Flexible Syntactic Matching of Curves

Yoram Gdalyahu and Daphna Weinshall

Institute of Computer Science, The Hebrew University,
91904 Jerusalem, Israel.
email:{yoram,daphna}@cs.huji.ac.il

Abstract. We present a flexible curve matching algorithm which performs qualitative matching between curves that are only weakly similar. While for model based recognition it is sufficient to determine if two curves are identical or not, for image database organization a continuous similarity measure, which indicates the amount of similarity between the curves, is needed. We demonstrate how flexible matching can serve to define a suitable measure. Extensive experiments are described, using real images of 3D objects. Occluding contours are matched under partial occlusion and change of viewpoint, and even when the two objects are different (such as the two side views of a horse and a cow). Using the resulting similarity measure between images, automatic hierarchical clustering of an image database is also shown, which faithfully capture the real structure in the data.

1 Introduction

Contour matching is an important problem in computer vision with a variety of applications, including model based recognition, depth from stereo and tracking. In these applications the two matched curves are usually very similar. For example, a typical application of curve matching to model based recognition would be to decide whether a model curve and an image curve are the same, up to an image transformation (e.g., translation, rotation and scale) and some permitted level of noise.

In this paper we are primarily interested in the case where the similarity between the two curves is weak. This is the situation, for example, when a recognition system is equipped with prototype shapes instead of specific exemplars. In this case the goal is to classify a given shape as belonging to a certain family, which is represented by the prototype. Another example (demonstrated in section 3) is the organization of an image database in a hierarchy of shape categories.

We use flexible curve matching to relate between feature points that are extracted on the boundaries of objects. We do not have a precise definition of what a reasonable matching between weakly similar curves is. As an illustrative example, consider two curves describing the shape of two different mammals: possibly we would like to see their limbs and head correspondingly matched. The matched pairs of points are then aligned using an optimal similarity transformation. From the residual distances between corresponding features we compute a robust dissimilarity measure between silhouettes.

To put our method in the context of existing curve matching methods, we first distinguish between dense matching and feature matching. Dense matching is usually formulated as a parameterization problem, with a cost function to be minimized. The cost might be defined as "elastic energy" needed to transform one curve to the other [4,7], but other alternatives also exist [2,10,9]. The main drawbacks of these methods are their high computational complexity (which is reduced significantly if only key points are matched), and the fact that none of them is invariant under both rotation and scaling. Computation of elastic energy (which is defined in terms of curvature) also requires an accurate evaluation of second order derivatives.

Feature matching methods may be divided into three groups: proximity matching, spread primitive matching, and syntactic matching. The idea behind proximity matching methods is to search for the best matching while permitting the rotation, translation and scaling (to be called alignment transformation) of each curve, such that the distances between matched key points are minimized [13,3,25]. The method is rather slow, and if scaling is permitted an erroneous shrinking of one feature set may result, followed by the matching of this set with a small number of features from the other set. One may avoid these problems by excluding many-to-one matches and by using the order of points, but then the method becomes syntactic (see below). As an alternative to the alignment transformation, features may be mapped to an intrinsic invariant coordinate frame [17,20,21]; the drawback of this approach is that it is global, the entire curve is needed to compute the mapping.

Features can be used to divide the curves into shape elements, or primitives. If a single curve is decomposed into shape primitives, it is reasonable to constrain the matching algorithm to preserve their order (see below). In the absence of any ordering information (like in stereo matching of many small fragments of curves), the matching algorithm may be called "spread primitive matching". In this category we find algorithms that seek isomorphism between attributed relational graphs [5,15,8], and algorithms that look for the largest set of mutually compatible matches. Here, compatibility means an agreement on the induced coordinate transformation, and a few techniques exist to find the largest set of mutually compatible matches (e.g., by constructing an association graph and searching for the maximal clique [14]).

For our purpose of matching complex outlines, it is advantageous to use the natural order of primitives and there is no need to solve the more general problem, which requires additional computational cost. Note that isomorphism of attributed relational graphs is found by different relaxation methods (sometimes called "relaxation labeling"), and they depend on successful choice of initial conditions. Scale and rotation invariance is achieved in these methods by using invariant relations, which suffer from the same drawbacks as invariant attributes (see below).

A syntactical representation of a curve is an *ordered* list of shape elements, having attributes like length, orientation, bending angle etc. Hence, many syntactical matching methods are inspired by efficient and well known string comparison algorithms, which use edit operations (substitution, deletion and insertion) to transform one string to the other [26, 18, 12]. The pattern recognition problem is different from the string matching problem in two major aspects, however: first, in pattern recognition invariance to certain geometrical transformations is desired; second, a resolution degradation (or smoothing) may create a completely different list of elements in the syntactical representation.

There are no syntactic algorithms available which satisfactorily solve both of these problems. If invariant attributes are used, the first problem is immediately addressed, but then the resolution problem either remains unsolved [1, 11, 16] or it is addressed by constructing for each curve a cascade of representations at different scales [24]. Moreover, invariant attributes are either non-local (e.g., length that is measured in units of the total curve length), or they are non-interruptible (see discussion in section 2.5). Using variant attributes is less efficient, but provides the possibility to define a merge operator which can handle noise [19, 22, 23], and might be useful (if correctly defined) in handling resolution change. However, the methods using variant attributes could not ensure rotation and scale invariance.

In this paper we present a local method which can cope both with occlusion and with image similarity transformations, and yet uses variant attributes that make it possible to cope with true scale (resolution) changes. The algorithm is presented in section 2. We are primarily concerned with the amount of flexibility that our method achieves, since we aim to apply it to weakly similar curves. Section 3 shows extensive experiments with real images, where excellent matching is obtained between weakly similar shapes. We demonstrate silhouette matching under partial occlusion, under substantial change of viewpoint, and even when the occluding contours describe different (but related) objects, like two different cars or mammals. Our method is efficient and fast, taking only a few seconds to match two curves.

2 The proposed method

The occluding contours of objects are extracted in a pre-processing stage. In the examples shown below, objects appear on a dark background, and segmentation is done using a standard clustering algorithm. The occluding contour is then approximated automatically by a polygon whose vertices are either points of extreme curvature, or points which are added to refine the approximation.

The polygon is our syntactic representation: the primitives are line segments, and the attributes are length and absolute orientation. The number of segments depends on the chosen scale and the shape of the contour, but typically is around 50. Coarser scale descriptions may be obtained using merge operators.

Our algorithm uses a variant of the edit algorithm, dynamic programming and heuristic search. We define a novel similarity measure between primitives, a novel merge operation, and introduce a penalty for interrupting a contour (in addition to the regular deletion/insertion penalty). The result is an algorithm that is robust, invariant under scaling and rigid transformations, suitable for partial matching, and fast.

2.1 Similarity between primitives

The similarity between two line segments is a function of their orientation and length attributes (θ, ℓ) and (θ', ℓ') respectively. Since global rotation and scale are arbitrary, the similarity between the segments cannot be determined independently, and at least one other pair of segments is needed for meaningful comparison. We pick two reference segments, (θ_0, ℓ_0) and (θ'_0, ℓ'_0), which define the reference rotation $\theta_0 - \theta'_0$ and reference scale ℓ_0/ℓ'_0. The similarity between (θ, ℓ) and (θ', ℓ') is determined with respect to these reference segments.

Local and scale-invariant matching methods usually replace ℓ by the normalized length ℓ/ℓ_0, defining the scale similarity function as $S_\ell(\ell/\ell_0, \ell'/\ell'_0)$. For example, the ratio between normalized lengths $\frac{\ell/\ell_0}{\ell'/\ell'_0}$ is used in [16,15] (with global normalization the difference $|\ell/L - \ell'/L'|$ can be used [24,22]). The ratio between normalized lengths may be viewed as the ratio between the relative scale $c = \ell/\ell'$ and the reference relative scale $c_0 = \ell_0/\ell'_0$.

We define a different measure of similarity between the two scales. Instead of dividing c by c_0, we map each scale factor to a direction in the (ℓ, ℓ')-plane, and we measure the angle between the two directions. The cosine of twice this angle is our measure of scale similarity (figure 1). This measure is numerically stable. It is not sensitive to small scale changes, nor does it diverge when c_0 is small. It is measured in intrinsic units between -1 and 1, in contrast with the scale ratio which is not bounded. The measure is symmetric, so that the labeling of the contours as "first" and "second" is arbitrary.

Let δ be the angle between the vectors $[\ell, \ell']$ and $[\ell_0, \ell'_0]$. Our scale similarity measure can be expressed as:

$$S_\ell(\ell, \ell' | \ell_0, \ell'_0) = \cos 2\delta = \frac{4cc_0 + (c^2 - 1)(c_0^2 - 1)}{(c^2 + 1)(c_0^2 + 1)} \qquad (1)$$

The factor of 2 in the cosine argument simplifies the expression, lets it take values in the whole $[-1, 1]$ range (since $0 < \delta < \pi/2$), and reduces the sensitivity at the extreme cases (similar or unsimilar scales). S_ℓ thus depends explicitly on the scale values c and c_0 rather than on their ratio, hence it cannot be computed from the invariant attributes ℓ/ℓ_0 and ℓ'/ℓ'_0. The arbitrariness of labeling can be readily verified, since $S_\ell(c, c_0) = S_\ell(c^{-1}, c_0^{-1})$. Note that this property is not shared by the quantity $|c - c_0|$, which is not a good measure also because it depends explicitly on ℓ_0 and ℓ'_0 rather than on their ratio.

We are familiar with only one other definition of a symmetric, bounded and scale invariant measure for segment length similarity [15]. The matching algorithm there is not syntactic and very different from ours. In addition, there is

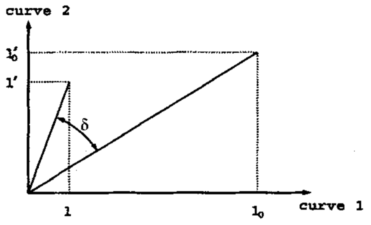


Fig. 1. Length similarity is measured by comparing the reference relative scale $c_0 = \ell_0/\ell'_0$ with the current relative scale $c = \ell/\ell'$. Each scale is mapped to a direction in the plane, and similarity is defined as $\cos(2\delta)$. This value is bounded by -1 and 1.

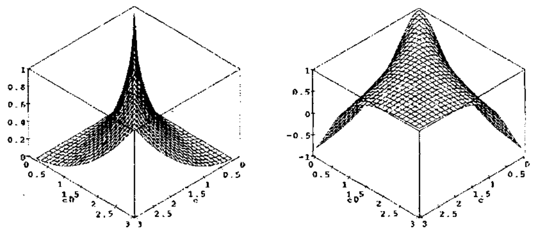


Fig. 2. Scale similarity: the scale $c = \ell/\ell'$ is compared with the reference scale $c_0 = \ell_0/\ell'_0$. Left: The binary relation used by Li [15] to measure scale similarity is $\exp(-|log(c/c0)|/\sigma)$ with $\sigma = 0.5$. Right: Our measure function (equation 1) is not sensitive to small scale changes, since it is flat near the line $c = c_0$.

an important qualitative difference between the two definitions (see figure 2), where our measure is more suitable for flexible matching.

We turn now to the orientation similarity S_θ between two line segments whose attributes are θ and θ' respectively. The relative orientation between them is measured in the trigonometric direction (denoted $\theta \rightarrow \theta'$) and compared with the reference rotation ($\theta_0 \rightarrow \theta'_0$):

$$S_\theta(\theta, \theta' | \theta_0, \theta'_0) = \cos\left[(\theta \rightarrow \theta') - (\theta_0 \rightarrow \theta'_0)\right]$$

As with the scale similarity measure, the use of the cosine introduces non-linearity; we are not interested in fine similarity measurement when the two segments are close to being parallel or anti parallel. Our matching algorithm is designed to be flexible, in order to match curves that are only weakly similar; hence we want to encourage segment matching even if there is a small discrepancy between their orientations. Similarly, the degree of dissimilarity between two nearly opposite directions should not depend too much on the exact angle between them. On the other hand, the point of transition from acute to obtuse angle between the two orientations seems to have a significant effect on the degree of similarity, and therefore the derivative of S_θ is maximal when the line segments are perpendicular.

We note that a similar *linear* measure has been widely used by others [22, 23, 16, 8]. The non linear measure used by [15] differs from ours in exactly the same way as discussed above concerning length.

Finally, the combined similarity measure is defined as the weighted sum:

$$S = w_1 S_\ell + S_\theta$$

The weight w_1 (which equals 1 in all our experiments) controls the decoupling of scale and orientation similarity. In [19] a coupled measure is used: the segments are superimposed at one end, and their dissimilarity is proportional to the distance between their other ends. However, this measure is too complicated for our case, and it has the additional drawback that it is sensitive to the arbitrary reference scale and orientation (in the character recognition task of [19] it is assumed that characters are in the same scale and properly aligned).

2.2 Syntactic operations: gaps and merges

The goal of a classical string edit algorithm is to find a sequence of elementary edit operations, which transform one string into the other at a minimal cost. The elementary operations are substitution, deletion and insertion. Converting the algorithm to the domain of pattern recognition, substitution is interpreted as matching two shape primitives, and the symbol substitution cost is replaced by the dissimilarity between matched primitives. In our scheme we use a similarity function (instead of dissimilarity) and maximize the transformation gain (instead of minimizing cost). The similarity measure was discussed in the previous section. We now discuss the insertion/deletion gain, and the novel merge operation.
Gap opening: In string matching, the null string λ serves to define deletion and insertion operations, $a \rightarrow \lambda$ and $\lambda \rightarrow a$ respectively, where a is a string of length 1. In our case, a is a line segment and λ is interpreted as a "gap element". We define, customarily, the same gain for both operations, making the insertion of a into one sequence equivalent to the deletion of it from the other. This means that either the segment a is matched with some a' on the other curve, or the other curve is interrupted and a gap element λ is inserted into it, to be matched with a.

All the syntactical shape matching algorithms that we are familiar with make use of deletions and insertions as purely local operations, like in classical string matching. That is, the cost of inserting a sequence of gaps into a contour is equal to the cost of spreading the same number of gap elements in different places along the contour. We distinguish the two cases, since the first typically arises from occlusion or partial matching, while the second arises typically from curve dissimilarity. In order to make the distinction we adopt a technique frequently used in protein sequence comparison, namely, we assign a cost to the contour interruption itself, in addition to the deletion/insertion gain.

The gain from interrupting a contour and inserting ξ connected gap elements into it (that are matched with ξ consecutive segments on the other curve) is taken to be $w_2 \cdot \xi - w_3$. That is, a gain of w_2 for every individual match with a gap element, and a penalty of w_3 for the single interruption. This pre-defined quantity competes with continuous matching of the ξ segments (by substitutions), whose gain is lower only if the matching is poor. Note that $w_2 \cdot \xi - w_3$ is scale independent, since w_2 is a constant that does not depend on the length of the segment which is matched with the gap.

In all our experiments we used $w_2 = 0.8$ and $w_3 = 8.0$. (These values were determined in an ad-hoc fashion, and not by systematic optimization tuning, which is left for future research.) These numbers make it possible to match a gap with a long sequence of segments, as required when curves are partially occluded. On the other hand, isolated gaps are discouraged due to the high interruption cost. The parameters were so chosen because this mechanism is not intended to handle noise. Noise is better handled by the merging mechanism that we describe next.

Segment merging: One advantage of using variant attributes (length and absolute orientation) is that segment merging becomes possible. We use segment merging as the syntactic homologue of curve smoothing, accomplishing noise reduction by local resolution degradation. Segment merging, if defined correctly, should simplify a contour representatirn by changing its scale from fine to coarse.

A similar approach was taken in [24], but their use of invariant attributes made it impossible to realize the merge operator as an operation on attributes. Specifically, there is no analytical relation between the attributes being merged to the attributes of the equivalent primitive. Instead, a cascade of alternative representations is used, each one obtained by a different Gaussian smoothing of the two dimensional curve; a primitive sequence is replaced by its "ancestor" in the scale space description[1].

Merging was defined as an operation on attributes by [22], who also applied the technique to Chinese character recognition [23]. Their algorithm suffers from some drawbacks concerning invariance and locality[2]; below we concentrate on their merging mechanism, and compare it to our own.

Assume that the two line segments characterized by (ℓ_1, θ_1) and (ℓ_2, θ_2) are to be merged into one segment (ℓ, θ). In [22] $\ell = \ell_1 + \ell_2$, and θ is the weighted average between θ_1 and θ_2, with weights $\ell_1/(\ell_1+\ell_2)$ and $\ell_2/(\ell_1+\ell_2)$, and with the necessary cyclic corrections[3]. Usually, the polygonal shape that is obtained using this simple ad-hoc merging scheme *cannot* approximate the smoothed contour very well. Satisfactory noise reduction is achieved in one of the two extreme cases: either one segment is dominant (much longer than the other one). or the two segments have similar orientation. If two or more segments having large variance are merged, the resulting curve may not bear a direct relation to the shape of the original curve (figure 3). Hence, by performing segment merging on

[1] The primitive elements used in [24] are convex and concave fragments, which are bounded by inflection points. The attributes are the fragment length divided by total curve length (a non-local attribute), and the accumulated tangent angle along the fragment (a non-interruptible attribute). The algorithm cannot handle occlusions or partial distortions, and massive preprocessing is required to prepare the cascade of syntactical representations for each curve, with consistent fragment hierarchy.

[2] The primitives used by [22] are line segments, the attributes are relative length (with respect to the total length) and absolute orientation (with respect to the first segment). The relative length is, of course, a non-local attribute, and in addition the algorithm uses the total number of segments, meaning that the method cannot handle occlusions. The problem of attribute variance due to a possible rotation transformation remains in fact unsolved. The authors assume that the identity of the first segments is known. They comment that if this information is missing, one may try to hypothesize an initial match by labeling the segment that is near the most salient feature as segment number one.

[3] For example, an equal weight average between 0.9π and -0.9π is π and not zero.

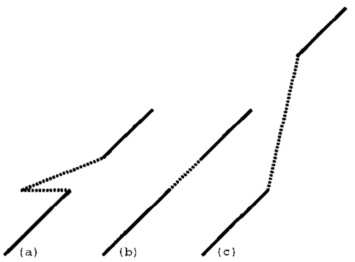


Fig. 3. Comparison between merging rules: (a) A polygonal approximation of a curve, with two dotted segments which are to be merged. (b) Merging result according to our scheme. A coarser approximation is obtained. (c) Merging according to Tsai and Yu [22]. The new polygon is not a good approximation.

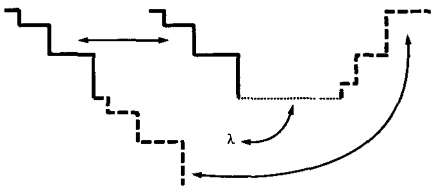


Fig. 4. When the primitive attribute is measured relative to a preceding primitive, interrupting the sequence creates problems. Here, for example, the orientation information is lost when the dotted segment is matched with a gap element. As a result, the two contours may be matched almost perfectly to each other and considered as very similar.

a fine scale polygonal approximation, one typically does not obtain an acceptable coarse approximation of the shape.

We define a different merging rule: two adjacent line segments are replaced by the line connecting their two furthest endpoints. If the line segments are viewed as vectors oriented in the direction of propagation along the contour, then the merging operation of any number of segments is simply their vectorial sum.

Compare the polygonal approximation after merging with the polygon that would have been obtained if the curve was first smoothed and then approximated. The two polygons are not identical, since smoothing may cause displacement of features (vertices). However, a displaced vertex cannot be too far from a feature of the finest scale; the location error caused by "freezing" the feature points is clearly bounded by the length of the longest fragment in the initial (finest scale) representation.

To ensure good multi scaled feature matching our sub-optimal polygonal approximation is sufficient, and the expensive generation of the multi scale cascade is not necessary. Instead, the attributes of the coarse scale representation may be computed directly from the attributes of the finer scale.

2.3 The optimization algorithm

The optimization problem is constructed of two parts:

- finding the two reference segments $a_0 = (\ell_0, \theta_0)$ on contour A and $a'_0 = (\ell'_0, \theta'_0)$ on contour A'.
- finding the syntactic operations which maximize the matching gain.

The first part solves for the global alignment between the two curves, since matching the features a_0 and a'_0 uniquely determines the relative global rotation and scale of the curves. If A (A') is constructed of N (N') segments, then we assume that the optimal alignment transformation is approximated well by at least one of the $N \cdot N'$ possible selections of a_0 and a'_0.

We combine the alignment step and the matching step into one optimization problem. An optimal sequence of edit operations is found by standard dynamic programming, while at the same time a good reference pair is found by heuristic search.

Best syntactic operations: Let us assume for the moment that a reference pair of segments is given. Thus the optimal edit transformation between the sequence $\{a_0, a_1, \ldots, a_{N-1}\}$ and the sequence $\{a'_0, a'_1, \ldots, a'_{N'-1}\}$ can be found by dynamic programming. A virtual array $R_{N \times N'}$ is assigned (it is virtual since only a fraction of it really exists in memory), where the entry $R[i,j]$ holds the maximal gain that can be achieved when the first i elements of A are matched with the first j elements of A'. The updating scheme of R is "block completion", meaning that the upper left block of size $\mu \times \nu$ holds the evaluated elements, and in the next updating step the block is extended to size $(\mu + 1) \times \nu$ or to $\mu \times (\nu + 1)$. Every single entry is updated according to the following rule:

$$R[i,j] = \max\{r_1, r_2, r_3\} \tag{2}$$

where
$$\begin{aligned} r_1 &= \max_{\alpha,\beta \in \Omega} \{R[\alpha,\beta] + S(\overline{\alpha i}, \overline{\beta j})\} \\ r_2 &= \max_{0<\alpha<i} \{R[\alpha,j] + w_2 \cdot (i-\alpha) - w_3\} \\ r_3 &= \max_{0<\beta<j} \{R[i,\beta] + w_2 \cdot (j-\beta) - w_3\} \end{aligned}$$

$\overline{xy}$ denotes the vectorial sum of the segments $(x+1), \ldots, y$.

Unlike in the "classical" editing algorithm, the term r_1 is computed over a domain Ω, generalizing the simple substitution operation to the substitution of merged segments. If $K - 1$ is the maximal number of segments that may be merged together, then

$$\Omega = \{\alpha, \beta \mid 0 < \alpha < i,\ 0 < \beta < j,\ (i-\alpha) + (j-\beta) \leq K\}$$

and the computation of r_1 involves $K(K-1)/2$ evaluations of alternatives merges.

The single element deletion operation is generalized to the deletion of ξ consecutive elements, which is associated with an interruption penalty (w_3) and pre-insured gain ($w_2\xi$). We keep one index (α_0) for each column j, such that $r_2 = R[\alpha_0, j] + w_2(i - \alpha_0) - w_3$. The initial value of α_0 is 1, and after entry $R[i,j]$ has been updated, the value of α_0 should be set to i if $R[i,j] - R[\alpha_0, j] \geq w_2(i - \alpha_0)$. A similar procedure applies to the computation of r_3. Hence both the computation of r_2 and r_3 have complexity $O(1)$.

Alignment of curves: The updating scheme of the virtual array $R_{N \times N'}$ is therefore completely defined, and only the last $\lfloor K/2 \rfloor$ rows and columns of R need to be stored in memory. We next show how to determine the pair of reference segments, a_0 and a'_0. A naive approach would be to try all the NN' possible selections of reference pairs (candidate alignments). One would then compute the matching gain for each candidate, and select the best result at the end. To avoid this massive computation we use two complementary strategies: heuristic search and statistical filtering.

The heuristic search is the familiar A^* algorithm, where the updating process of all the arrays R_l ($l \leq NN'$) becomes competitive. We define a potential function ("optimistic estimation") $f(R_l)$, which decreases monotonically during the update, and which gives an upper bound on R_l's score. The potential $f(R_l)$ is re-evaluated (decreased) after every block completion, which in turn is performed for the array with the largest potential. The process terminates when all the potentials are below the best score that has been already achieved, and an optimal solution is guaranteed, since $f(R_l)$ is based on optimistic estimation.

The potential $f(R)$ is defined as the maximum over entry potentials $f_{i,j}(R)$, where i (j) belongs to the last $\lfloor K/2 \rfloor$ rows (columns). The entry potential $f_{i,j}(R)$ is defined as[4]:

$$\begin{aligned} f_{i,j}(R) &= R[i,j] + \epsilon(\eta - 1) + w_2(\zeta - \eta) \\ \eta &= \min(N - i, N' - j) \\ \zeta &= \max(N - i, N' - j) \end{aligned}$$

and ϵ denotes the maximal segment similarity value ($\epsilon = 1 + w_1 = 2$ in our case).

While the heuristic search is only effective at the later stages, the complementary filtering strategy is effective at the initial stages. Assume that we try to match the first few (say, 10) segments in each array. Acquiring high score after an occasional (wrong) match of about 10 segments must be rare, and on the other hand the number of feasible starting points is of the order of $\min(N, N')$. Hence, the majority of the arrays which are associated with high potentials (after few updeting steps) will agree on the same global transformation (rotation and scale). As a result, we can rely on the high potential arrays for a reliable estimation of these global parameters.

Our statistical filtering strategy is therefore the following: we perform 10 updating steps for all the arrays, without evaluating their potentials. Then we look for central tendency among the rotations associated with the "best" arrays, and if such tendency is found - we eliminate all the arrays that are associated with a very different rotation angle.

2.4 Outliers removal

Since wrong matches are unavoidable, we use two different techniques for outliers detection. The first one is iterative elimination: in every iteration the matched

[4] Here we assume the worst case scenario, where $[i,j]$ is in a gap. Hence it is possible to increase the gain by extending the gap first, leaving space for a diagonal path of length $\eta - 1$ (the longest diagonal path from $[i,j]$ to the edge of R). We also assume that $w_3 > (w_2 - \epsilon/2)\min(N, N')$, otherwise the pre-insured gain is always higher than any other gain. In our case, since $\epsilon = 2$, this relation holds for $w_2 < 1$ and any value of N and N'.

pairs that are most distant are eliminated, and the rest are re-aligned. We chose to eliminate 10% of the pairs at every iteration. The motivation is the following: features are matched when the local pieces of curve around them have similar shape; if after alignment they are also proximal, meaning that they agree with the global alignment, then the match is likely to be correct.

Another pruning technique can be used when three related images are available (and not only two). Assume that a feature point p on contour 1 is matched with the point p' on contour 2, and p' is matched with p'' on contour 3. If the matching between 1 and 3 supports the mapping between p and p'', then the involved correspondences ($p - p'$, $p' - p''$, $p - p''$) are accepted. Note that the order of matching is arbitrary.

2.5 Discussion and comparison to other methods

Available syntactic matching methods usually achieve scale and rotation invariance (if at all) by using invariant attributes. The drawback is that invariant attributes cannot be smoothed by merging, and they are either non local or non interruptible. For example, in [16] the orientation of a line segment is measured with respect to its successor, hence the opening of a gap between segments introduces ambiguity into the representation (see figure 4). In [11] the attributes which describe curve fragments are Fourier coefficients, in [1] it is a measure called "sphericity". Both are invariant attributes, but non-interruptible[5].

Moreover, it seems to be impossible to find operators on invariant attributes that are equivalent to smoothing in real space. Instead, a cascade of different scale representations must be used [24], where few fragments may be replaced by a single one which is their "ancestor" in a scale space description. This requires massive preprocessing, building a cascade of syntactical representations for each curve with consistent fragment hierarchy.

The benefit of using invariant attributes is efficiency. Yet our algorithm is invariant with respect to scaling, rotation and translation without relying on invariant attributes, and is still efficient, capable of comparing complex real image curves in a few seconds. Furthermore, a novel merging operation was defined, which accomplished curve simplification and helped in noise reduction and resolution change.

[5] The Fourier coefficients are normalized individually, which means that if every fragment undergoes a different rigid or scaling transformation, the representation remains unchanged. The sphericity representation behaves in the same way. The relative size and orientation information is preserved as long as the sequence is not interrupted, since overlapping fragments are used. Note that in spite of this property the algorithms are applied to partial matching in the framework of model based recognition, since the solution that preserves the correct relative size and orientation information between primitives remains a valid solution, and the danger of finding an undesired solution (as is demonstrated in figure 4) is small.

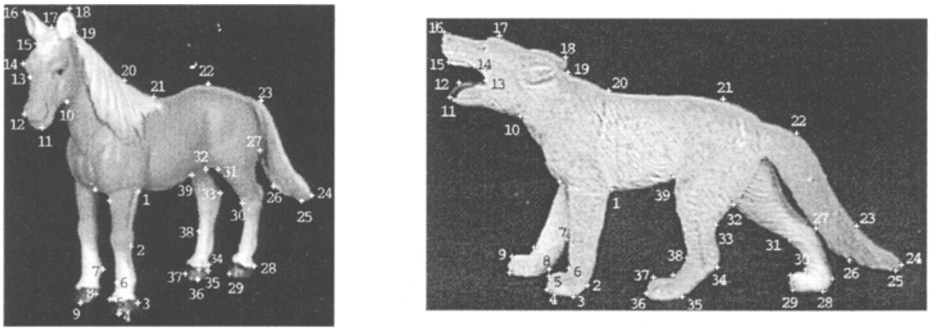


Fig. 5. Qualitative matching between toy models of a horse and a wolf. Note the correct correspondence between the feet of the wolf to those of the horse, and the correspondence between the tails. The results are shown without outliers pruning. In this example, all the features to which no number is attached had been merged; e.g. the segment 9-10 on the horse outline was matched with 3 segments on the wolf outline.

3 Results

In Section 3.1 we present a few image pairs and triplets together with the matching results. This is a direct evaluation of the algorithm, using a highly subjective notion of success (as there is no "correct" way to match weakly similar curves). In Section 3.2 we present an indirect objective examination using the matching of a few thousands image pairs.

3.1 Subjective investigation

Figure 5 shows two images of different objects. There is geometrical similarity between the two silhouettes, which has nothing to do with the semantic similarity between them. The geometrical similarity includes five approximately vertical swellings or lumps (which describe the four legs and the tail). In other words, there are many places where the two contours may be considered locally similar. This local similarity is captured by our matching algorithm.

The two occluding contours of the two mammals and the feature points were automatically extracted in the pre-processing stage. Corresponding points are marked in figure 5 by the same numbers. Hence the tails and feet are nicely matched, although the two shapes are only weakly similar. The same matching result is obtained under arbitrarily large rotation and scaling of one image relative to the other.

Figure 6 demonstrates the local nature of our algorithm, namely, that partial matching can be found when objects are occluded. Since our method does not require global image normalization, the difference in length between the silhouette outlines does not impede the essentially perfect matching of the common parts. Moreover, the common parts are not identical (note the distance between the front legs and the number of ears) due to a small difference in viewpoint; this also does not impede the performance of our algorithm.

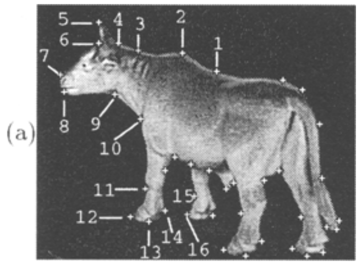
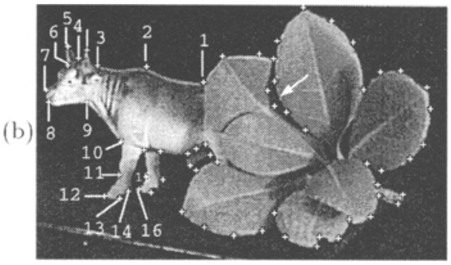


Fig. 6. Partial matching between three images. Points are mapped from (a) to (b) to (c) and back to (a). Only points which are mapped back to themselves are accepted (order is not important). The points on the tail in (a) are matched with the shadow (pointed by the arrow) in (b) , but matching (b) with (c) leaves the shadow unmatched. Hence the tail is not matched back to itself, and the correspondence with the shadow is rejected.



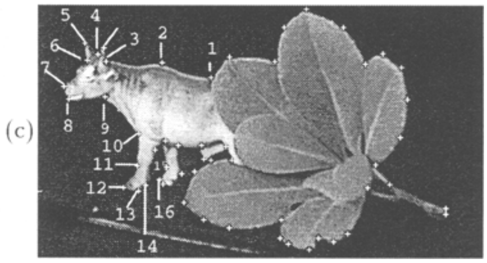


Figure 6 also demonstrates outliers pruning using three images. In image 6b there is a shadow between two of the leaves (pointed by the arrow), and as a result the outline penetrates inward. The feature points along the penetration are (mistakenly) matched with features along the tail in image 6a, since the two parts are locally very similar. However, we map the points of 6a to 6b, then to 6c and back to 6a. Only points which are mapped back to themselves are accepted as correct matches, which appear as common numbers in figure 6.

Figures 7 and 9 show the application of the algorithm to match images taken from very different points of view of different rigid objects. In figure 7 two different views of the same object are matched, and the method of iterative elimination of distances is demonstrated. Figure 9 shows matching between three different cars, subjected to both different orientations and to occlusion. Matching under a large viewpoint perturbation can be successful as long as the silhouettes remain similar enough. Note that preservation of shape under change of viewpoint is a quality that defines "canonical" or "stable" views. Stable images of 3D objects were proposed as the representative images in an appearance based approach to object representation [27].

The last example (figure 8) shows matching of human limbs at different body configurations.

We note again that all these examples were generated with the same values of parameters: $w_1 = 1$, $w_2 = 0.8$, $w_3 = 8.0$ and $K = 4$ (with the exception of figure 9, where $K = 5$). Each pairing assignment took only a few seconds (see next section).

3.2 Objective investigation

The test presented in this section is based on automatic matching of thousands of contours. The task was the partitioning of 90 images into hierarchical clus-

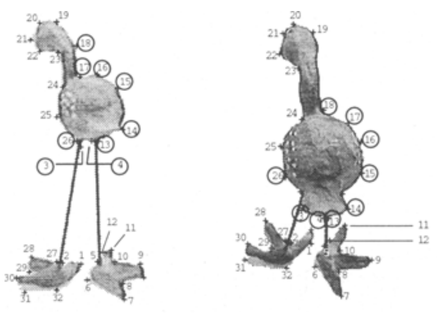


Fig. 7. Matching two views subjected to a large forthshortening effect. Rejected pairs (in circles) were detected in four iterations of eliminating the (10%) most distant pairs and re-aligning the others. 33 and 35 features were extracted on the two outlines; 32 pairs were initially matched, and 9 pairs were rejected (28%).

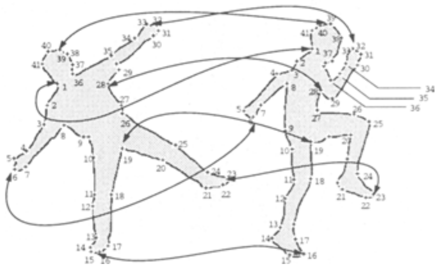


Fig. 8. Matching of human limbs at different body configurations. In this case the outlines were extracted with snakes rather than by gray level clustering (see acknowledgments). Original images are not shown.

ters. These were 90 images of 6 different objects (toy models of a cow, wolf, hippopotamus, two different cars, and a child). Each object contributed 15 images, taken from different points of view (in a sector range of 40° azimuth and 20° elevation).

The outlines were automatically extracted, and every two different contours were matched. The task involves matching 4005 image pairs, which took 6.5 hours on an INDY R4400 175Mhz workstation (5.8 seconds per match, on average).

Based on the matching results, a dissimilarity value was assigned to every pair of images, yielding a 90×90 dissimilarity matrix. It is beyond our scope here to describe in details how this value is defined. For our purpose it is sufficient to say that the dissimilarity value is computed from the residual distances between matched features. Hence, correct feature pairing is essential to achieve reliable dissimilarity estimation.

The dissimilarity matrix constituted the input to a newly developed clustering algorithm [6]. The algorithm has an additional scale parameter (called "temperature") which defines the level of specification. When this parameter is varied, the dendrogram shown in figure 10 is obtained. In the final level of classification (the lowest level in the hierarchy), the 90 images are grouped precisely according to their identity. Even the very similar car images are correctly separated at this level. More interesting is the hierarchical structure, which reflects our intuition regarding families of objects.

This kind of database structuring could not have emerged without the reliable estimation of the dissimilarities between weakly similar images. Hence the classification tree is an indirect and objective evidence to the quality of our matching method.

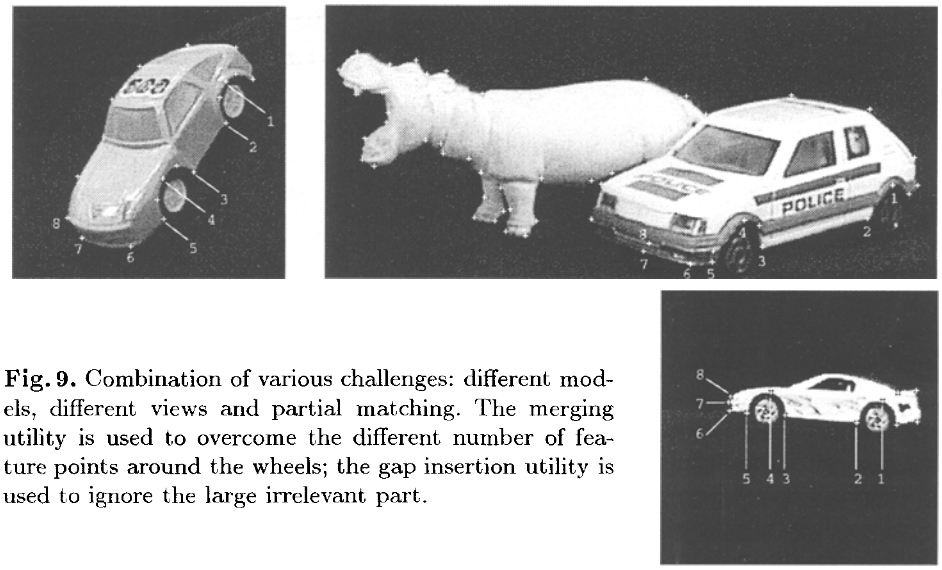


Fig. 9. Combination of various challenges: different models, different views and partial matching. The merging utility is used to overcome the different number of feature points around the wheels; the gap insertion utility is used to ignore the large irrelevant part.

4 Conclusions

This paper is concerned with a problem that is inherently ill posed, the matching of weakly similar curves. A simple heuristic was used, guided by the principle that matched features should lie on locally similar pieces of curve. Naturally, this principle cannot guarantee that results would agree with our own human intuition for "good" matching in specific examples, but our examples demonstrate that satisfactory and intuitive results are typically obtained. Note that "successful" matching depends on the application that the matching is used for. Our method is not suitable for recovering depth from stereo, but it is well suited for more qualitative tasks, such as the organization of image database, the selection of prototypical shapes, and image morphing for graphics or animation.

In order to achieve large flexibility we introduced a non linear measure of similarity between line segments, which was not sensitive to either very small or very large differences in their scale and orientation. Our specific choice of segment similarity, combined with a novel merging mechanism and an improved interruption operation, added up to a powerful and successful algorithm.

We demonstrated excellent results, matching similar curves under partial occlusion, matching similar curves where the curves depict the occluding contours of objects observed from different viewpoints, and matching different but related curves (like the silhouettes of different mammals or cars). Furthermore, we used the method to compare a range of curves, some of them very different, others rather similar. We then used the results to classify the data with an automatic hierarchical clustering algorithm, getting excellent results which faithfully captured the real structure in the data. This serves as indirect evidence to the quality of our matching algorithm.

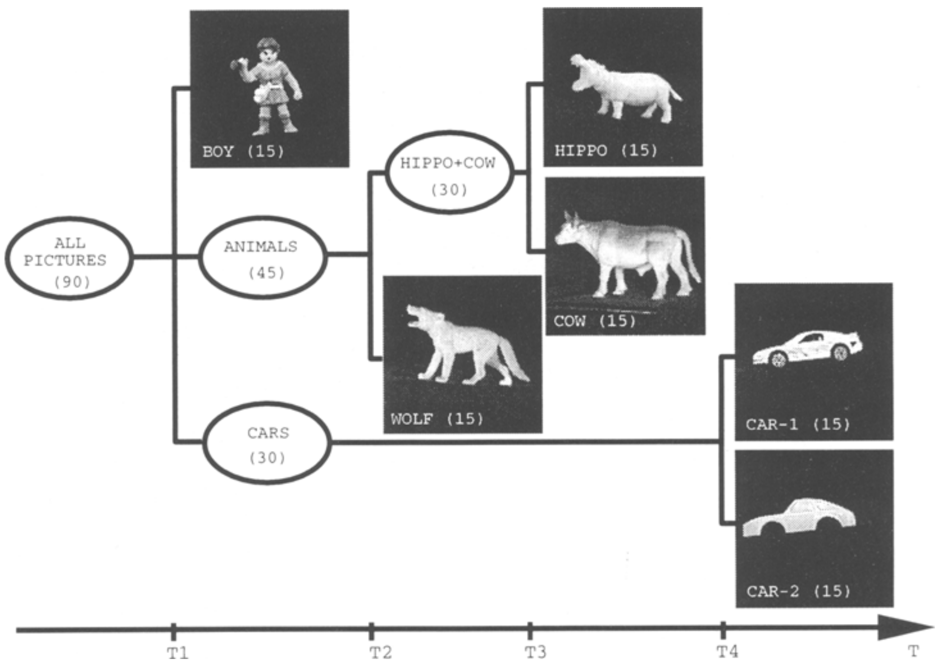


Fig. 10. The classification tree (dendrogram) obtained by hierarchical clustering algorithm, using the pairwise dissimilarity matrix of 90 images of 6 objects. The temperature axis (unscaled) determines the level of specification. At T1 three groups are identified, separating the pictures of the boy, animals and cars. At T2 and T3 the animal group is segmented into three sub groups, corresponding to the pictures of cow, wolf and hippo. Finally, at T4 the car group is segmented into two sub-groups, each containing images of a different car. The numerical values of the transitions are: T1=0.018, T2=0.029, T3=0.037 and T4=0.078. The final automatic classification is 100% correct, and the hierarchy reflects the true structure of the database.

Acknowledgments

We thank Marcello Blatt from the Weizmann institute for the clustering work, and Golan Yona from the Hebrew University for the discussions. Tyng-Luh Liu and Davi Geiger from New-York University are acknowledged for the data used to generate figure 8. This research is partialy founded by the Israeli ministery of science.

References

1. Ansari N., Delp E., "Partial shape recognition: a landmark based approach", PAMI 12, 470-489, 1990.
2. Arkin E., Paul Chew L., Huttenlocher D., Kedem K. and Mitchel J., "An efficiently computable metric for comparing polygonal shapes", PAMI 13, 209-216, 1991.

3. Ayach N. and Faugeras O., "HYPER: a new approach for the recognition and positioning of two dimensional objects", PAMI 8, 44-54, 1986.
4. Basri R., Costa L., Geiger D. and Jacobs D., "Determining the similarity of deformable shapes" *IEEE Workshop on Physics Based Modeling in Computer Vision*, 135-143, 1995.
5. Bhanu B. and Faugeras O., "Shape matching of two dimensional objects", PAMI 6, 137-155, 1984.
6. Blatt M., Wiseman S. and Domany E., "Data Clustering Using a Model Granular Magnet", Neural Computation 9, 1805-1842, 1997.
7. Brint A. and Brady M., "Stereo matching of curves", *Image and vision computing* 8, 50-56, 1990.
8. Christmas W., Kittler J. and Petrou M., "Structural matching in computer vision using probabilistic relaxation", PAMI 17, 749-764, 1995.
9. Del Bimbo A. and Pala P., "Visual image retrieval by elastic matching of user sketches", PAMI 19, 121-132, 1997.
10. Geiger D., Gupta A., Costa L. and Vlontzos J., "Dynamic programming for detecting, tracking and matching deformable contours", PAMI 17, 294-302, 1995.
11. Gorman J., Mitchell O. and Kuhl F., "Partial shape recognition using Dynamic programming", PAMI 10, 257-266, 1988.
12. Gregor J. and Thomason M., "Dynamic programming alignment of sequences representing cyclic patterns", PAMI 15, 129-135, 1993.
13. Huttenlocher D. and Ullman S., "Object recognition using alignment", Proc. ICCV (London), 102-111, 1987.
14. Koch M. and Kashyap R., "Using polygons to recognize and locate partially occluded objects", PAMI 9, 483-494, 1987.
15. Li S., "Matching: invariant to translations, rotations and scale changes", *Pattern Recognition* 25, 583-594, 1992.
16. Liu H. and Srinath M., "Partial shape classification using contour matching in distance transformation", PAMI 12, 1072-1079, 1990.
17. Lu C. and Dunham J., "Shape matching using polygon approximation and dynamic alignment", PRL 14, 945-949, 1993.
18. Marzal A. and Vidal E., "Computation of normalized edit distance and applications", PAMI 15, 926-932, 1993.
19. Rocha J. and Pavlidis T. "A shape analysis model with applications to a character recognition system", PAMI 16, 393-404, 1994.
20. Sclaroff S. and Pentland A., "Modal matching for correspondence and recognition", PAMI 17, 545-561, 1995.
21. Shapiro L. and Brady M., "Feature based correspondence: an eigenvector approach", *Image and vision computing* 10, 283-288, 1992.
22. Tsai W. and Yu S., "Attributed string matching with merging for shape recognition", PAMI 7, 453-462, 1985.
23. Tsay Y. and Tsai W., "Attributed string matching by split and merge for on-line chinese character recognition", PAMI 15, 180-185, 1993.
24. Ueda N. and Suzuki S., "Learning visual models from shape contours using multiscale covex/concave structure matching", PAMI 15, 337-352, 1993.
25. Umeyama S., "Parameterized point pattern matching and its application to recognition of object families", PAMI 15, 136-144, 1993.
26. Wang Y. and Pavlidis T., "Optimal correspondence of string subsequences", PAMI 12, 1080-1086, 1990.
27. Weinshall D. and Werman M., "On View Likelihood and Stability", PAMI 19, 97-108, 1997.