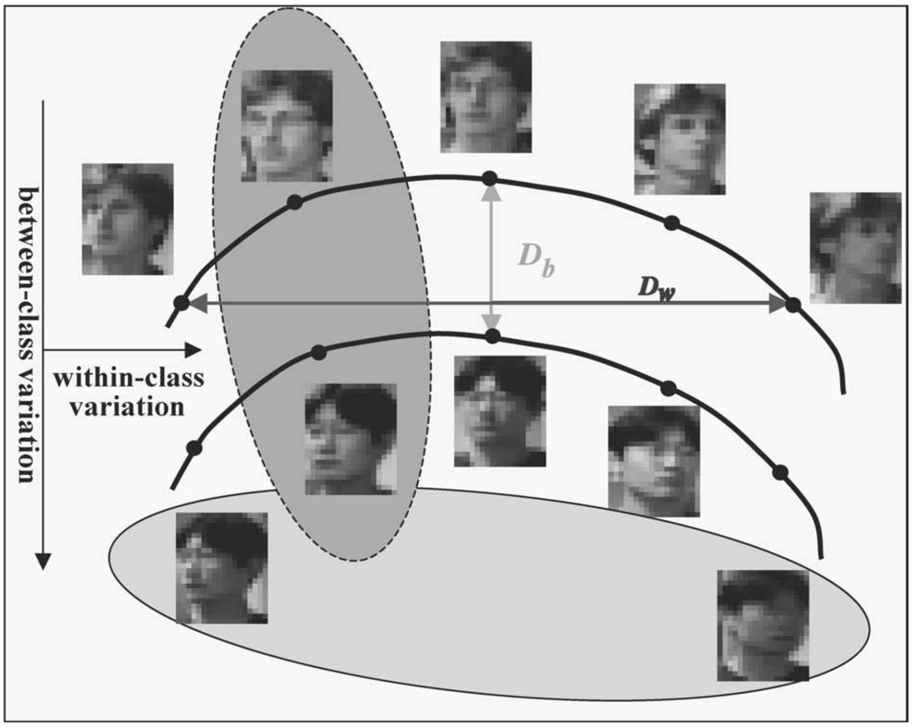

Fig. 1. For clusters of multi-view face sequences, generally the intra-class distance is greater than the inter-class distance.

face *images* and between face *sequences*, which will be needed to obtain the input pairwise proximity relations. Preference is being given to distance measures which are not computationally over-demanding, in view of the large volume of data which has to be processed when image sequences, rather than still images are used. Several algorithms, which seem to be able to implement the proposed general approach, will be described in Section 3. First, some definitions, which will be needed for the description of the clustering algorithms, will be given in Section 3.1. The proposed new algorithms *CAR1* and *CAR2* will be introduced in Sections 3.2 and 3.3, while two other recently proposed pairwise clustering algorithms (which also will be tried on our data for comparison) will be briefly reviewed in Sections 3.4 and 3.5. Experimental results will be reported in Section 4 and Section 5 will conclude the paper.

2. Distance measures between face images and between face sequences

As we mentioned above, we will assume that the input data we deal with are given in the form of a proximity matrix $\mathbf{P}_{N\times N} = \{p_{ij}\}$, in which p_{ij} is the distance between the ith and jth face sequences $S^{(i)}$ and $S^{(j)}$, and $i, j : 1, \ldots, N$. Details describing the preprocessing which extracts face-only image sequences from the input video stream will be given in Section 4, while here we will first consider how to define a suitable distance measure between two such face-only image sequences. Actually, it is necessary to define two distance measures: (a) distance (i.e., a measure of dissimilarity) between two face *images*; and (b) distance between two face image *sequences*. Different ways to do this are conceivable, but for simplicity and out of computational considerations, we will limit ourselves here (and in the experiments described in Section 4) only to two different instances of distance measures for (b) and one for (a). First, for the distances between two face images, represented as M-dimensional vectors $\mathbf{a}$ and $\mathbf{b}$, we will use the L_2 norm

$$L_2(\mathbf{a} - \mathbf{b}) \equiv \left(\sum_i^M |a_i - b_i|^2 \right)^{1/2}, \tag{1}$$

which calculates the Euclidean distance between two face patterns. Although we conducted experiments using also L_1 (the city-block metric) and the L_0^* norm (proposed in [27] as a better choice for face images than (1) above), defined as the number of pixel locations in $\mathbf{a}$ and $\mathbf{b}$ that differ in value more than a certain pre-defined constant δ:

$$L_0^*(\mathbf{a} - \mathbf{b}) \equiv \sum_{|a_i - b_i| > \delta} |a_i - b_i|^0, \tag{2}$$

we did not observe any important differences due to the particular metric used.

For the distance between two face sequences $S^{(i)}$ and $S^{(j)}$ we will define two measures: (a) the *minimal distance* $d_{\min}$ and (b) the modified Hausdorff distance d_{h}. The *minimal distance* is defined as

$$d_{\min}(S^{(i)}, S^{(j)}) = \min_{\mathbf{a} \in S^{(i)},\ \mathbf{b} \in S^{(j)}} \|\mathbf{a} - \mathbf{b}\|, \tag{3}$$

where $\| \cdot \|$ is the L_2 norm, i.e., $d_{\min}$ gives the distance between the nearest two faces among $S^{(i)}$ and $S^{(j)}$. The (directed) *Hausdorff distance* between two finite point sets A and B is defined as

$$H(A, B) = \max(h(A, B), h(B, A)),$$

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\|,$$

but as has been found for other image processing applications [28,29], a slightly modified version called a *modified Hausdorff distance* seems to be more suitable for practical tasks (for example, in our case a single face image $\mathbf{a}$ in $S^{(i)}$ that is far from any other face image in $S^{(j)}$ would cause $H(S^{(i)}, S^{(j)})$ to become very large), which we re-define here for the case of image sequences as

$$d_{\mathrm{h}}(S^{(i)}, S^{(j)}) = \frac{(f^{th}_{\mathbf{a} \in S^{(i)}} \min_{\mathbf{b} \in S^{(j)}} \|\mathbf{a} - \mathbf{b}\|) + (f^{th}_{\mathbf{b} \in S^{(j)}} \min_{a \in S^{(i)}} \|\mathbf{a} - \mathbf{b}\|)}{2}, \tag{4}$$

where $f_{\mathbf{x}\in S^{(x)}} g(x)$ denotes the fth quantile value of $g(x)$ over the set of face images in $S^{(x)}$, for some value of f between zero and one (e.g., $f = 0.5$ would give the *median* Hausdorff distance).

More elaborate *between-face* and *between-sequence* distance measures than those defined above might be used, if processing time is not a problem. Naturally, better distance measures will lead to better clustering results, but unfortunately more sophisticated measures tend to be very time consuming (especially when image sequences of hundreds of faces are involved, as in our case), so we have decided to limit our efforts here to showing that the proposed algorithm can work quite well even with sub-optimal distance measures (as those defined above), which obviously would generate some unreliable measures due to their limitations (see also [30] for an alternative way to organize the relations between face sequences by using a VQ-like approach).

3. Methods

3.1. Preliminaries

After the distance metrics are chosen and the proximity (dissimilarity) matrix $\mathbf{P}$ is calculated, the available data can be partitioned into clusters by pairwise clustering. Usually first $\mathbf{P}$ is modified into an *affinity* (or *similarity*) matrix $\mathbf{A}_{N\times N} = \{a_{ij}\}$ defined as

$$a_{ij} = \begin{cases} \exp\left(-\frac{p_{ij}^2}{2\sigma^2}\right) & \text{if } \exp\left(-\frac{p_{ij}^2}{2\sigma^2}\right) > r, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where σ is a free parameter reflecting some reasonable local scale and values less than a certain small constant r are set to zero. In our case, if the face sequences are thought of as nodes in a graph, interacting with each other, the ith row in $\mathbf{A}$ will describe the strength of interaction (or *affinity*) between the ith node (sequence) and all other nodes (sequences), which will always be positive or zero.

However, rather than zeroing all interactions between nodes further away from each other than a certain distance, we propose instead of (5) above to use the following transformation on $\mathbf{P}$ to form matrix $\mathbf{W}$, which permits both positive and negative interactions between nodes:

$$w_{ij} = \begin{cases} \exp\left(-\frac{p_{ij}^2}{2\sigma^2}\right) - \exp\left(-\frac{(p_{ij}-2\gamma)^2}{2\sigma^2}\right) & \text{if } p_{ij} < 2\gamma, \\ \exp\left(-\frac{4\gamma^2}{2\sigma^2}\right) - 1 & \text{if } p_{ij} \geqslant 2\gamma. \end{cases} \tag{6}$$

In (6), γ is a parameter whose meaning and the way to set it will become clear below. The form of the transformation (6) is compared to the one in (5) in Fig. 2, and several examples of matrices $\mathbf{P}$ and their corresponding matrices $\mathbf{A}$ and $\mathbf{W}$ (obtained for data used in the experiments described in Section 4) can be seen in Figs. 3–5.

In this way we define two types of interactions between our nodes (the face sequences): nodes for which w_{ij} is positive are said to *attract* each other with strength

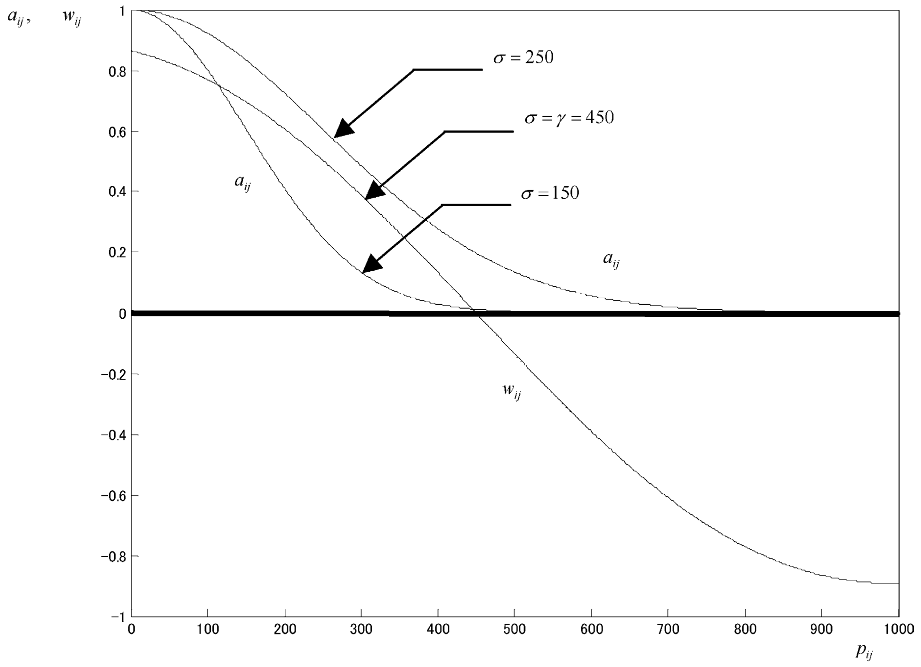

Fig. 2. Comparison of the general forms of the transformations in (5) and (6).

proportional to w_{ij}, and nodes for which w_{ij} is negative are said to *repel* each other with strength proportional to $|w_{ij}|$. For a certain cluster **C** we can define the resultant strength of interaction $F(i|\mathbf{C})$ for each node $i \in \mathbf{C}$ as $F(i|\mathbf{C}) = \sum_{j\in\mathbf{C},j\neq i} w_{ij}$ and use it to define a measure for the overall structural stability of a cluster **C** as

$$Z(\mathbf{C}) = \sum_{k\in\mathbf{C}} F(k|\mathbf{C}) = \sum_{i\in\mathbf{C}} \sum_{j\in\mathbf{C}} w_{ij}. \tag{7}$$

The more positive (7) is, the more stable the structure of **C** is considered to be. For the purposes of the clustering algorithm introduced in Section 3.2, a given set of nodes **C** is recognized as a valid cluster only if

$$F(k|\mathbf{C}) \geqslant 0 \quad \text{for all } k \in \mathbf{C}, \tag{8}$$

i.e., when each of the nodes k receives as a whole more attraction than repulsion from the rest of the members of **C** (note that $\mathrm{Z}(\mathbf{C}) \geqslant 0$ is not a sufficient condition for **C** to be a valid cluster).

A set of points **C** which does not satisfy (8) can be modified into a valid cluster by *splitting* (Fig. 6), i.e., removing from it those points l^* for which $F(l^*|\mathbf{C}) < 0$, to obtain a new cluster $\mathbf{C}^* = \mathbf{C} - \{l^*\}$. The order in which points are being removed from **C** is important. Assume that the distribution of interaction forces in a certain cluster **C** is given by the solid curve in Fig. 6a. Then the point removal starts with the point $l^* = \arg\min_{l\in\mathbf{C}}\{F(l|\mathbf{C})\}$ (the node whose corresponding resultant interaction force is

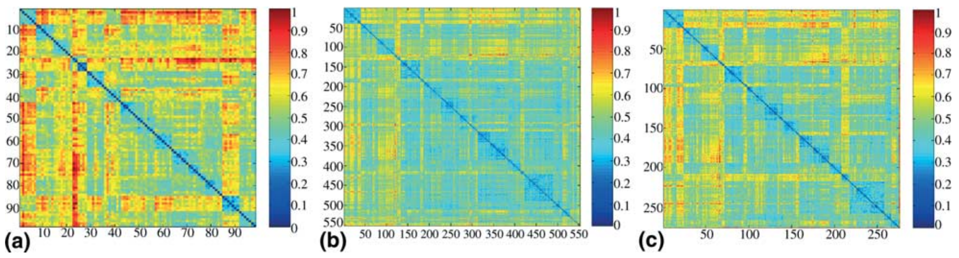

Fig. 3. The original proximity (dissimilarity) matrices **P** obtained when the *minimal distance* between face sequences is used as a metric. In **P**, each entry p_{ij} represents the distance between the ith and jth face sequences. For visualization purposes only, in **P** the face sequences/nodes have been ordered into their corresponding correct groups, that is nodes belonging to the same cluster have been put next to each other. Of course, such ordering information is not available a priori (neither used by the grouping algorithm), but has to be found by the clustering itself: (a) data for experiment I (98 sequences, both frontal and multi-view faces); (b) data for experiment II (552 sequences, both frontal and multi-view faces); (c) data for experiment III (275 sequences, frontal faces). Details about the experiments are given in Section 4.

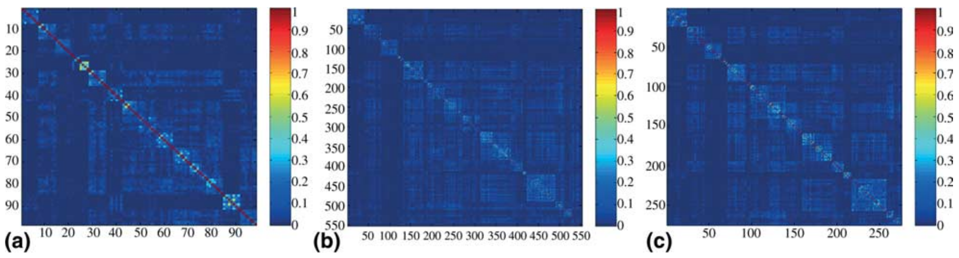

Fig. 4. The affinity (similarity) matrices **A** obtained from their corresponding proximity matrices **P** by using the transformation (5). The examples for **A** given in (a)–(c) correspond to the examples of **P** in Figs. 3a–c.

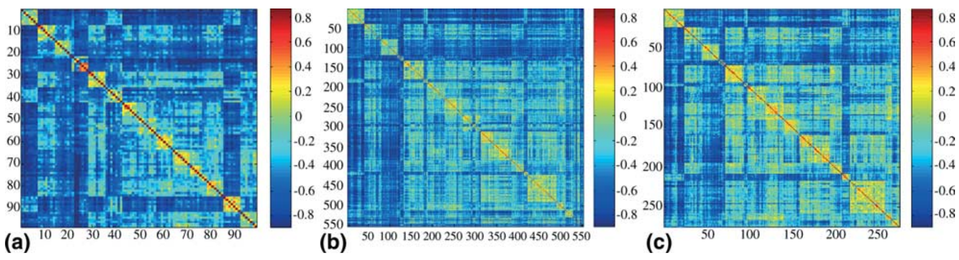

Fig. 5. The matrices **W** obtained from their corresponding proximity matrices **P** by using the attraction/repulsion transformation (6). The examples for **W** given in (a)–(c) correspond to the examples of **P** in Figs. 3a–c.

shown leftmost in Fig. 6a) and after it is removed, the resultant interaction forces $F^*(k|\mathbf{C}^*)$ for all $k \in \mathbf{C}^*$ are recalculated, leading to a new distribution, enveloped by the dotted curve in Fig. 6a, and the change in resultant interaction forces for each

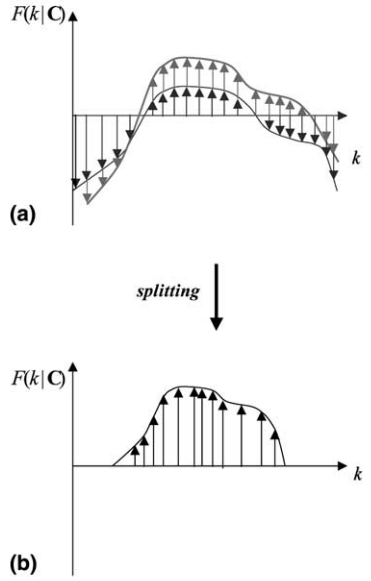

Fig. 6. Illustration of the splitting mechanism. In (a), the distribution of the resultant interaction forces for each sequence k in cluster $\mathbf{C}$ *before* and *after* the splitting of a single sequence (the leftmost one) is shown, respectively, under the *solid* and *dotted* curves. The distribution of the interaction forces for the sequences left in $\mathbf{C}$ after the splitting process has reached a state of balance between attraction and repulsion (i.e., after the cluster validity criterion is satisfied) is shown in (b).

node being displayed by dotted arrows. After the removal, nodes (sequences) which have been near the removed node, i.e., had received large attraction from it, will become more negative, while nodes which have been far from the removed node, i.e., had been strongly repelled by it, will become more positive. After this splitting is done, again the node with the most negative resultant interaction (if such one exists) is removed, the forces acting on the remaining nodes are recalculated, and this procedure is repeated until either no more nodes with negative resultant forces exist (as in Fig. 6b), or $\mathbf{C}^*$ is a singleton, i.e., until the cluster validity criterion (8) is satisfied.

3.2. Clustering by attraction and repulsion (CAR1)

The actual clustering algorithm will be explained here (pseudocode is given in Fig. 7), making use of the definitions introduced in the previous subsection. We assume that initially given are N unlabelled face sequences from L categories, and the objective is to group them into clusters without using any category-specific information provided in advance (L, the number of different people is also unknown). Essentially, our algorithm repeats the following two *merge* and *split* steps:

Step 1. Merging. Assume that each of the available face sequences $S^{(a)}$, $S^{(b)}, \dots, S^{(k)}, \dots$ can be represented by nodes $a, b, \dots, k, \dots$, each node initially forming a separate set, so that we have $K = N$ singletons $\mathbf{C_a} = \{a\}$, $\mathbf{C_b} = \{b\}, \dots, \mathbf{C_k} =$

```
ALGORITHM CAR1

INITIALIZE $\gamma = 0$; all sequences form separate clusters $\mathbf{C}_i = \{i\}$ for $i$: $1\ldots N$;
WHILE $\gamma < \gamma^*$
    $\gamma := \gamma + \Delta\gamma$;
    REPEAT
        merge step
            merge any cluster $\mathbf{C}_\mathbf{k}$ with such cluster $\mathbf{C}_\mathbf{l}$ (if existing) for which
                $l = \arg\max_{\mathbf{i}}\{Z(\mathbf{C}_\mathbf{k} \cup \mathbf{C}_\mathbf{i})\}$,
                where $Z(\mathbf{C}_\mathbf{x}) = \sum_{i\in\mathbf{C}_\mathbf{x}}\sum_{j\in\mathbf{C}_\mathbf{x}} w_{ij}$
                and $\mathbf{C}_\mathbf{l}$ must satisfy (1)-(3) below:
                    (1) $\sum_{i,j\in\mathbf{C}_\mathbf{k}\cup\mathbf{C}_\mathbf{l}} w_{ij} \geq 0$;
                    (2) $\exists w_{ij} > 0,\ i \in \mathbf{C}_\mathbf{k},\ j \in \mathbf{C}_\mathbf{l}$;
                    (3) $|\mathbf{C}_\mathbf{k}| \leq |\mathbf{C}_\mathbf{l}|$;
        split step
            (assume the number of clusters formed during the previous merge step was $K$)
            FOR each $\mathbf{C}_\mathbf{k}$ (k: $1..K$)
                IF $Z_\mathbf{k} = \sum_{i,j\in\mathbf{C}_\mathbf{k}} w_{ij} < 0$
                    REPEAT
                        remove sequence $l^* = \arg\max_{l\in\mathbf{C}_\mathbf{k}}\{\sum_{i,j\in\mathbf{C}_\mathbf{k}-\{l\}} w_{ij}\}$ from $\mathbf{C}_\mathbf{k}$
                        to form a new singleton $\mathbf{C}_{K+1} = \{l^*\}$
                        $K := K+1$;
                    UNTIL $Z_\mathbf{k} \geq 0$
    UNTIL no changes in the obtained clusters
```

Fig. 7. Pseudocode for the *CAR1* algorithm. How to determine γ^* is explained in the text.

$\{k\}, \ldots \mathbf{C}_{K=N} = \{N\}$. For all k $(k : 1 \ldots K)$, merge set $\mathbf{C_k}$ with set $\mathbf{C_l}$ for which $l = \arg\max_\mathbf{i}\{Z(\mathbf{C_k} \cup \mathbf{C_i})\}$ and which also must satisfy the following conditions (a)–(c):

$$\text{(a)} \qquad \sum_{i,j\in\mathbf{C_k}\cup\mathbf{C_l}} w_{ij} \geqslant 0, \qquad (9)$$

$$\text{(b)} \qquad \exists w_{ij} > 0, \quad i \in \mathbf{C_k},\ j \in \mathbf{C_l}, \tag{10}$$

$$\text{(c)} \qquad |\mathbf{C_k}| \leqslant |\mathbf{C_l}|, \tag{11}$$

so that $\mathbf{C_k}$ is not changed if no $\mathbf{C_l}$ satisfying (9)–(11) exists. Condition (10) is necessary to guarantee that there exists a force of attraction between the two sets candidates for a merge (this is not necessarily satisfied if (9) is satisfied).

Step 2. Splitting. If K sets are obtained after the merging step, for each set $\mathbf{C_k}(k : 1, \ldots, K)$ check whether it forms a valid cluster satisfying (8), that is whether each member of that cluster totally receives more attraction than repulsion from the other members of the same cluster. The sequence-member l^* with lowest value of $F(l^*|\mathbf{C_k}) < 0$ (if such l^* exists), is removed from $\mathbf{C_k}$ to form a new singleton $\mathbf{C}_{K+1}$, which will participate in the following merge step (as a candidate for a merge). After the removal of the most negative member, the resultant interaction forces for the remaining sequences are updated, again the most negative member (if existing) is removed to form the singleton $\mathbf{C}_{K+2}$ and the above procedure is repeated until either no more negative members remain in $\mathbf{C_k}$ or it is a singleton.

The merging and splitting steps above are repeated until their execution does not lead to the formation of any new clusters or to changes in the already existing ones. Each repetition of a merge step followed by a split step forms one *cycle*. Several cycles may be necessary before the partitioning reaches a steady state for a certain value of the parameter γ, which determines the point where attraction turns into repulsion (see Fig. 2). Our algorithm starts with a very low value of γ (i.e., attraction exists only between nodes the distance between which is very short), and each time a steady state is reached for the partitions, γ is increased by a step $\Delta\gamma$ (which can be set depending on the available computational resources, see Fig. 8 for an example of a sequence of transforms $w_{ij}(\gamma) = f(p_{ij})$), and everything repeated again until γ reaches a certain value γ^* and the partition obtained for the steady state at this value is reported as the final result. This γ^* is the only parameter which our algorithm needs to be determined, apart from σ in (6), which we can conveniently set to $\sigma = \gamma$, as changing σ produces only some change in the slope of the repulsion/attraction curves in Fig. 8, but does not influence significantly the general form of the transformation which is most important (but note that σ is important for algorithms which use (5) instead of (6), since for them it determines which entries in $\mathbf{A}$ will be zeroed!). Since γ^* depends on the metric being used, one way to deal with it is to find experimentally a value which produces good results for a certain metric and a sufficiently large dataset, and fix γ^* to that value, in the expectation that it will work also when new data is added to the same dataset. However, it would be much better if there is a way to determine γ^* automatically for each different instance of a metric or dataset, without making any additional assumptions. In the following lines we will propose one such method to determine γ^* automatically in a more objective and data-driven way, which seemed to work fine for all our experiments, and was used to obtain the results reported in Section 4.

We define the *normalized global partition energy* obtained for a certain value of γ as

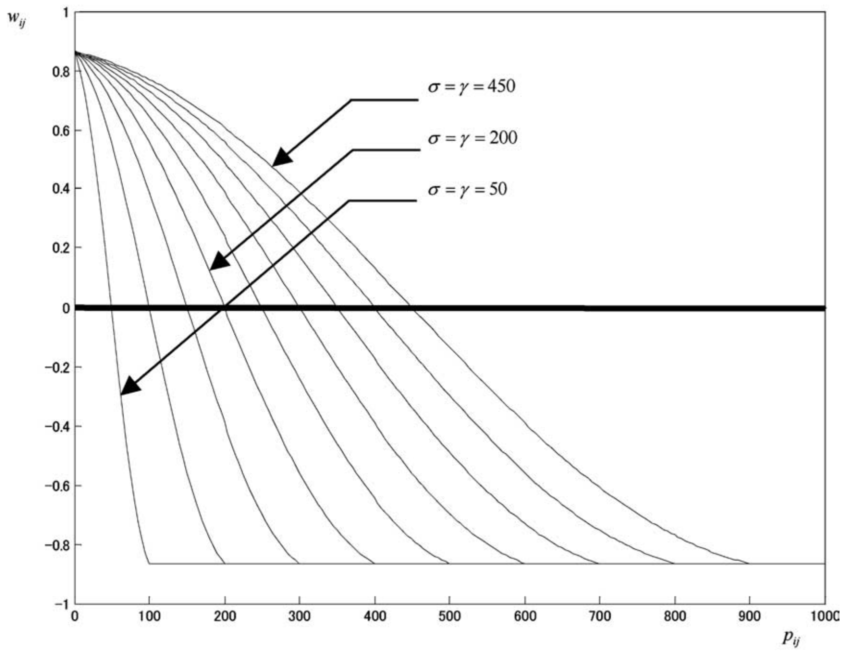

Fig. 8. A sequence of the attraction/repulsion transform (6) obtained in successive steps of $\Delta\gamma = 50$ for the parameter $\gamma\sigma$ was set to be equal to γ, although the form of the transform does not change significantly for a wide range of values for σ).

$$E_{\mathrm{n}}(\gamma) = \sum_{k=1}^{K} \sum_{i \in \mathbf{C_k}} \sum_{j \in \mathbf{C_k},\, j \neq i} \frac{w_{ij}(\gamma)}{\Psi^{+}(\gamma)}, \tag{12}$$

where K is the number of clusters (partitions) obtained by the algorithm for a certain value of γ, and

$$\Psi^{+}(\gamma) = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij}(\gamma), \tag{13}$$

$$\Psi^{-}(\gamma) = \sum_{i=1}^{N} \sum_{j=1}^{N} r_{ij}(\gamma), \tag{14}$$

$$a_{ij}(\gamma) = \begin{cases} w_{ij}(\gamma), & w_{ij}(\gamma) \geqslant 0, \\ 0, & w_{ij}(\gamma) < 0, \end{cases} \tag{15}$$

$$r_{ij}(\gamma) = \begin{cases} w_{ij}(\gamma), & w_{ij}(\gamma) < 0, \\ 0, & w_{ij}(\gamma) \geqslant 0. \end{cases} \tag{16}$$

When γ is small, the *total* attraction $\Psi^{+}(\gamma)$ will be small, the total repulsion $\Psi^{-}(\gamma)$ will be large, and the number of clusters K also will be large (although most of them will be singletons)—the resulting partition will be an oversegmentation. As γ increases, $\Psi^{+}(\gamma)$ will grow larger, while $\Psi^{-}(\gamma)$ and K will decrease. At some point, $\Psi^{+}(\gamma)$ will become larger than $\Psi^{-}(\gamma)$ and eventually K will become 1, that is all nodes will be squeezed into one enormous cluster. In the range between $K = N$ and $K = 1$ there will be some value of γ (the sought γ^{*}) for which there will be neither an oversegmentation nor an undersegmentation of the data, and the resulting partition will represent the structural groups in the best possible way, at least according to the selected metric and the available dataset.

If we plot the graph of $E_{\mathrm{n}}(\gamma)$ for datasets of different size and content and for different metrics (see Figs. 9a–c for some datasets used in our experiments here), we will notice an interesting pattern in its behavior. In all cases, there is a certain value of γ before which $E_{\mathrm{n}}(\gamma)$ increases with quite a constant slope (while $K + S$, as shown in Figs. 10a–c, decreases in a similar manner; where S is the number of singletons and K the number of clusters with more than one node inside them), and after which it enters a plateau phase (or the slope changes abruptly), before starting a steep and steady growth again. We will select the value of γ obtained at the beginning of the plateau phase as the sought γ^{*} (an arrow in Figs. 9a–c shows which value has been used for the experiments), and we will try to explain the behavior of $E_{\mathrm{n}}(\gamma)$ and the reasoning behind our choice with the following arguments. As γ increases, $\Psi^{+}(\gamma)$ also increases, i.e., there is more attraction available to more nodes to interact between themselves, leading to the formation of fewer and larger clusters, and the balance between attraction an repulsion is such that the new partitions are characterized by a steadily increasing energetic level, quantified by $E_{\mathrm{n}}(\gamma)$. At some point, however, even though new portions of attraction forces are supplied as before, the balance between attraction and repulsion for the new structures does not lead to a proportionally steep increase in $E_{\mathrm{n}}(\gamma)$ compared to the previous partitions. This situation, which most likely indicates the beginning of a process of unreasonable "squeezing" of clusters together, will generally continue for some time, during which $E_{\mathrm{n}}(\gamma)$ plateaus, until finally the system will make a new phase transition (or change of scale in terms of the global partition), characterized by a new period of steady growth of $E_{\mathrm{n}}(\gamma)$. Stopping the algorithm in the beginning of the first plateau phase produced the best results for all our experiments in the sense that the trade off between *error of misclassification* (or recognition rate, quantified by expression (24) in Section 4) and *oversegmentation* (generally quantified by the number of clusters obtained) was optimal for the concrete dataset and metric. (Note that oversegmentation, characterized by too many clusters, leads to very low error of misclassification, but produces partitionings which are practically useless. This problem will be addressed again in more detail in Section 4.3.)

To summarize, we hold that using repulsion (rather than discarding the negative values in **W** as unnecessary information) together with attraction to guide the grouping process permits much fuller exploitation of the structural information hidden in the relation values of the proximity matrices, which is especially important in the case when real-world data with lots of noise have to be processed. As can be seen

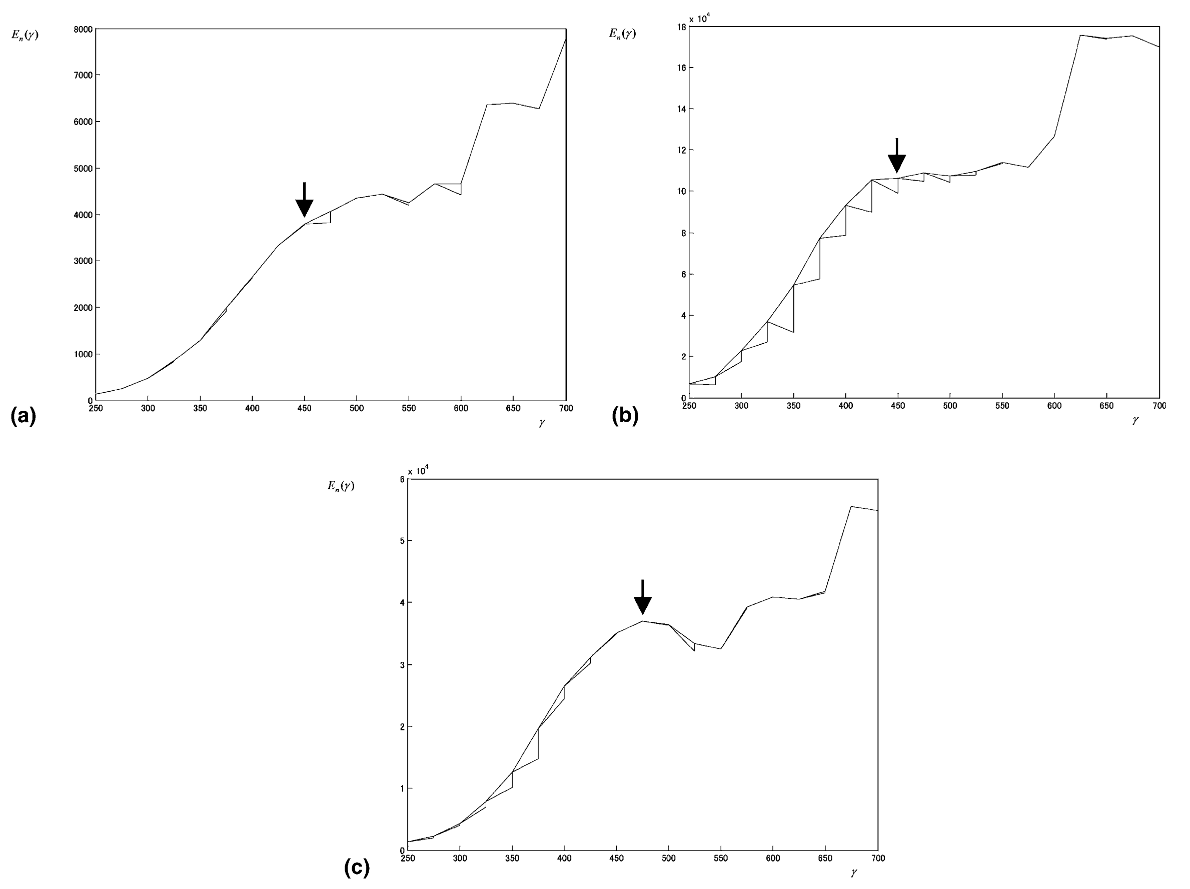

Fig. 9. The normalized global partition energy $E_{\rm n}$ for a succession of partitions obtained by changing the parameter γ in small steps $\Delta\gamma$. The figures (a)–(c) show results for $E_{\rm n}$ obtained when the minimal distance metric was used between the face sequences, respectively, in experiments (I)–(III). The arrows show the value of $E_{\rm n}$ for which the corresponding γ^* was determined.

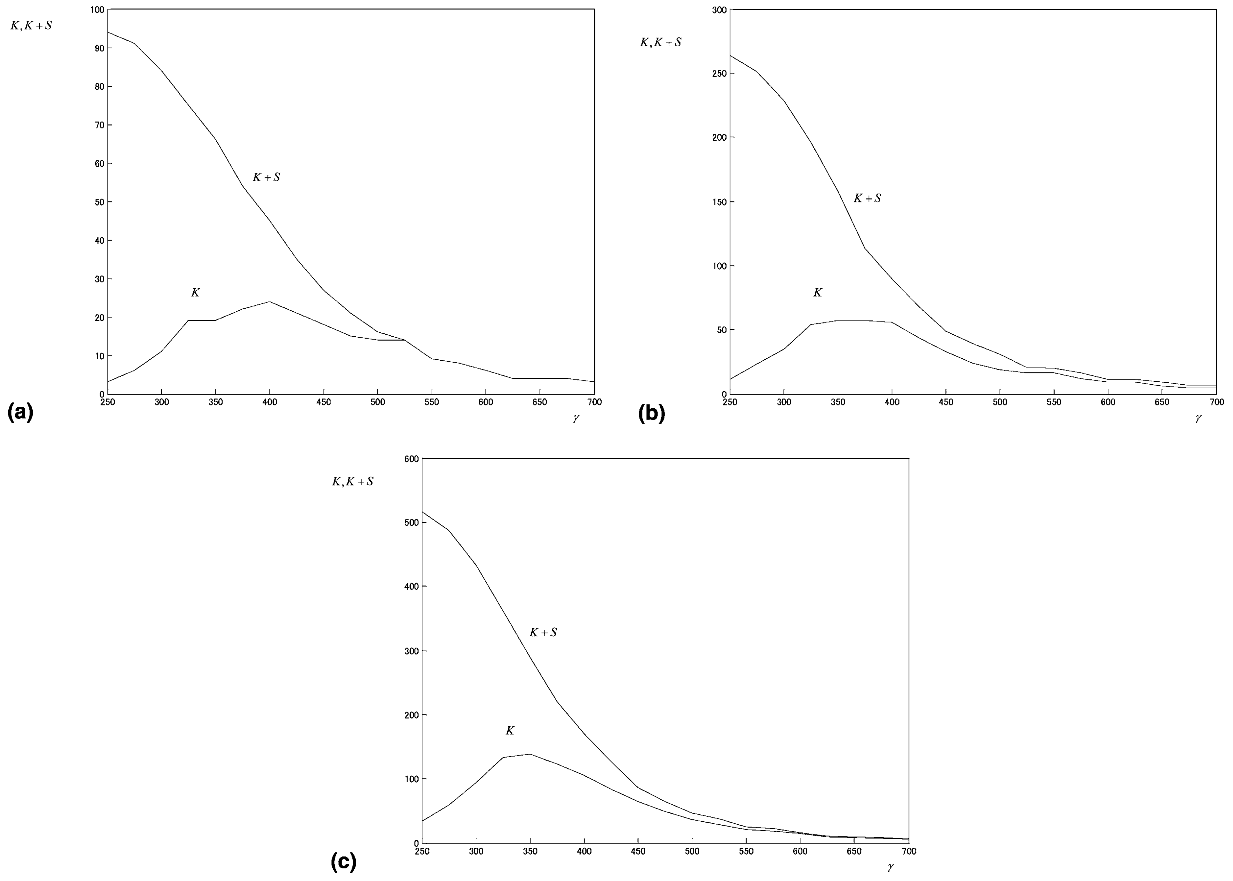

Fig. 10. Plots showing how K and $K+S$ change for a succession of partitions obtained by changing the parameter γ in small steps $\Delta\gamma$. K is the number of clusters having more than one node inside them, while S is the number of singleton (outlier) clusters. The figures (a)–(c) correspond to Figs. 9a–c.

from the data in Figs. 3–5, the diagonal blocks, that is the clusters to be found, contain a lot of noise (caused, especially in our case, by factors like the non-uniform ranges of face-view change among sequences, problems due to illumination changes, imperfect preprocessing and so on), they are not clearly and unambiguously separated from each other and the off-diagonal blocks (remember also that in Figs. 3–5 the nodes have already been ordered into correct groups, while such information is not available a priori, this has to be found by the clustering algorithm!), but the nuances and gradation of the "noise" can be utilized both to distinguish between structures which otherwise would be considered homogeneous (leading to undersegmentation) or to find similarities between structures which otherwise might be taken apart (leading to oversegmentation).

3.3. Clustering by attraction and repulsion—global optimization (CAR2)

The *CAR1* algorithm introduced in the previous subsection optimizes a *local* criterion (the structural cluster stability in (8) subject to conditions (9)–(11)) and we would like to know how this would compare with an algorithm based on the same underlying strategy (the balance between attraction and repulsion) but optimizing a *global* criterion (even if it is too slow for a real-time performance). For this purpose we propose the following algorithm, *CAR2*, outlined below.

We define a symmetric Boolean matrix $\mathbf{X} = \{x_{ij}\}$ of size $N \times N$, called an *indicator matrix*, for which $x_{ij} = 1$ if nodes (i, j) belong to the same partition (cluster), and $x_{ij} = 0$ otherwise (note that x_{ii} should always be 1). The ith row of $\mathbf{X}$ is the indicator vector $\mathbf{x}_i^t$ $(i = 1, \dots, N)$, whose non-zero entries' indexes show which other nodes are grouped in the same cluster as node i. According to our definition, a permutation of $\mathbf{X}$ in which all nodes belonging to the same cluster are ordered next to each other would consist of diagonal blocks of all '1's, and off-diagonal blocks of all '0's. The aim of the algorithm then will be to determine $\mathbf{X}$ by optimizing a certain global criterion which is a suitable function of the matrices $\mathbf{X}$ (the sought partition) and $\mathbf{W}$ (the input data on which the attraction/repulsion transformation (6) has been applied). We consider the minimization of the following two criterion (cost) functions:

$$C_1(\mathbf{X}) = \min\left\{ -\sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij}w_{ij} \right\}, \tag{17}$$

$$\begin{aligned} C_2(\mathbf{X}) &= \min\left\{ -\sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij}w_{ij} + \sum_{i=1}^{N}\sum_{j=1}^{N}(1 - x_{ij})w_{ij} \right\} \\ &= \min\left\{ \sum_{i=1}^{N}\sum_{j=1}^{N}(1 - 2x_{ij})w_{ij} \right\}. \end{aligned} \tag{18}$$

The criterion function in (17) maximizes the within-cluster attraction (in the presence of repulsion), while the criterion function in (18) maximizes the within-cluster attraction and the between-cluster repulsion at the same time. In order to solve the combinatorial optimization problems (17) or (18) we use simulated annealing

[31], a stochastic optimization strategy in which state space is stochastically sampled by a Markov process, and new solutions are accepted or rejected according to the Metropolis algorithm [32] with the following transition probabilities:

$$P(\mathbf{X}^{\text{old}} \to \mathbf{X}^{\text{new}}) = \begin{cases} 1 & \text{if } \Delta C(\mathbf{X}) \leqslant 0, \\ \exp(-\Delta C(\mathbf{X})/T) & \text{otherwise,} \end{cases} \tag{19}$$

where $\Delta C(\mathbf{X}) \equiv C(\mathbf{X}^{\text{new}}) - C(\mathbf{X}^{\text{old}})$. State changes corresponding to decreases in the cost function are always accepted, while in addition, to avoid local minima, state changes corresponding to an increased cost are accepted with an exponentially weighted probability, determined by the temperature parameter T. The temperature is gradually reduced during the stochastic search process according to a predetermined *cooling schedule* [33].

Another important detail is to determine the way in which the configuration $\mathbf{X}^{\text{old}}$ has to be perturbed in order to obtain the new configuration $\mathbf{X}^{\text{new}}$, or the definition of a "move" between two configurations. Here we would like a move between two configurations to implement our idea of merges/splits of nodes to/from clusters. For this purpose, we initially set $\mathbf{X}$ to be equal to the unit matrix $\mathbf{I}$, that is the N available nodes form N singleton clusters, and each move randomly selects some x_{ij} $(i \neq j)$ whose binary value is flipped as

$$x_{ij}^{\text{new}} := \begin{cases} 1 & \text{if } x_{ij}^{\text{old}} = 0, \\ 0 & \text{if } x_{ij}^{\text{old}} = 1, \end{cases} \tag{20}$$

which can be interpreted in the following way.

(1) If x_{ij} is set to '1', this means that the jth node will be added (merged) to the cluster which has node i among its members, after being removed (split) from its current cluster. In addition to that, the following update of $\mathbf{X}$ is necessary:

$$x_{kj}^{\text{new}} := 1 \quad \text{for } \forall k \text{ for which } x_{ik}^{\text{old}} = 1, \tag{21}$$

$$x_{kj}^{\text{new}} := 0 \quad \text{for } \forall k \text{ for which } x_{jk}^{\text{old}} = 1 \text{ (and } k \neq j). \tag{22}$$

The updates (21) and (22) performed on $\mathbf{X}$ complete the split of the jth node from its previous cluster and its merge to the new cluster.

(2) If x_{ij} is set to '0', this means that the jth node will be removed (split) from its current cluster (which has also node i among its members). Also the following update of $\mathbf{X}$

$$x_{kj}^{\text{new}} := 0 \quad \text{for } \forall k \text{ for which } x_{ik}^{\text{old}} = 1 \text{ (and } k \neq j) \tag{23}$$

is necessary in order to complete the split of the jth node from its previous cluster to form a singleton (which can be merged again to another cluster at some future move). If the cooling schedule (the schedule for lowering T) is chosen to be

$$T = T_{t=0}/\ln(1+t) \tag{24}$$

the algorithm *CAR2* introduced above is guaranteed (according to the "annealing theorem" proved in [34]) to converge to the globally optimal partition of the dataset for the given metric, in terms of the global criterions (17) or (18). However, as such a slow cooling schedule is impractical, we have used instead $T(t+1) = cT(t)$ with $0.8 < c < 0.99$ as recommended in [18] for practical problems (c was fixed to 0.9 for the experiments reported in Section 4).

As with *CAR1*, *CAR2* does not need to know the number of identity categories/clusters K in advance (although it can be easily modified to find exactly K clusters, if such information is available, by using an $N \times K$ matrix $\mathbf{X}$, and slightly modifying expressions (20)–(23)), and the grouping process is guided by both attraction and repulsion. Its major drawback is that it becomes too slow for more than a few hundred sequences to be useful for real-time applications. However, as it finds a globally optimal solution to our combinatorial optimization problem, it can be used as a benchmark against which to compare the performance of other candidate algorithms. Results obtained by using *CAR2* with our datasets will be reported in Section 4.

3.4. The normalized cut algorithm (NCut)

In this and the following subsection we will briefly review two other recently proposed pairwise clustering algorithms, which have been applied successfully to various data partitioning problems, and as such can be considered as alternative methods to try to solve our problem here. Like our algorithms above, both of them are able to find the natural partitions in a dataset without the need to know the number of clusters K in advance, although they cannot handle negative values in the affinity matrix, so in terms of the definitions given in Section 3.1 they permit only positive interactions (attraction) between the nodes. Results obtained by using those two algorithms with our datasets will also be reported in Section 4 (readers familiar with the algorithms reviewed in this and the following subsection might go straight to Section 4).

The normalized cut (*NCut*) algorithm [35] is representative of the general technique called *spectral clustering* [36]. Spectral methods [37–41] identify good partitions based on the eigenvectors of the affinity matrix $\mathbf{A}$ (defined in (5)), or other matrices derived from it. Unfortunately, at this time there seems to be no general agreement among authors regarding the question which eigenvectors to be used and how to obtain the clusters. In particular, *NCut* considers the generalized eigenvectors $\mathbf{y}_i$ as a solution to

$$(\mathbf{D} - \mathbf{A})\mathbf{y}_i = \lambda_i \mathbf{D}\mathbf{y}_i, \tag{25}$$

where λ_i is the ith generalized eigenvalue and the diagonal matrix $\mathbf{D}$ is the degree matrix of $\mathbf{A}$ defined as

$$d_{ii} = \sum_{j=1}^{N} a_{ij}. \tag{26}$$

Thus, (26) represents the total interaction (attraction) between node i and all other nodes. *NCut* partitions the data in a reverse direction to the merge/split strategy of *CAR1* and *CAR2*: initially all nodes are considered to form one big cluster ($K = 1$), which is split into two ($K = 2$) by thresholding the generalized eigenvector corresponding to the second smallest λ_i, and similarly each of the two resultant partitions (clusters) can be recursively split into two, until some stopping criterion is met. The second generalized eigenvector provides a continuous approximation to a discrete optimization problem (to minimize the so-called "normalized cut," defined in (28)–(30)), which requires to minimize the Rayleigh quotient

$$\frac{\mathbf{y}^t(\mathbf{D} - \mathbf{A})\mathbf{y}}{\mathbf{y}^t\mathbf{D}\mathbf{y}} \tag{27}$$

subject to the constraints $\mathbf{y}_i \in \{1, -b\}$ and $\mathbf{y}^t\mathbf{D}\mathbf{1} = 0$ ($\mathbf{1}$ being a vector of all '1's). The normalized cut *NCut A*, *B*) is defined as

$$NCut(A,B) = \frac{cut(A,B)}{asso(A,V)} + \frac{cut(A,B)}{asso(B,V)}, \tag{28}$$

$$cut(A,B) = \sum_{i\in A}\sum_{j\in B} a_{ij}, \tag{29}$$

$$asso(A,V) = \sum_{i\in A}\sum_{j\in V} a_{ij} \tag{30}$$

for a weighted undirected graph $\mathbf{G}(\mathbf{V},\mathbf{E})$, which can be partitioned into two disjoint sets $A, B, A \cup B = \mathbf{V}$. (In $\mathbf{G}$, the weights of the edges $\mathbf{E}$ connecting the N nodes $\mathbf{V}$ are given by the affinity matrix $\mathbf{A}$). In other words, the discrete optimization problem above tries to find a bipartition that minimizes the affinity (or attraction) *between* the two clusters as a fraction of the total attraction available to each cluster.

The *NCut* algorithm has the following problems. First, since the constraint that the generalized eigenvector should take only two discrete values is ignored in order to be able to obtain the *continuous* approximation to the original *discrete* optimization problem, it may not be always obvious how to obtain the bipartition, and therefore some thresholding has to be introduced to determine the "splitting point" of the values of the eigenvector (as demonstrated in Fig. 11). Another heuristic has to be used to decide when to stop the recursive splitting process. These problems make this (and most other spectral methods) difficult to use, limiting the possibilities for their use in a completely automated system, at least until they are not formalized in a more satisfying way. In order to obtain the experimental results for our datasets, we tried several different heuristics for the thresholds. As these failed to produce consistently good results, we inspected visually the eigenvector entries and set the thresholds manually for each bipartition.

3.5. The typical cut algorithm

The typical cut (*TCut*) algorithm [42] is a recently proposed stochastic pairwise clustering algorithm, representative of a class of clustering algorithms inspired by statistical mechanics [21,25], which apply stochastic simulations of certain dynamics to partition the input data. This particular algorithm is related to the super paramagnetic

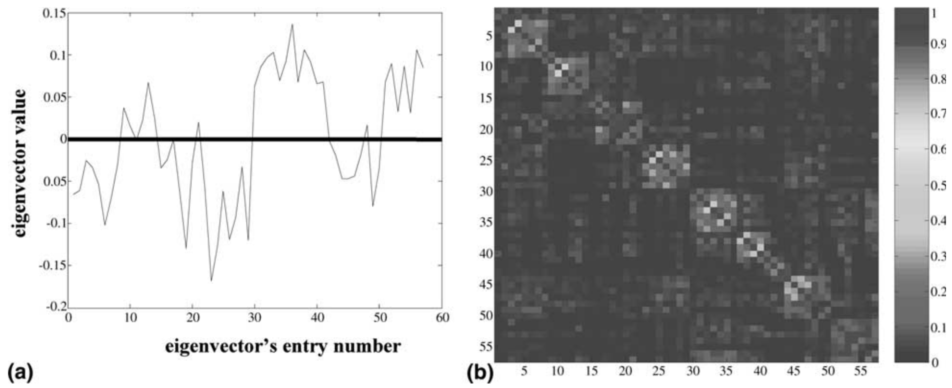

Fig. 11. In (a) is shown the generalized eigenvector corresponding to the second smallest eigenvalue of Eq. (25) in the *NCut* algorithm, obtained for the graph described by the affinity matrix given in (b). As the entries in the eigenvector take continuous values, a suitable threshold has to be chosen to obtain a bipartition. Some heuristic has to be employed in order to set a threshold, but it is difficult to obtain good results in this way for all subsequent bipartitions in the recursively divisive process.

clustering (SPC) method, based on the granular magnet model of [43]. *TCut* uses the contraction algorithm proposed in [44] to generate samples of cuts in the graph to be partitioned (described by the affinity matrix), from which the "average" or "typical" cut is computed. An *r-way* cut is a cut which partitions the graph $\mathbf{G}(\mathbf{V},\mathbf{E})$ into r disjoint sets $(V_1, V_2, \ldots, V_r)$, and its *capacity* is defined as $\sum_{\alpha\neq\beta, i\in V_\alpha, j\in V_\beta} a_{ij}$. For a given value of r ($r = 1 \ldots N$) the contraction algorithm generates a sample of M possible *r-way* cuts (by merging nodes with probability proportional to the affinity between them) and uses it to estimate the probability p^r_{ij} that the edge (i,j) between nodes i and j is not a crossing edge of a random *r-way* cut. Then, for every integer r between 1 and N, the typical cut $(C_1, C_2, \ldots, C_{s(r)})$ is defined as the partition of $\mathbf{G}$ into connected components, such that $p^r_{ij} < 0.5$ for every $i \in C_\alpha$, $j \in C_\beta$ $(\alpha \neq \beta, \alpha, \beta = 1, \ldots, s(r))$. The typical cut for each r is found by removing all edges for which $p^r_{ij} < 0.5$ and computing the connected components in the remaining graph. All N resultant typical cuts correspond to different clustering solutions to the same problem, and in order to provide a measure how "meaningful" or "interesting" a certain solution is, the following function is defined

$$T(r) = \frac{2}{N(N-1)} \sum_{i>j} N_i N_j, \tag{31}$$

where $N_k = |C_k|$ is the cardinality of the kth cluster. Only partitions which are associated with large change in $T(r)$ for subsequent values of r are considered, that is only those for which $\Delta T(r) > \delta$ (for some threshold δ).

To summarize, *TCut* does not seek the global optimum of a certain cost function, but is interested in "averaging" the partitions with weights proportional to their quality (measured by the capacity of the cuts). Like the other algorithms described above, *TCut* is able to find the natural partitions in a dataset without the need to be given the number of clusters K in advance. Also, unlike *NCut* (and most hierarchical methods), it computes the whole hierarchy of partitions at once, without the necessity to apply recursive divisions into two parts, thus possessing a natural way to leave outliers unclassified (which in the *CAR1* and *CAR2* algorithms is achieved by the singleton-splitting mechanism: all singletons in the final partition can be thought of as unclassified nodes-outliers, forming a single "cluster of outliers"). The main problem we met with the *TCut* method is that all solutions obtained for $\Delta T(r) > \delta$ have to be inspected, since there is no guarantee that the best solution will correspond to the maximum of $\Delta T(r)$, and also it is not clear how to determine the threshold δ in an objective and automatic way. In the experimental results reported for this algorithm, we inspected the candidate solutions corresponding to the top 20 values of $\Delta T(r)$ and reported the *best* result found among them (which of course could not be done in case we did not know the correct answers).

4. Experimental results

In order to evaluate and compare the performance of the above methods, several experiments were conducted using a dataset of about 600 face image sequences obtained over a period of several months from 33 different subjects. A typical example of the experimental setting can be seen in Fig. 12, and several time-subsampled face sequences for different people, extracted from the raw camera input by the preprocessor described in Section 4.1 can be seen in Fig. 13. Illumination conditions were very demanding and varied significantly with the time of the day during which the samples were taken. The video sequences' length varied between 30 and 300 frames, depending on the speed at which the subjects walked in front of the camera, in the range between slow walking with occasional stops, and running. For each one of 17 of the subjects were gathered between 10 and 50 sequences (about one-third of which were taken when the subjects did not know that they were being monitored, that is they moved in their natural way), while less than three (typically only 1) sequences were available for the remaining 16 subjects (these were called "rare visitors" and they did not know that they were being monitored by the camera while they walked).

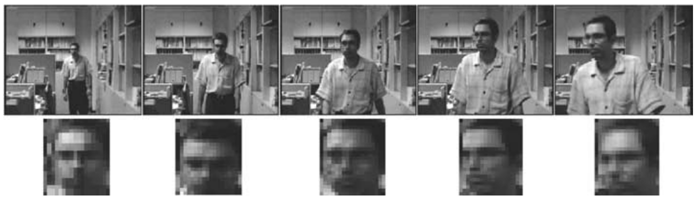
Fig. 12. An example of an original face image sequence (temporally subsampled) together with the corresponding normalized face-only sequence extracted from it.

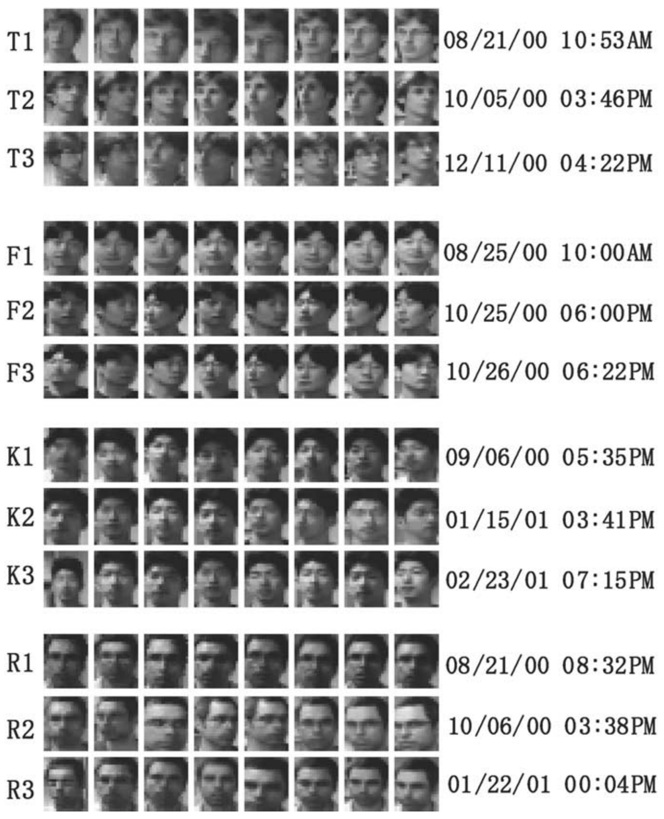

Fig. 13. An example of several time subsampled face sequences with category labels (to be obtained by the algorithm), shown to the left and the time stamp labels available from the preprocessor, shown to the right.

4.1. Preprocessing

Since the concrete implementation of the preprocessor is not essential for the operation of the learning algorithms, only a brief overview will be provided here (for more details see our previous work in [45]). All that is required from the preprocessing is to obtain image sequences of the moving objects of interest and to guarantee that each separate image sequence corresponds to one and the same object only (for example by tracking). Here we assume that input is provided from a video camera fixed in a constant position and continuously monitoring the scene in front of it. The subjects enter the scene, walk towards the camera, and finally exit the scene. To extract the face-only image sequences, a multi-resolution image pyramids are formed from the binary silhouettes of the moving subjects and the face area is extracted after analyzing the x- and y-histograms of the binary silhouettes at different resolutions (alternative algorithms for face tracking/extraction may be employed, depending on the concrete task; see [46,47] for examples).

The extracted and size-normalized (by subsampling) face-only image sequences are used to calculate the proximity matrix **P** using the metrics explained in Section 2. The matrices **A** and **W** needed for the experiments were calculated using respectively (5) and (6).

4.2. Datasets

We prepared two different datasets to be used in the following experiments:

(a) *Dataset A*. In this dataset, the subjects were just walking forward toward the camera. As a result, this dataset contained predominantly frontal faces, with only a few side-view faces included at the end of the sequences, when the subjects passed beside the camera. Sequences T1, F1, K1, K3, R1, R3 in Fig. 13 are representative for the data included in this set.

(b) *Dataset B*. In this dataset, the subjects were told to look to the left and right, up and down, as they moved towards the camera. Both frontal and side-view faces were represented. Sequences T2, T3, F2, F3, K2, R2 in Fig. 13 are representative for the data included in this set.

Samples with and without glasses were included for all subjects (except for the "rare visitors") and hairstyles changed with time. Resolution of the original images was 320×240 pixels, and 18×22 pixels for the size-normalized face-only images. Because of the large volume of data, we were unable to manually inspect all the face sequences output from the preprocessing module, but from the few inspected ones it was obvious that the dataset on which the system had to perform contained many instances of noisy data, in the form of erroneous face croppings and misalignments, large variations in illumination with face shadows, and so on, as would be expected in a real-world situation.

4.3. Evaluation of the clustering

We propose the following formula to calculate the self-organization (recognition) rate ρ of the final clusters:

$$\rho = \left(1.0 - \frac{E_{AB} + E_O}{N}\right) \times 100\%, \tag{32}$$

where N is the total number of sequences to be grouped, E_{AB} is the number of sequences which are mistakenly grouped into cluster for certain category A, although in reality they come from category B, and E_O is the number of samples gathered in clusters in which no single category occupies more than 50% of the nodes inside them. While the meaning of E_{AB} above is obvious, analysis of many different partitions obtained for different datasets by different algorithms revealed the following interpretation for E_O. A small E_O (in comparison to E_{AB}) usually signals the relatively harmless presence of some small clusters of outliers, which have happened to be very near to each other, while a large value of E_O most probably is a sign of bad partitioning (undersegmentation), and most likely occurs when the clustering algorithm has been unable to discriminate between the members belonging to several different identity categories, effectively mixing them together in some huge cluster(s).

It should be noted, however, that although the self-organization (recognition) rate proposed above can evaluate the error of misclassification contained in the final partitioning precisely, it provides only partial information about the structural quality of the obtained partitioning. That is, it can detect (from the value of E_O above) when a certain clustering leads to an undersegmentation of the data, but (especially in the absence of a suitable cluster validation criterion, like the normalized global partition energy E_n for the *CAR1* algorithm, or $\Delta T(r)$ for the *TCut*) it can be fooled by a clustering leading to oversegmentation, which might produce very high ρ, although the resulting partitioning might be practically useless (as the data is split into too many clusters). When the true partitioning is known (as in the case of the experiments reported below), a more objective judgement about the partitioning quality can be obtained by the combined information provided by (a) the number of the *true* partitions (that is the number of identity categories); (b) the number of the *obtained* partitions K; (c) the number of detected *singleton-outliers* S; and (d) the recognition rate ρ, above. All these have been provided in the experimental results summarized in Tables 1–4 (see also Fig. 10).

Table 1
Recognition results for the CAR1 algorithm

Experiment (sequences)	Distance used	Identity clusters	Clusters found	Singletons found	γ	E_{AB}	E_O	ρ (%)
I(98)	M	14	18	9	450	3	5	91.8
I(98)	H	14	20	7	515	6	0	93.9
II(552)	M	33	64	22	450	19	30	91.1
II(552)	H	33	59	29	525	30	20	90.9
III(275)	M	17	24	15	475	11	3	94.9
III(275)	H	17	23	15	550	18	5	91.6

Results when the between-sequence distance was the minimal distance (M) and the modified Hausdorff distance (H) are shown.

Table 2
Recognition results for the CAR2 algorithm

Experiment (sequences)	Distance used	Identity clusters	Clusters found	Singletons found	γ	E_{AB}	E_O	ρ (%)
I(98)	M	14	19	14	430	2	0	98.0
I(98)	H	14	20	8	515	2	4	93.9
II(552)	M	33	65	25	450	12	44	90.0
II(552)	H	33	61	31	525	25	99	77.5
III(275)	M	17	22	18	475	13	17	89.1
III(275)	H	17	23	15	550	21	16	86.2

Results when the between-sequence distance was the minimal distance (M) and the modified Hausdorff distance (H) are shown.

Table 3
Recognition results for the *NCut* algorithm

Experiment (sequences)	Identity clusters	Clusters found	Singletons found	σ	E_{AB}	E_O	ρ (%)
I(98)	14	18	8	200	1	4	94.9
II(552)	33	54	26	200	38	53	83.5
III(275)	17	34	15	200	21	0	92.4

Results when the between-sequence distance was the minimal distance are shown.

Table 4
Recognition results for the *TCut* algorithm

Experiment (sequences)	Distance used	Identity clusters	Clusters found	Singletons found	σ	E_{AB}	E_O	ρ (%)
I(98)	M	14	15	12	180	9	0	90.8
I(98)	H	14	14	23	220	10	0	89.8
II(552)	M	33	27	195	150	0	225	59.2
II(552)	H	33	25	187	200	0	211	61.8
III(275)	M	17	8	49	180	0	101	63.3
III(275)	H	17	17	50	200	15	67	70.2

Results when the between-sequence distance was the minimal distance (M) and the modified Hausdorff distance (H) are shown.

4.4. Experiments (unsupervised learning)

The following three experiments were conducted.

Experiment I. This experiment used a dataset containing face sequences randomly selected from both datasets A and B, that is both frontal and multi-view sequences were included. The samples included a total of 98 sequences, 7 sequences being available for each of 14 different subjects (identity categories). The purpose of this experiment is to test the algorithms on a dataset which is (a) easy to be visualized, as the samples are distributed evenly among the categories; (b) relatively clean, as the small size of the datasets permitted the results of the face extractor to be inspected and "cleaned" from obviously wrong face croppings; (c) relatively small, which might render this experiment relatively easier (although definitely not trivial), but useful, in conjunction with the other two experiments below, to provide an information about how the performance of the algorithms changes with change in the size of the input dataset. Additionally, there are many practical cases when the algorithms have to perform on small datasets, possibly because of insufficient data.

Experiment II. This experiment used all available data, that is all data in sets A and B put together. Both frontal and side-view faces were represented in this relatively large dataset, which included 552 face sequences from 33 subjects.

Experiment III. Only data from dataset A (frontal or near-frontal faces only) were used in this experiment. The purpose was to isolate the "multi-view" factor and report results obtained for unsupervised *frontal* face recognition. The dataset included 275 face sequences from 17 subjects.

Each of the three experiments above were conducted using the four different algorithms described in Section 3 and the recognition results are summarized in Tables 1–4.

4.5. Experiments (supervised learning)

For the sake of completeness and for reference, we also conducted a supervised learning based version of experiments I–III. We prepared a modified (for multi-view face recognition) version of the supervised algorithm ARENA [27], which is summarized in (1)–(4) below:

(1) the proximity matrix $\mathbf{P}_{N\times N}$ of N face sequences from K face categories is calculated in the same way as for the unsupervised learning case;
(2) p% of the samples ($p = 20$, 40, 60, and 80%) from each category are selected randomly as training data, the rest is used as test data;
(3) for each test sample, the distance to each of the training samples is calculated (actually read from the already calculated proximity matrix $\mathbf{P}$), and the test sample is classified to the face category corresponding to the best match (minimal distance);
(4) the final recognition rate r (percent correctly classified test samples) is calculated as the average between 30 different runs of (2)–(3), that is using different random partitions of training/testing sample sequences.

The same datasets used in experiments I–III from Section 4.4 have been used, although subjects for which there were less than three sequences (the rare visitors) were excluded from the training set. The results are summarized in Tables 5(a) and (b).

Table 5
Recognition results for the supervised learning based experiments

Experiment	Sequences	Face categories	r (%)			
			$p = 20\%$	$p = 40\%$	$p = 60\%$	$p = 80\%$
(a)						
I	98	14	83.47	93.98	97.89	99.05
II	534	17	87.80	94.75	97.33	98.90
III	275	17	88.55	95.97	97.76	98.73
(b)						
I	98	14	83.10	94.22	97.72	99.05
II	534	17	85.92	93.95	96.70	98.65
III	275	17	85.99	94.94	97.31	98.55

The between-sequence distance was the minimal distance, while the between-face distance in (a) was the L_0^* norm in (2) (as proposed in [27]) and the L_2 norm for (b). The recognition rates r were calculated as the average between 30 different runs of the algorithm described in Section 4.5, for different randomly selected partitions of training/testing sample sequences (p is the percent of the samples from each category which are selected randomly as training data, the rest being used as test data).

4.6. Results

As can be seen from the experimental results in Tables 1–4, all four clustering methods managed to cluster the face sequences into identity categories to some extent. Overall, the *CAR1* algorithm produced the best results, with recognition rates over 90% for all experiments and acceptable structural quality of the obtained partitions, the latter being judged by comparing the number of obtained clusters to the number of the original identity clusters (as was explained above, if those are not very different, it is unlikely to have an oversegmentation) and small values for E_O (which means there was no undersegmentation either). A certain difference between the number of obtained clusters and the number of the original identity clusters is acceptable and inevitable, having in mind that the illumination conditions were very demanding and the data was taken over a long period of time, while the distance measures were not specifically chosen to be invariant under such conditions.

The results obtained for the *CAR2* algorithm were comparable to those obtained by *CAR1*, probably reflecting their common strategy to use both repulsion and attraction, although their optimization strategies are different. An undersegmentation was produced by *CAR2* for experiment II when the Hausdorff metric was used, as can be judged by the high value of E_O for this case. Also, *CAR2* is the only algorithm among those compared here, which cannot run in real-time for more than several hundreds of sequences, that means that it is necessary to find faster ways to implement the optimization in (17) and (18).

Of the other two algorithms, *NCut* also produced results that were comparable to those of *CAR1* and *CAR2*, although it should be noticed that for this algorithm, the thresholds which determine the split into two partitions were adjusted manually at each step (as explained in Section 3.4) in order to obtain the best possible results. *TCut*, also fully automatic like *CAR1* and *CAR2* (although it provides no guidance how to determine σ and how to choose between candidate partitions corresponding to different values of $\Delta T(r)$, so we tried different values for those and reported the best results), produced acceptable results for experiment I, although for experiments II and III it typically produced either severe oversegmentation (more than half of the sequences forming singletons, which is meaningless as a partitioning), or undersegmentation, characterized by several huge clusters in which several categories were mixed together (thus the high values of E_O).

The results obtained from the supervised learning based experiments in Section 4.5 show that if sufficient training data is used, nearly perfect results can be obtained in the supervised mode. This shows that whenever category-specific information is available and the required labeling of the input video stream into categories is not too costly, the use of such information is advantageous. On the other hand, in situations in which category-based information is either unavailable or impractical to use, the general unsupervised approach for face recognition we propose in this paper can be utilized, producing results comparable to the supervised case, even though based on much less information.

5. Conclusion

In this paper we have proposed a novel method for unsupervised face recognition from video sequences of time-varying face images obtained over an extended period of time in real-world conditions. The learning process implemented by the method does not rely on category-specific information provided by human teachers in advance, but rather lets the system find out by itself the structure and underlying relations inherent in the sensory input. The proposed method provides the following important advantages: (a) it allows all stages of the resulting system to be completely automated, avoiding the need for manual segmentation and labeling of the input stream; (b) it can be easily implemented as a pairwise clustering algorithm (*CAR1*) which is simple, fast and robust to noise in the data; (c) there are no free parameters which cannot be set in an objectively determined way; (d) both frontal and side-view faces can be recognized by the method.

In addition, CAR1 and CAR2, the two novel pairwise clustering algorithms introduced in this paper, exhibit several properties which render them interesting as general clustering methods, even outside the context of the specific problems treated in this paper. The clustering process is guided by two types of interaction forces, attraction and repulsion, imposing both positive and negative values on the matrices of pairwise relations derived from the original proximity matrices, allowing fuller exploitation of the pairwise information hidden in the proximity data. This also leads to a natural formulation both of an optimization criterion (defined in terms of the interplay between attraction and repulsion) and a clustering validation criterion (determining the suitable trade-off between over- and undersegmentation of the data, based on the normalized global partition energy). Additional experiments (not reported here) on several standard clustering datasets revealed also that by controlling the level of repulsion, the clustering algorithm can be modified to detect clusters of either complex forms, or of more compact ones. Another important advantage of the proposed clustering algorithms is that it is not necessary to know in advance the number of clusters (which is a too strong and unrealistic assumption for many practical problems)—it is found directly from the data as the one giving the most natural partitioning of the dataset, in terms of the optimized partitioning criterion, for the given metric.

In this paper we report results from several face recognition test experiments using both frontal and side-view face sequences obtained under demanding real-world conditions. The results seem encouraging, having in mind the difficulty of the task, the bottleneck of the unreliable preprocessor output and the sub-optimal distance measures used. In these tests, the new algorithms compared favorably to two other pairwise clustering algorithms, which have been applied successfully for solving similar problems in image segmentation tasks.

It is expected that the proposed method can find application in video surveillance systems, as an integral part of a human–computer interface, for content-based information retrieval from video databases of multi-view objects (like faces), and generally in situations when manual segmentation and labeling of the input video stream into categories might be considered impractical or impossible.

Acknowledgments

The authors are grateful to Dr. K. Ishii and Dr. N. Sugamura of NTT CS Laboratories for their help and encouragement, and to the anonymous reviewers for their comments.

References

[1] A. Samal, P.A. Iyengar, Automatic recognition and analysis of human faces and facial expressions: a survey, Pattern Recognition 25 (1992) 65–77.
[2] W. Zhao, R. Chellapa, A. Rosenfeld, P.J. Phillip, Face recognition: a literature survey. Available from ftp://ftp.cfar.umd.edu/TRs/CVL-Reports-2000/TR4167-zhao.ps.gz, 2000.
[3] M.A. Grudin, On internal representation in face recognition systems, Pattern Recognition 33 (2000) 1161–1177.
[4] H. Wechsler, P.J. Philips, V. Bruce, F.F. Soulie, T.S. Huang (Eds.), Face Recognition: From Theory to Applications, Springer, Berlin, 1998.
[5] H. Ando, S. Suzuki, T. Fujita, Unsupervised visual learning of three-dimensional objects using a modular network architecture, Neural Networks 12 (1999) 1037–1053.
[6] J.J. Weng, W.S. Hwang, Toward automation of learning: the state self-organization problem for a face recognizer, in: Proc. 3rd Int. Conf. on Automatic Face and Gesture Recognition, 1998, pp. 384–389.
[7] B. Moghaddam, A. Pentland, Face recognition using view-based and modular eigenspaces, in: Automatic Systems for the Identification and Inspection of Humans, SPIE, vol. 2277, 1994.
[8] L. Wiskott, J.M. Fellous, N. Kruger, C. von der Malsburg, Face recognition by elastic bunch graph matching, IEEE Trans. PAMI 19 (7) (1997) 775–779.
[9] A.L. Yuille, Deformable templates for face recognition, J. Cognitive Neurosci. 3 (1) (1991) 59–70.
[10] D. Beymer, T. Poggio, Face recognition from one example view, in: Proc. ICCV, 1995, pp. 500–507.
[11] S.J. McKenna, S. Gong, Face recognition in dynamic scenes, in: British Machine Vision Conference, 1997.
[12] G. Edwards, C. Taylor, T. Cootes, Learning to identify and track faces in sequences, in: Proc. 3th Int. Conf. on Automatic Face and Gesture Recognition, 1998, pp. 260–267.
[13] S. Satoh, Comparative evaluation of face sequence matching for content-based video access, in: Proc. 4th Int. Conf. on Automatic Face and Gesture Recognition, 2000, pp. 163–168.
[14] V. Krueger, S. Zhou, Exemplar-based face recognition from video, in: Proc. ECCV 2002, Lecture Notes in Computer Science, vol. 2353, 2002, pp. 732–746.
[15] C. Stauffer, E. Grimson, Similarity templates for detection and recognition, in: Proc. CVPR 2001, 2001, pp. I221–I228.
[16] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surveys 31 (3) (1999) 264–323.
[17] B.S. Everitt, Cluster Analysis, Wiley, New York, 1993.
[18] R. Duda, P. Hart, D. Stork, Pattern Classification, Wiley, New York, 2001.
[19] A.K. Jain, R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[20] B. Mirkin, Mathematical Classification and Clustering, Kluwer Academic Publishers, Dordrecht, 1996.
[21] K. Rose, E. Gurewitz, G. Fox, Statistical mechanics and phase transition in clustering, Phys. Rev. Lett. 65 (1990) 945–948.
[22] K. Rose, E. Gurewitz, G. Fox, Constrained clustering as an optimization method, IEEE Trans. PAMI 15 (1993) 785–794.
[23] J.S. Nowlan, G.E. Hinton, Evaluation of adaptive mixtures of competing experts, in: Advances in Neural Information Processing Systems, vol. 3, 1991, pp. 774–780.
[24] R.N. Dave, R. Krishnapuram, Robust clustering methods: a unified view, IEEE Trans. Fuzzy Syst. 5 (2) (1997) 270–293.

[25] T. Hofmann, J. Buhmann, Pairwise data clustering by deterministic annealing, IEEE Trans. PAMI 19 (1) (1997) 1–14.
[26] T.C. Hu, Combinatorial Algorithms, Addison-Wesley, Reading, MA, 1982.
[27] T. Sim, R. Sukthankar, M. Mullin, S. Baluja, Memory-based face recognition for visitor identification, in: Proc. IEEE Conf. on Face and Gesture Recognition, Grenoble, 2000.
[28] M.P. Dubuisson, A.K. Jain, A modified Hausdorff distance for object matching, in: Proc. ICPR, Jerusalem, Israel, 1994, pp. A566–A568.
[29] W.J. Rucklidge, Efficiently locating objects using the Hausdorff distance, Int. J. Comput. Vision 24 (3) (1997) 251–270.
[30] B. Raytchev, H. Murase, VQ-faces—unsupervised face recognition from image sequences, in: Proceedings ICIP, 2002, pp. II-809–II-812.
[31] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[32] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, Equations of state calculations by fast computing machines, J. Chem. Phys. 21 (1953) 1087–1092.
[33] P.J.M. van Laarhoven, E.H.L. Aarts, Simulated Annealing: Theory and Applications, Reidel, Hingham, MA, 1987.
[34] S. Geman, D. Geman, Stochastic relaxation, Gibbs distribution and Bayesian restoration of images, IEEE Trans. Pattern Anal. Machine Intell. 6 (1984) 721–741.
[35] J. Shi, J. Malik, Normalized cuts and image segmentation, in: Proc. CVPR, 1997, pp. 731–773.
[36] F.R.K. Chung, Spectral Graph Theory, American Mathematical Society, Providence, RI, 1997.
[37] J. Costeira, T. Kanade, A multibody factorization method for motion analysis, in: Proc. ICCV, 1995, pp. 1071–1076.
[38] S. Sarkar, K.L. Boyer, Quantitative measures of change cased on feature organization: eigenvalues and eigenvectors, in: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 1996, pp. 478–483.
[39] P. Perona, W.T. Freeman, A factorization approach to grouping, in: H. Burkardt, B. Neumann (Eds.), Proc. ECCV, 1998, pp. 655–670.
[40] Y. Weiss, Segmentation using eigenvectors: a unifying view, in: Proc. ICCV, 1999, pp. 975–982.
[41] R. Kannan, S. Vempala, A. Vetta, On Clustering: Good, Bad and Spectral. in: Proc. 41s Annual Symposium on Foundations of Computer Science, 2000.
[42] Y. Gdalyahu, D. Weinshall, M. Werman, Self-organization in vision: stochastic clustering for image segmentation, perceptual grouping and image database organization, IEEE Trans. PAMI 23 (10) (2001) 1053–1074.
[43] M. Blatt, S. Wiseman, E. Domany, Data clustering using a model granular magnet, Neural Computation 9 (1997) 1805–1842.
[44] D. Karger, C. Stein, A new approach to the minimum cut problem, J. ACM 43 (1996) 601–640.
[45] B. Raytchev, H. Murase, Unsupervised face recognition by associative chaining, Pattern Recognition 36 (1) (2003) 245–257.
[46] E. Hjelmas, B.K. Low, Face detection: a survey, Comput. Vision Image Understanding 83 (2001) 236–274.
[47] M.H. Yang, D.J. Kriegman, N. Ahuja, Detecting faces in images: a survey, IEEE Trans. Pattern Anal. Machine Intell. 24 (1) (2002) 34–58.