Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 40 (2007) 3069–3080
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/pr

An elastic partial shape matching technique

Longin Jan Latecki*, Vasileios Megalooikonomou, Qiang Wang, Deguang Yu

Computer and Information Sciences Department, Temple University, 1805 North Broad Street, Philadelphia, PA 19122, USA

Received 23 March 2006; received in revised form 21 December 2006; accepted 5 March 2007

Abstract

We consider the problem of partial shape matching. We propose to transform shapes into sequences and utilize an algorithm that determines a subsequence of a target sequence that best matches a query. In the proposed algorithm we map the problem of the best matching subsequence to the problem of a cheapest path in a directed acyclic graph (DAG). The approach allows us to compute the optimal scale and translation of sequence values, which is a nontrivial problem in the case of subsequence matching. Our experimental results demonstrate that the proposed algorithm outperforms the commonly used techniques in retrieval accuracy.

Keywords: Shape similarity; Sequences matching; DAG; Shortest path

1. Introduction

Shape matching deals with the problem of describing a shape and calculating its similarity to others. There are a huge variety of shape descriptors; most of them require the presence of the whole shape, only some of them tolerate minor missing or distorted parts, e.g., due to minor occlusion. In general, the performance of existing shape matching techniques tends to drop significantly when too much noise exists in the shape. Due to occlusion or segmentation errors, it may be the case that only parts of objects in a given image have correct contours. Therefore, a shape similarity measure based on parts of objects is needed.

The cognitive importance of parts of visual form in human perception has been theoretically and experimentally verified [1,2]. However, the identification of shapes given their parts is still an unsolved problem in shape similarity. As a good example, Fig. 1 clearly demonstrates the difficulties we may encounter in partial shape matching.

Given a significant part of visual form as a query, e.g., the fish tail shown in Fig. 1(a), our goal is to find similar shapes (fishes) containing the query part; this means that we need to find this tail as part of some other fish contours, e.g., it could be as shown in Fig. 1(b) or (c). Three serious problems arise:

1. length problem,
2. scale problem, and
3. distortion problem.

We describe these problems in details in Section 2. None of current shape matching techniques provides solutions to all problems listed above. Even feature-based approaches, although potentially being based on local features, require the presence of most of the object to compute the statistics of the features. This statement applies to all shape descriptors presented in the special issue of pattern recognition on shape similarity, Latecki et al. [3], as well as to the shape descriptors presented in Belongie et al. [4] and Grigorescu and Petkov [5]. The existing partial shape similarity measures (e.g., Ref. [6]) require that the query part is nearly identical to the corresponding part of the target contour, which is clearly an unrealistic assumption due to noise distortions and due to (even small) perspective projection changes. Veltkamp and Tanase [7] proposed to use an extended dynamic programming approach directly on the turn angle function (also known as Ψ function) representation of object contours. Since their matching of a single part is not elastic, their approach is not scale invariant, e.g., if part of the target contour that is similar to the query part is twice longer

* Corresponding author. Tel.: +1 215 204 5781; fax: +1 215 204 5082.
E-mail address: latecki@temple.edu (L.J. Latecki).

0031-3203/$30.00
doi:10.1016/j.patcog.2007.03.004

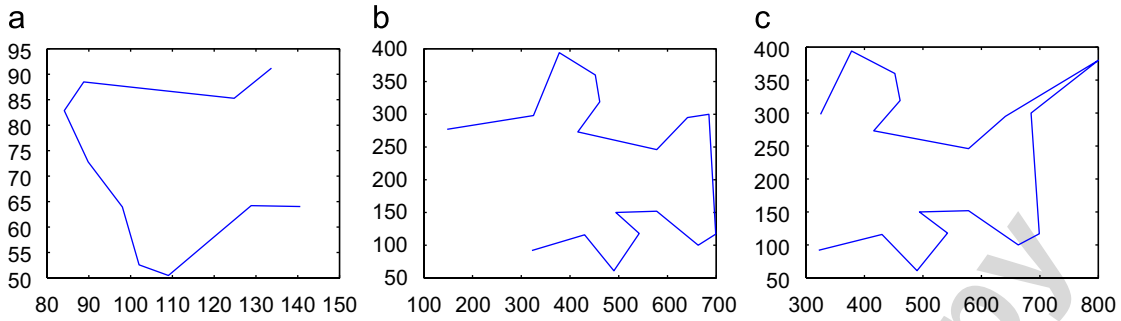

Fig. 1. (a) A fish tail contour t_1. (b) A contour of partially occluded fish t_2. (c) A distortion led to a spike on the fish tail contour t_3.

(due to a different scale), it will not be recognized as similar. Therefore, their approach requires scale normalization, which is a nontrivial and unsolved problem.

In this paper, we present a novel approach for evaluating shape similarity. By transforming shapes into sequences of numbers, we propose a dynamic programming algorithm to find the best corresponding part of the contour of a target object to a given query part. The measure of similarity is calculated simultaneously. Compared to traditional techniques, this new approach has the following advantages:

- It performs well even in the case that only a small part of the shape is visible, therefore allowing for significant occlusion.
- It is the first approach (to the best of our knowledge) that can recognize a given shape when only a small part of its contour is given and the given part can be noisy. Being different from existing similarity measures, it does not require the query part to be nearly identical to the corresponding part on the target contour, which makes it useful to real life applications.
- In contrast to early AI and CV approaches, it does not require objects to be composed of primitive parts or a specific shape vocabulary such as generalized cylinders, superquadrics, or geons [8–11].
- It has no restriction on shape complexity since the visual parts it considers are not built of any primitive elements.
- By mapping the problem of the best matching subsequence to the problem of a cheapest path in a DAG it allows for the computation of the optimal scale and translation of sequences values, which is a nontrivial problem in the case of subsequence matching.

The rest of the paper is organized as follows. In Section 2 we introduce our motivation and the background of sequence matching techniques in the community of data mining. Section 3 presents the details about the proposed technique, minimum variance matching (MVM) [12]. Experimental results are given in Section 4 and conclusion remarks are given in Section 5.

2. Motivation and background

Given a query part, our goal is to find the corresponding part in shapes that actually contain this part, though the correspondence may be noisy and therefore the shapes may not be identical to the query. The three problems listed in the previous section make shape matching a very complicated and challenging task. A query and its corresponding part may have different lengths, which makes the problem hard to address with most shape descriptors. In addition, the scale and distortion problems bring even more difficulties in the process of shape matching.

To improve the effectiveness of partial shape matching, we need an elastic matching for shapes that is robust enough to cancel out the effect of the three problems. Motivated by the existing techniques in the field of sequence analysis dealing with similar problems, we propose to represent the object shapes with sequences of numbers and hence transform the shape matching problem into a problem of sequence matching. Sequences of real numbers are commonly used in all research fields. Due to their simplicity, a sequence is a ubiquitous and increasingly prevalent type of data. Therefore, there has been much research effort devoted to sequence matching or similarity in recent years. Many data mining algorithms have sequence similarity measurements at their core. Examples include motif discovery [15], anomaly detection [16], rule discovery [17], classification [18] and clustering [19].

There exist many methods to transform contours of planar shapes to sequences. Most methods begin with placing N equidistant sample points $s_1, s_2, \ldots, s_N$ on the given contour, and then map the contour to a sequence of numbers $f(s_1), f(s_2), \ldots, f(s_N)$ that represent some shape feature f. For example, as illustrated in Fig. 2(a), we can equidistantly sample the contour and compute for every sample point the distance to the contour centroid. However, the obtained sequence values suffer then from the scale problem, e.g., an object twice closer to the viewer leads to twice larger sequence values. A different method is to compute the tangent line at every sample point, and represent the contour with the sequence of directions of the tangent lines (their angles with x-axis). An example tangent line is shown in Fig. 2(b). The sequence obtained this way for the contour in 2(b) is shown in 2(c). It is a sequence of numbers between 0 and 360 that represent the angles of tangent lines at corresponding contour points with x-axis. The starting point is the lowest contour point, and we traverse the contour in the counter clockwise direction. The obtained sequence is a discrete version of the so-called Ψ function. For an arc length parameterized contour, $\Psi(x)$ is the tangent angle at each contour point x. We selected this contour representation, since it

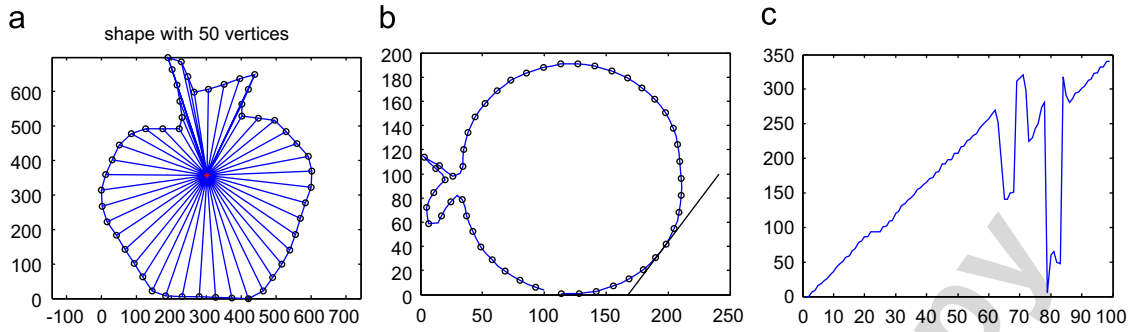

Fig. 2. Different ways to convert contours to sequences of numbers.

is scale invariant; the usage of this representation provides a solution to the scaling problem. It is also translation invariant, but it is not rotation invariant since, for example, if we rotate an object by $+60°$, all sequence values will be translated by 60 (modulo 360°). Thus, object rotation implies shift of sequence values (modulo 360°).

However, a more serious problem is robust estimation of the tangent lines, since due to possible noise that results in displacement of the contour points, the directions of the tangent lines may become unstable. To robustly estimate the tangent lines, we use the process of discrete curve evolution (DCE) introduced in Refs. [13,14]. DCE is capable of removing noisy contour points and retaining the shape relevant points.

Now we describe in detail the three problems illustrated in Fig. 1, and show how they are addressed in the proposed approach. Let a significant visual part be given as a query t_1. For example, this can be a fish tail shown in Fig. 1(a). Our goal is to find a fish in a given image, which really means that we need to find t_1 as part of some other part t_2 of the fish contour, e.g., t_2 could be as shown in Fig. 1(b). Let t_2' be the fish tail part of contour t_2 shown in Fig. 1(b). Our goal is to find out that t_1 and t_2' are similar. (Observe that t_1 and t_2' are not identical; they are just similar.) Problems (1) and (2) imply that arcs t_1 and t_2' may have different lengths. In addition, the scale problem may imply (depending on the shape representation used) that the values of sequences representing t_1 and t_2' are at different scales. As stated above, we solve the scale problem (2) by representing each shape as a sequence of tangent directions at each sample point.

Observe that the length problem (1) can be easily solved when complete contours are given, by arc length normalization to one. However, length normalization does not work for partial contours, since normalizing the lengths of arcs t_1 and t_2 to be equal, does not imply that the lengths of corresponding arcs t_1 and t_2' are equal. One could think that a sliding window approach would provide a solution to problem (1). In addition to the computational cost of enumerating all possible sliding window sizes, i.e., matching t_1 to all subarcs of t_2, there is a more serious issue with this solution. It is problem (3); it affects sequences representing complete contours as well. Distortions may arise from segmentation errors (segmentation artifacts), change of view angle, or from occlusions. For example, consider the arc t_3 in Fig. 1(c), which can be interpreted as a distorted version of arc t_2. It might have been caused by a spike occluding part of the fish tail. Due to the spike, there does not exist a single subarc of t_3 that is similar to t_1. Contour part t_1 can only correctly match to two disconnected subarcs of t_3 obtained by removing the spike. While sliding windows are computationally tractable, the fact that a query part may only correctly match to several subarcs of the target contour makes it necessary to use many disconnected sliding windows, which leads to combinatorial explosion.

Since we represent contour parts as sequences of numbers representing sample points, the length problem (1) leads also to the problem of determining the number and position of the sample points. For example, if t_1 is represented by more sample points than its corresponding part of t_2, and our matching function is 1–1, then it is impossible to match t_1 to its corresponding part of t_2.

We obtain a solution to both the length problem (1) and the distortion problem (3) by using an elastic sequence matching approach called MVM, introduced in Section 3. The proposed approach allows us to match sequences of different lengths, and to skip the elements of the target sequence (e.g., the spike) that do not correspond to the query sequence. As discussed earlier, when we transform shape data into sequences using tangent angles, the scaling problem (2) is resolved. However, tangent angles are not rotation invariant; when a shape is rotated, we obtain a shift in sequence values. Therefore, we need to extend MVM to allow us to identity the shift in sequences values. The extension of MVM that solves the shift problem is described at the end of Section 3.

We now provide a brief overview of sequence distance measures, before we introduce MVM. As many researchers have mentioned in their work [17,18,20], the Euclidean distance, even though most commonly used, is not always the optimal distance (dissimilarity) measure for sequence similarity searches. For example, in some sequences, different parts have different levels of significance in their meaning. Also, the Euclidean distance does not allow shifting in time axis and becomes infeasible when the compared sequences have different lengths.

To solve the problems of time shifting and different length, dynamic time warping (DTW) [21,22] was proposed to align the time axis prior to the calculation of the distance. To align two sequences $X = (x_1, x_2, \ldots, x_m)$ and $Y = (y_1, y_2, \ldots, y_n)$

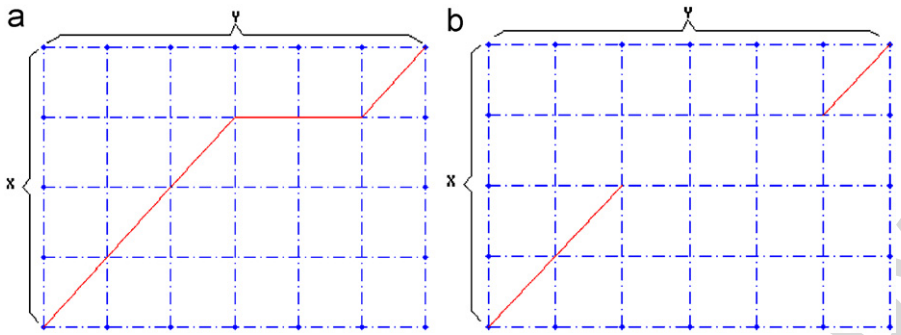

Fig. 3. Time series alignment with (a) DTW and (b) MVM.

using DTW, we first need to build an m-by-n matrix M with each element $M(i, j)$ $(1 \leqslant i \leqslant m; 1 \leqslant j \leqslant n)$ being the squared distance between $X(i)$ and $Y(j)$: $M(i, j) = d(X(i), Y(j))^2$. To find the optimal correspondence between X and Y, we need to find a path through the matrix that has the minimum cumulative distance. The minimum cumulative distance of the path leading to the entry (i, j) is calculated with dynamic programming using the following recurrence:

$$DTW(i, j) = M(i, j) + \min(DTW(i-1, j), DTW(i, j-1), DTW(i-1, j-1)).$$

After the optimal path is found, the DTW distance between two sequences is calculated as the sum of all the $M(i, j)$ entries included in the path. Fig. 3(a) demonstrates the alignment path found with DTW for two sequences $X = (1, 2, 8, 6, 8)$ and $Y = (1, 2, 9, 3, 3, 5, 9)$. The corresponding sequence indices (not the values) are (1,1), (2,2), (3,3), (4,4), (4, 5), (4, 6), (5,7).

The DTW distance has been shown to be superior to the Euclidean distance in many cases [19,23–25] (see Ref. [26] for a detailed discussion of DTW). However, it requires the matched sequences to be well aligned, and it is particularly sensitive to outliers, since it is not able to skip any elements of the target series. DTW always matches the query sequences to the whole target sequences. This fact not only leads to unintuitive correspondences of elements, but also influences negatively the sequence distance. This is illustrated by the index pairs (4, 5), (4, 6), which mean that the fourth element of X (with value 6) is forced to match to the fifth and sixth elements of Y (of value 3). Intuitively, the fourth element of X (with value 6) should only correspond to the fifth element of Y (of value 3), as shown in Fig. 3(b).

Another distance measure, the longest common subsequence (LCSS), has been used in sequences [27,28] to deal with the alignment and outliers problems. Given a query and a target series, LCSS determines their longest common subsequence, i.e., LCSS finds subsequences of the query and target (of the same length) that best correspond to each other. The distance between two sequences is calculated based on the ratio between the length of their longest common subsequence and the length of the whole sequence. Since the problem of longest common subsequence has the property of optimal substructure, it is often solved with dynamic programming. Given two sequences $X = (x_1, x_2, \ldots, x_m)$ and $Y = (y_1, y_2, \ldots, y_n)$, the length of their longest common subsequence is calculated as:

$$LCSS(X_{1\ldots i}, Y_{1\ldots j}) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0, \\ LCSS(X_{1\ldots i-1}, Y_{1\ldots j-1}) + 1 & \text{if } x_i = y_j, \\ \max(LCSS(X_{1\ldots i-1}, Y_{1\ldots j}), & \\ \quad LCSS(X_{1\ldots i}, Y_{1\ldots j-1})) & \text{otherwise.} \end{cases}$$

In the above formula, $X_{1\ldots i} = (x_1, x_2, \ldots, x_i)$ $(0 \leqslant i \leqslant m)$, $Y_{1\ldots j} = (y_1, y_2, \ldots, y_j)$ $(0 \leqslant j \leqslant n)$.

As shown in the formula, the longest common subsequence found with LCSS does not need to consist of consecutive points, the order of points is not rearranged, and some points can remain unmatched. However, since LCSS was originally proposed for symbolic sequences (i.e., character strings), when one tries to apply it to sequences of numeric values, a threshold is required to determine when two close numeric values can be treated as equal [28]. The actual performance of LCSS heavily depends on the correct setting of this threshold, which may be a particularly difficult problem for some applications.

In this paper, we propose to use a new algorithm called MVM [12] for partial shape matching. MVM combines the strengths of both DTW and LCSS, while overcoming their constraints. MVM computes the distance value between two sequences that are obtained from shape boundaries. It calculates the shape similarity directly based on the distances of corresponding elements, just as DTW does, but also allows the query sequence to match to only a subsequence of the target sequence, just as LCSS does. Like DTW, MVM also tries to find an optimal path including all the corresponding pairs. But MVM is able to skip outliers during the matching process and the path does not need to be consecutive (as shown in Fig. 3(b)). As Fig. 4 shows, MVM can skip some elements of the target sequence while DTW demands that every point on both sequences has a match. The main difference between LCSS and MVM is that LCSS optimizes over the length of the longest common subsequence (which requires a distance threshold), while MVM directly optimizes the sum of distances of corresponding elements (without any distance threshold). LCSS allows skipping elements of both the query and the target sequence. Therefore, MVM should be used when one is interested in finding the best matching part of the target sequence for a given query sequence, since it guarantees that the whole query sequence will

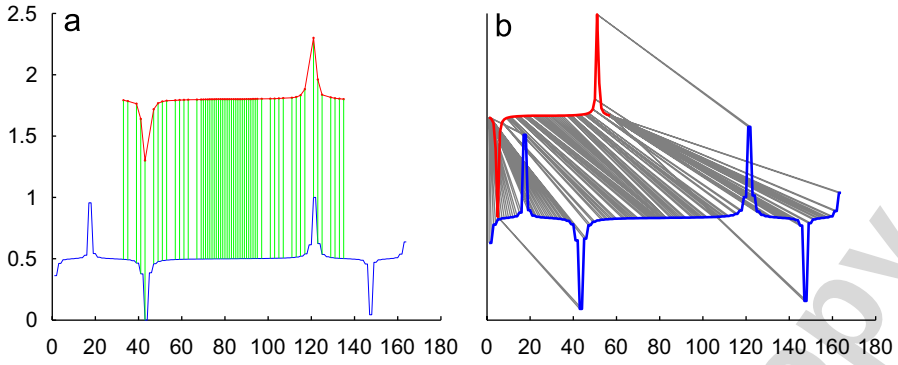

Fig. 4. The correct correspondence established by MVM for a query part matched to the target. (b) DTW was not able to establish a correct correspondence for the same sequences, since it cannot skip any elements.

be matched. This is, for example, the case, when the query is a model sequence, one wants to find in a given data set. However, when the query sequence contains outliers and skipping them is allowed, then LCSS should be used.

3. Minimal variance matching

Before we can do the actual matching for the shape data, we first need to extract their descriptors and represent them with sequences of real numbers. To obtain the sequences representing object shapes, we selected a simple geometric transformation that is very suitable for robust shape representation. To be concise, we will concentrate on 2D objects.

In general we do not know the object orientation, and normalization does not solve the problem if the query corresponds only to part of the target sequences. Therefore, we need a sequences similarity function that estimates the shift of values of corresponding elements. Ideally, this similarity function should be able to deal with elastic matching of two sequences a and b of possibly different lengths m, n, correspondingly, where $m < n$. Currently, the most popular method for elastic matching of sequences is DTW, which minimizes the Euclidean distance of corresponding points. DTW yields an optimal (order preserving) relation R of all elements of sequence a to all elements of sequence b. Thus, each a_i must correspond to some b_j and vice versa. A potential problem arises when sequence a corresponds only to part b' of sequence b, e.g., b' is contained in the first half of sequence b, since the final dissimilarity value between a and b is the sum of dissimilarity values of pairs (of corresponding elements of a and b) in R. The problem with DTW is that elements of b that are not in b', which can be treaded as outliers, must correspond to some elements of a.

Here we propose a solution to this problem in that we replace the relation R with an injection (1–1 mapping) from a to b. Precisely, we require that each a_i maps to exactly one b_j in an order preserving manner while not all elements of b are required to participate in R. This requirement allows us to eliminate the influence of outliers in b on the dissimilarity value between a and b, since sequence a matches to a subsequence of b. We introduce a method to find such an optimal subsequence of b next. When a query sequence a matches only to part of sequence b, this method allows us to find the part of b best matching a while computing a dissimilarity value between a and b, i.e., no additional computation is needed in this case.

The new algorithm, which is called MVM, works for elastic matching of two sequences of different lengths m and n. More specifically, for two finite sequences of real numbers $a = (a_1, \ldots, a_m)$ and $b = (b_1, \ldots, b_n)$ with $m < n$, the goal is to find a subsequence b' of b of length m such that a best matches b'. Thus, we want to find the best possible correspondence of sequence a to a subsequence b' of b. Formally, we define a *correspondence* as a monotonic injection

$$f: \{1, \ldots, m\} \to \{1, \ldots, n\}.$$

(i.e., a function f such that $f(i) < f(i+1)$ such that a_i is mapped to $b_{f(i)}$ for all $i \in \{1, \ldots, m\}$). The set of indices $\{f(1), \ldots, f(m)\}$ defines the subsequence b' of b. (Recall that in the case of DTW, the correspondence is a relation on the set of indices $\{1, \ldots, m\} \times \{1, \ldots, n\}$, i.e., a one-to-many and many-to-one mapping.) Once the correspondence is known, it is easy to compute the distance between the two sequences. We do not have any restrictions on distance functions, i.e., any distance function is possible. To allow for comparison to the existing techniques, we use the Euclidean distance in this paper:

$$d(a, b, f) = \sqrt{\sum_{i=1}^{m} (b_{f(i)} - a_i)^2}. \tag{1}$$

Our goal is to find a correspondence f so that $d(a, b, f)$ is minimal. More precisely, an optimal correspondence $\hat{f}$ of values in series a to values in series b is defined as the one that yields the global minimum of $d(a, b, f)$ over all possible correspondences f:

$$\hat{f} = \arg\min\{d(a, b, f) : f \text{ is a correspondence}\}. \tag{2}$$

Finally, the optimal distance is obtained as

$$d(a, b) = d(a, b, \hat{f}) = \sqrt{\sum_{i=1}^{m} (b_{\hat{f}(i)} - a_i)^2}. \tag{3}$$

In other words $d(a, b)$ is the global minimum over all possible correspondences.

$$r = \begin{bmatrix} (0) & 1 & 8 & 2 & 2 & 4 & 8 \\ 1 & (0) & 7 & 1 & 1 & 3 & 7 \\ -7 & -6 & (1) & -5 & -5 & -3 & 1 \\ -5 & -4 & 3 & -3 & -3 & (-1) & 3 \\ -7 & -6 & 1 & -5 & -5 & -3 & (1) \end{bmatrix}$$

Fig. 5. The difference matrix of two sequences $t_1 = (1, 2, 8, 6, 8)$ and $t_2 = (1, 2, 9, 3, 3, 5, 9)$ formed with rows corresponding to elements of t_1 and columns to elements of t_2.

We can also state the correspondence problem in a statistical framework. Let us assume that there is a subsequence b' of b that is a noisy version of a such that

$$a \sim b' - N(0, v),$$

where $N(0, v)$ denotes a zero-mean Gaussian noise variable with variance v, and $b' = (b_{f(i)})_i$ for $i \in \{1, \ldots, m\}$. Since the mean of the differences $(a_i - b_{f(i)})_i$ is zero, i.e., $a - b' \sim N(0, v)$, the variance σ^2 of difference sequence $(a_i - b_{f(i)})_i$ is given by

$$\sigma^2(a, b, f) = \frac{1}{m} \sum_{i=1}^{m} (b_{f(i)} - a_i)^2. \tag{4}$$

Clearly, $\sigma^2(a, b, f) = v$ (the variance of the Gaussian noise). Observe that in this case the variance corresponds to the Euclidean distance (1). Thus, the variance of the difference sequence is minimal when mapping f establishes a correct correspondence of elements of both sequences.

Now we describe the method used to minimize Eq. (4). We first form the difference matrix of two sequences a and b as:

$$r = (r_{ij}) = (b_j - a_i).$$

It is a matrix with m rows and n columns with $m < n$, with r_{ij} being the difference between the values of a_i and b_j. For example, the difference matrix for two sequences $t_1 = (1, 2, 8, 6, 8)$ and $t_2 = (1, 2, 9, 3, 3, 5, 9)$ is shown in Fig. 5. Observe that t_1 and t_2 are similar if we ignore the two elements in t_2 with value 3.

Clearly, r_{ij} can be viewed as a surface over a rectangle of size m by n, where the height at point (i, j) is the value r_{ij}. We obtain the correspondence with minimal variance by solving the least-value path problem on the difference matrix. To obtain the solution, we treat r_{ij} as a directed graph with the following links:

r_{ij} is directly linked to r_{kl}

$$\Leftrightarrow \quad k - i = 1 \text{ and } j + 1 \leqslant l \leqslant j + n - m.$$

When traversing the obtained directed graph, the meaning of both conditions is as follows: for any two consecutive points r_{ij} and r_{kl} in each path, $k - i = 1$ means that we always go to the next row, while $j + 1 \leqslant l \leqslant j + n - m$ means that we can skip some columns with certain elasticity (maximum $n - m$), but cannot go backwards. Fig. 6 shows the constructed DAG.

We want to have a least-value path with respect to the following cost function for each pair of nodes:

$$linkcost(r_{ij}, r_{kl}) = \begin{cases} (r_{kl})^2 = (b_k - a_i)^2 & \text{if } k = i + 1 \\ \quad \text{and } j+1 \leqslant l \leqslant j+1+(n-m)-(j-i), & \\ \infty \quad \text{otherwise.} & \end{cases}$$

The conditions (1) and (2) imply that we can obtain a DAG (directed acyclic graph) G whose nodes are the elements of $(r_{ij})_{ij}$ and weights are defined by the function *linkcost*. Denoting a path leading to r_{ij} as $SP(i, j)$ and its cost as $pathcost(i, j)$, we want to find a path with minimized $pathcost(i, j)$ that satisfies two conditions:

- Starting in the first row, between columns 1 and $n - m + 1$, i.e., at r_{1j} for $j = 1, \ldots, n - m + 1$
- Ending at some node in the last row r_{mj} for $j = m, \ldots, n$.

It is well known that we can solve the least-value path problem using a shortest path algorithm on G. The obtained least-value path defines exactly correspondence $\hat{f}$, which minimizes Eq. (4) in accordance with Eq. (2). We propose a dynamic programming algorithm to solve this optimization problem. Dynamic programming is a method efficient in solving problems exhibiting the properties of overlapping sub-problems and optimal substructure. In our case, the shortest path *SP* can be found based on the *pathcost* of every pair of reachable nodes in G. Taking $pathcost(r_{ij})$ as the sub-problem, the optimal structure of our problem can be formally defined as:

$$pathcost(i, j) = \begin{cases} (r_{ij})^2 \quad \text{if } i = 1, \ 1 \leqslant j \leqslant n; \\ \min(pathcost(i, j), pathcost(i - 1, k) \\ \quad + linkcost(r_{(i-1)k}, r_{ij})) \\ \text{if } 2 \leqslant i \leqslant m, i \leqslant k \leqslant i + n - m, \\ \quad k + 1 \leqslant j \leqslant k + 1 + (n - m); \\ \infty \quad \text{otherwise.} \end{cases}$$

The details of the algorithm are given in Table 1.

The inputs of the MVM algorithm include the lengths of the two sequences, m and n, and the difference matrix. There is also an optional input *winWidth* setting the width of a correspondence window which constraints the elasticity in the path finding process. The cost of the shortest path between each pair of nodes, $(pathcost_{ij})_{ij}$, is returned as output, together with another matrix $path_{ij}$, based on which we can backtrack the shortest path leading to r_{ij}. The actual algorithm starts with an initialization process (steps 1–10). The *pathcost* is set to the square of its own value for every node in the first row and infinity for all the other nodes. In the main loop (steps 11–22), i goes over each row while k and j go over columns in the $(i - 1)$th row and ith row, respectively. The *pathcost* for each node is updated only when there is a shorter path coming from a connected node in the previous row. The optimal structure condition guarantees that the returned matrix *pathcost* contains the cost of the shortest path leading to every node. Since we are looking for a shortest path ending at some node in the last row r_{mj} $(j = m, \ldots, n)$, we just need to check the corresponding

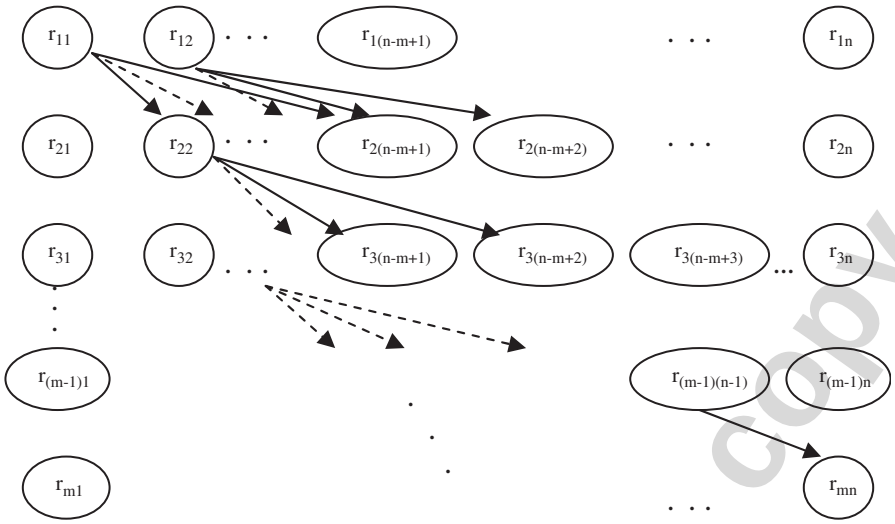

Fig. 6. DAG constructed for two sequences with length m and n.

Table 1
Minimum variance matching (MVM) algorithm

Algorithm: Mininum Variance Matching (MVM)

Input: $m, n, (r_{ij})_{ij}$, *winWidth* (optional)
Output: $(pathcost_{ij})_{ij}$, $(path_{ij})_{ij}$

```
elasticity = min(n − m, winWidth);
for i = 1 : m
    for j = 1 : n
        pathcost(i, j) = ∞;
        path(i, j) = 0;
    end
end
for j = 1 : elasticity + 1
    pathcost(1, j) = r_{1j}^2;
end
for i = 2 : m
    stopk = min(i − 1 + elasticity, n);
    for k = i : stopk
        stopj = min(k + 1 + elasticity, n);
        for j = k + 1 : stopj
            if pathcost(i, j) > pathcost(i − 1, k) + r_{ij}^2
                pathcost(i, j) = pathcost(i − 1, k) + r_{ij}^2;
                path(i, j) = k;
            end
        end
    end
end
```

values in *pathcost*; the minimum value is the distance (dissimilarity) between the two compared sequences.

The shortest path for the example matrix in Fig. 5 is marked with parentheses. Following the parentheses, the optimal correspondence $\hat{f}$ is given by

$$\hat{f}(1) = 1, \quad \hat{f}(2) = 2, \quad \hat{f}(3) = 3, \quad \hat{f}(4) = 6, \quad \hat{f}(5) = 7.$$

Finally, from Eq. (3) we obtain the distance $d(t_1, t_2) = \sqrt{3} \approx 1.732$.

The path computed this way gives us correspondence $\hat{f}$ with the smallest variance of the differences of the corresponding pairs. We recall that this is true, since we assumed that the mean of the differences of the corresponding pairs is zero. Observe that without this assumption, it would not be possible to use the shortest path algorithm on a DAG. The obtained optimal correspondence $\hat{f}$ automatically determines a subsequence $b' = \hat{f}(a)$ of a target sequence b that best matches a query sequence a. In particular, two intuitive interpretations are possible:

- Whole sequence matching: subsequence b' is dense in b, which indicates a similarity of a to b.
- Subsequence matching: subsequence b' is not dense in b but is dense in some parts of b which indicates a similarity of a to those parts of b.

The distinction between whole and subsequence matching is not required for the proposed approach. Therefore, we do not attempt to provide formal definitions. In practice, there are cases in which sharp distinction is not possible.

Under the assumption that the query sequence is shorter than the corresponding sequence of the target sequence, MVM subsequence matching provides a solution to problems (1) and (2). This is a realistic assumption for practical applications, since it only depends on the sampling rate of the query and target contour parts.

3.1. MVM and shift estimation

The definition of the MVM algorithm presented above allows us to estimate the linear transformation that best maps a query sequence a to a subsequence of a target sequence b. This gives a serious advantage with respect to existing sequences matching methods. The estimation is done while computing the correspondence $\hat{f}$, and it does not increase the computational complexity of the algorithm. To focus our attention, we

present here the estimation of the translation (shift) of values of sequences b. For sequences representing the tangent angle at the contour sample points, the shift estimation corresponds to rotation estimation.

Now for two finite sequences of real numbers $a = (a_1, \ldots, a_m)$ and $b = (b_1, \ldots, b_n)$ with $m < n$, the goal is to find a subsequence b' of b of length m (i.e., correspondence $\hat{f}$) and a translation tr such that a best matches $b' + tr$. This means that we want to minimize:

$$d(a, b, f) = \sqrt{\sum_{i=1}^{m} (b_{f(i)} + tr - a_i)^2}. \quad (5)$$

Observe that if a matches to the whole sequence b, a simple normalization of values of both sequences solves the translation problem. However, this is not the case when a matches only to part of b as we described in the introduction. Let f_k be any correspondence from $a = (a_1, \ldots, a_k)$ to $b = (b_1, \ldots, b_m)$ with $k < m$. Then we can estimate the translation for f_k as

$$tr(a, b, f, k) = \sum_{i=1}^{k} b_{f(i)} - a_i. \quad (6)$$

The main idea of the solution to Eq. (5) is the fact that we can update $tr(a, b, f, k)$ incrementally as

$$tr(a, b, f, k+1) = \frac{k}{k+1} tr(a, b, f, k) + \frac{1}{k+1}(b_{f(k+1)} - a_{k+1}). \quad (7)$$

By integrating this incremental update in the process of computation of the cheapest path on DAG, we obtain an optimal solution to Eq. (5).

3.2. Time complexity of MVM

Let m be the length of the query sequence and n the length of the target sequence. The complexity of the shortest path algorithm on a DAG is $O(V + E)$, where V is the number of vertices and E is the number of edges. For efficient implementation, we only need to have a subset of at most $n - m$ elements of each row of $(r_{ij})_{ij}$ as vertices of G. Since there are m rows, we obtain that the number of vertices V is bounded by $m \times (n - m)$. Every vertex r_{ij} in row i is linked to at most $n - m - j + 1$ vertices in row $i + 1$. Since

$$\sum_{j=1}^{n-m+1} (n - m - j + 1) = \sum_{j=1}^{n-m} j = \frac{(n-m)(n-m+1)}{2}$$

and there are m rows, we obtain that G has at most

$$m \times \frac{(n-m)(n-m+1)}{2}$$

edges. Hence there are $O(mn^2)$ edges, and consequently the complexity of our algorithm is $O(mn^2)$. However, this complexity can be reduced if a restriction on the index difference c of corresponding elements is set. We will call constant c (*winWidth* in Table 1) the *correspondence window bound* (CWB) if for each $i = 1, \ldots, m$ $f(i) = j$ implies $f(i+1) \leqslant j + c$.

The bound c directly corresponds to the warping window size restriction in DTW. As stated in Ref. [26], restricting the warping window not only reduces the computational complexity, but also leads to better matching results for DTW. In all our experiments, the usage of CWB did not reduce the retrieval accuracy. Given the CWD, the number of edges for each vertex r_{ij} of G in row i linking it to vertices in row $i + 1$ is bounded by c. Since there are at most $n - m$ vertices in row i and there are m rows, the total number of edges is bounded by $cm(n-m)$. Thus, the computational complexity of our algorithm is bounded by $cm(n - m)$, and consequently, our algorithm with CWB is of order $O(mn)$. This is the best possible complexity if the query string matches a limited part of the target string and we want to find this substring.

When the query string happens to match the whole target, the complexity reduces to linear (without reducing the matching accuracy). Thus, when the task is to match only whole sequences, MVM can be computed in linear time. In this case, we can limit the length of the difference of both sequences $n - m \leqslant K$, where K is a constant. Consequently, we obtain an algorithm of order $O(m)$, where m is the length of the query sequence. We believe that we can further mitigate the (amortized) time complexity of our approach by introducing an admissible lower bounding test in the spirit of Ref. [18]. However, in this work we focus on demonstrating the utility of our approach; we will address speedup and indexability of this approach in future work.

4. Experimental results

We compared the performance of the proposed MVM method to DTW and to LCSS measure. The particular strength of the proposed sequences matching method is the fact that the query may be only similar to part of the target sequences, since it can identify the best matching subsequence of the target sequence. This property is really desirable for partial shape matching.

When LCSS is applied to sequences of numeric values, one needs to set a threshold, θ, that determines when values of corresponding points are treated as equal [28]. LCSS performance heavily depends on the threshold that is a function of the sequence values. To judge the equivalence of two real values, we define this threshold θ as follows:

$$\frac{|b_j - a_i|}{|\min(a_i, b_j)|} \leqslant \theta,$$

where a_i and b_j are considered points of two matched sequences a and b. In our experiments, the value of θ that yields the best results was 0.1. Therefore, we used this value. We stress that optimal threshold settings may vary significantly for different data sets.

We performed partial shape matching experiments on a database of 350 shapes represented by their contours. These shapes are grouped into 70 classes with 5 shapes in each class. This database is part of the original MPEG-7 Core Experiment

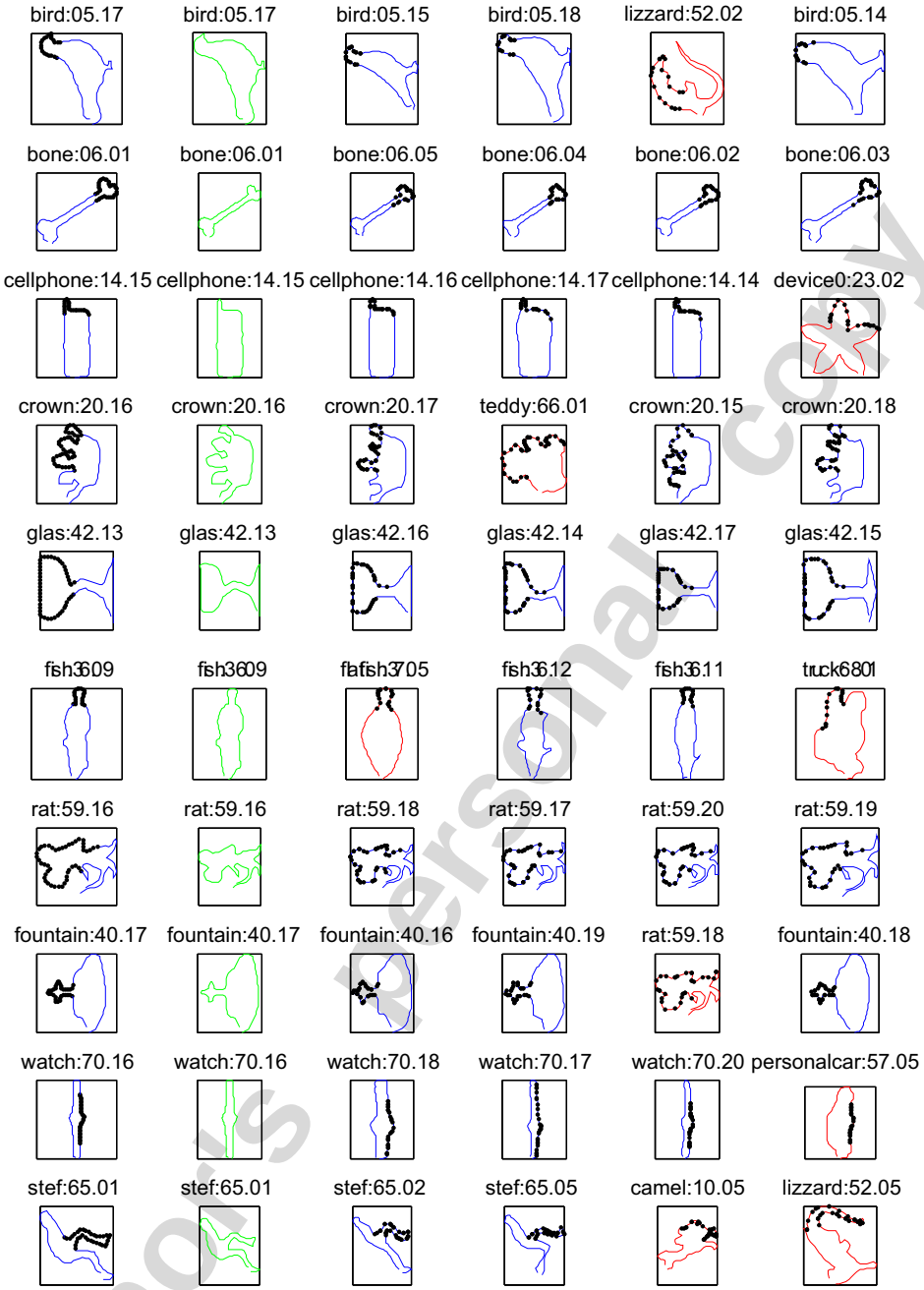

Fig. 7. Query parts and their best five matches found with MVM.

CE-Shape-1 database composed of 70 classes with 20 shapes in each class [13]. Each contour in the database is represented as a sequence of 100 tangent directions ranging between 0 and 360. We used as queries sequences that represent 10 manually selected visual parts of objects. They are represented as sequences of highly varying lengths of 20, 30, 25, 47, 46, 21, 45, 32, 30, and 33 points. We show the corresponding 10 query parts in the first column of Fig. 7. The query parts are marked with black dots on the contours of the shapes from which they are selected. High length variance of selected query sequences reflects the fact that parts of visual form may have different sizes.

Following Ref. [26], we use 1–NN classification accuracy, where the query sequence is assigned to the class of its first nearest neighbor (excluding the query sequence). To compute the classification accuracy, we find for each query part the most similar object excluding the object from which it was chosen. As a result, even though the query parts were selected from objects in the experimental data set, they are not identical to their best matching parts. The results for three different approaches are shown in Table 2.

To demonstrate the experimental results visually, we also show in Fig. 7 (columns 2–6) the five most similar shapes found with MVM for all query parts, with mismatches marked in red.

For most queries, all the best five matches come from the same class. Observe that those mismatches, although they belong to different object categories, actually have a contour part that is very similar to the query part. This is the case for objects shown in the following (row, column): (1,5), (3, 6), (4, 4), (6, 3), (6, 6), (9, 6), (10, 5), (10, 6). The only mismatch, where the query part seems not to be similar to the matching part is (8, 5).

With the same data set, we also performed experiments following the MPEG-7 Bulls-Eye test [30]. We measured the retrieval rate as the number of objects from the same class that are contained in the first K most similar objects to the query object. The value of K is set to be double the number of objects in the same class. In our experiment, we used $K = 10$. Table 3 gives the actual numbers of returned objects from the same class as the queries and the overall retrieval accuracy for the three approaches. Note that while MVM and DTW have similar performance as in the 1–NN experiment, the performance of LCSS drops dramatically in retrieval experiments showing the inability of LCSS to find the next-nearest neighbors. This results from the fact that LCSS requires a hard threshold that determines when two sequence elements are regarded as similar.

Table 2
Results of 1–NN classification experiments

MVM	LCSS	DTW
90%	70%	30%

Table 3
Results of the retrieval experiments

Query no.	MVM	LCSS	DTW
1	5	1	4
2	5	1	2
3	5	4	2
4	5	4	0
5	5	5	2
6	4	2	0
7	5	1	3
8	5	3	1
9	4	1	0
10	3	1	1
Overall accuracy	90%	46%	30%

In both experiments, the proposed MVM method delivers significantly better rates than LCSS and DTW. Observe that the retrieval rates obtained by DTW are extremely low. This is expected because DTW must put all elements in correspondence, while the query parts correspond only to parts of the whole shapes. This fact is illustrated in Fig. 4(b). The width of the correspondence window *winWidth* which constraints the elasticity of MVM was set to five in all our experiments.

One important reason for MVM to achieve its superior performance is its ability to automatically figure out the starting and ending points of the path in the graph, which corresponds to the starting and ending points of a query part on the target sequence. Even though not necessary, a carefully selected query part can surely further improve MVM's performance. As demonstrated in Fig. 7, there are mismatches for query parts 1, 3, and 4. However, after extending these three parts and making them more discriminative for their own class, we obtain perfect results for them, as Fig. 8 shows. The lengths of the query parts in Fig. 8 are 70, 35, 57 points, respectively.

We performed all the experiments on a PC with a 2.2 GHZ 64bit CPU and 1 GB RAM. The running time of MVM is acceptable. It took less than 1 minute to finish the similarity retrieval with 10 query parts against 350 targets. Note that the programming language we used is Matlab, which is notoriously slow with loop operations, the calculation efficiency should be improved dramatically if we choose some other high performance language such as C or C++.

5. Conclusions

The proposed new method for partial shape matching, called MVM, performs the following tasks:

1. automatically determines a subsequence of the target sequence best matching the query sequence;

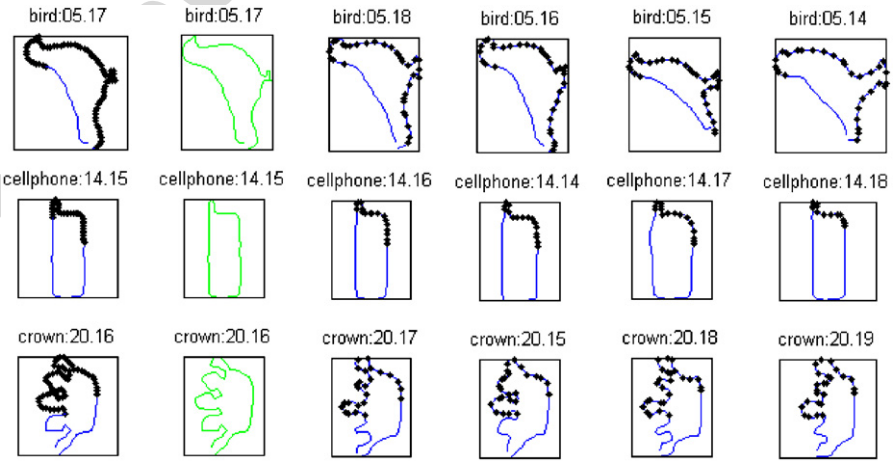

Fig. 8. Improved results of MVM with extended query parts.

2. automatically skips outliers that are present in the target sequence;
3. computes the translation or scale of corresponding values that minimizes the statistical variance of dissimilarities of corresponding elements.

Our experimental results demonstrate the benefits of MVM subsequence matching applied to partial shape matching. The reported experiments show that this method is able to perform partial shape matching effectively and significantly outperforms DTW and LCSS in terms of retrieval accuracy. By mapping the problem of elastic matching of sequences to the problem of finding a cheapest path in a DAG, we provide an efficient algorithm to compute the shape similarity.

Acknowledgments

This work was supported in part by NSF under Grant nos. IIS-0237921 and IIS-0534929, by DOE Grant no. DE-FG52-06NA27508, and by NIH under Grant no. R01MH68066-04 (funded by NIMH, NINDS, and NIA).

References

[1] D.D. Hoffman, W.A. Richards, Parts of recognition, Cognition 18 (1984) 65–96.
[2] D.D. Hoffman, M. Singh, Salience of visual parts, Cognition 63 (1997) 29–78.
[3] L.J. Latecki, A. Gross, R. Melter (Eds.), Special issue on shape representation and dissimilarity for image databases, Pattern Recognition 35(1) (2002).
[4] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, IEEE PAMI 24 (2002) 509–522.
[5] C. Grigorescu, N. Petkov, Distance sets for shape filters and shape recognition, IEEE Trans. Image Process. 12 (9) (2003).
[6] E. Saber, Y. Xu, A.M. Tekalp, Partial shape representation by sub-matrix matching for partial matching guided image labeling, Pattern Recogntion 38 (2005) 1560–1573.
[7] R. Veltkamp, M. Tanase, Part-Based Shape Retrieval, ACM Multimedia, Singapore, 2005.
[8] I. Biederman, Human image understanding: recent research and a theory, CVGIP 32 (1985) 29–73.
[9] T. Binford, Visual perception by computer, IEEE Conference on Systems and Control, 1971.
[10] R. Brooks, Symbolic reasoning among 3D models and 2D images, Artif. Intell. 17 (1981) 285–348.
[11] A. Pentland, Recognition by parts, Proc. ICCV 23 (1987) 612–620.
[12] L.J. Latecki, V. Megalooikonomou, Q. Wang, R. Lakaemper, C.A. Ratanamahatana, E. Keogh, Partial elastic matching of time series, in: Proceedings of IEEE International Conference on Data Mining (ICDM05), Houston, TX, USA, November 2005, pp. 701–704.
[13] L.J. Latecki, R. Lakämper, U. Eckhardt, Shape descriptors for non-rigid shapes with a single closed contour, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Hilton Head Island, South Carolina, June 2000, pp. 424–429.
[14] L.J. Latecki, R. Lakaemper, Shape similarity measure based on correspondence of visual parts, IEEE Trans. Pattern Anal. Mach. Intell. 22 (10) (2000) 1185–1190.
[15] B. Chiu, E. Keogh, S. Lonardi, Probabilistic discovery of sequences motifs, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, 2003.
[16] E. Keogh, S. Lonardi, C. Ratanamahatana, Towards parameter-free data mining, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, 2004.
[17] F. Höppner. Discovery of temporal patterns. Learning rules about the qualitative behavior of sequences, in: Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases, Freiburg, 2001, pp. 192–203.
[18] D. Rafiei, On similarity-based queries for sequences data, in: Proceedings of the International Conference on Data Engineering, Sydney, 1999, pp. 410–417.
[19] J. Aach, G. Church, Aligning gene expression sequences with time warping algorithms, Bioinformatics 7 (2001) 495–508.
[20] V. Megalooikonomou, Q. Wang, G. Li, C. Faloutsos, A multiresolution symbolic representation of time series, in: Proceedings of IEEE International Conference on Data Engineering (ICDE05), 2005, Tokyo, pp. 668–679.
[21] H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, IEEE Trans. Acoust. Speech Signal Process. 26 (1) (1978) 43–49.
[22] D. Berndt, J. Clifford, Using dynamic time warping to find patterns in sequences, in: Proceedings of AAAI-94 Workshop on Knowledge Discovery and Databases, 1994, pp. 229–248.
[23] G. Kollios, M. Vlachos, D. Gunopoulos, Discovering similar multidimensional trajectories, in: Proceedings of the International Conference on Data Engineering, 2002, San Jose, pp. 673–684.
[24] S. Chu, E. Keogh, D. Hart, M. Pazzani, Iterative deepening dynamic time warping for sequences, in: Proceedings of SIAM International Conference on Data Mining, 2002.
[25] B. Yi, K. Jagadish, C. Faloutsos, Efficient retrieval of similar time sequences under time warping, in: Proceedings of the International Conference on Data Engineering, 1998, pp. 23–27.
[26] C.A. Ratanamahatana, E. Keogh, Everything you know about dynamic time warping is wrong, in: Workshop on Mining Temporal and Sequential Data, Seattle, 2004.
[27] G. Das, D. Gunopoulos, H. Mannila, Finding similar sequences, in: Proceedings of the 1st PKDD Symposium, 2003, pp. 88–100.
[28] M. Vlachos, M. Hadjieleftheriou, D. Gunopoulos, E. Keogh, Indexing multi-dimensional time-series with support for multiple distance measures, in: Proceedings of ACM SIGKDD, 2003, Washington, pp. 216–225.
[30] F. Mokhtarian, M. Bober, Curvature Scale Space Representation: Theory, Applications and MPEG-7 Standardization, Kluwer Academic, Dordrecht, 2003.

About the Author—LONGIN JAN LATECKI is the winner of the Pattern Recognition Society Award together with Azriel Rosenfeld for the best article published in Pattern Recognition in 1998. He received the main annual award from the German Society for Pattern Recognition (DAGM), the 2000 Olympus Prize. He is an Editorial Board Member of Pattern Recognition. He co-chairs the IS&T/SPIE annual conference series on Vision Geometry. He published over 100 research papers and books. He is an associate professor of computer science at Temple University in Philadelphia.
He received a PhD in Computer Science from the Hamburg University in 1992. His main research areas are shape representation and similarity, robot mapping, video analysis, and digital geometry and topology.

About the Author—VASILEIOS MEGALOOIKONOMOU received his B.S. in Computer Engineering and Informatics from the University of Patras, Greece in 1991, and his M.S. and Ph.D. in Computer Science from the University of Maryland, Baltimore County in 1995 and 1997, respectively. He is currently an Associate Professor of Computer and Information Sciences and Director of the Data Engineering Laboratory at Temple University. He received a CAREER award from the National Science Foundation in 2003. His main research interests include data mining, data compression, multimedia database systems, pattern recognition, and biomedical informatics.

About the Author—QIANG WANG is currently a postdoctoral research fellow at the Fox Chase Cancer Center. He received his B.Sc. and M.Sc. degrees from Sichuan University (China) and a Ph.D. degree in Computer Science from Temple University. His major research areas are data mining, machine learning, bioinformatics, and computational biology. He has published 15 research papers in various journals and international conferences.

About the Author—DEGUANG YU is a Ph.D. student in the CIS Department at Temple University. His research interests include image processing and shape recognition.