

Flexible Syntactic Matching of Curves and its Application to Automatic Hierarchical Classification of Silhouettes

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

Abstract

Curve matching is one instance of the fundamental correspondence problem. Our flexible algorithm is designed to match curves under substantial deformations and arbitrary large scaling and rigid transformations. A syntactic representation is constructed for both curves, and an edit transformation which maps one curve to the other is found using dynamic programming. We present extensive experiments, where we apply the algorithm to silhouette matching. In these experiments we examine partial occlusion, viewpoint variation, articulation, and class matching (where silhouettes of similar objects are matched). Based on the qualitative syntactic matching we define a dissimilarity measure, and we compute it for every pair of images in a database of 121 images. We use this experiment to objectively evaluate our algorithm: First, we compare our results to those reported by others. Second, we use the dissimilarity values in order to organize the image database into shape categories. The veridical hierarchical organization stands as evidence to the quality of our matching and similarity estimation.

1 Introduction

Given a large collection of images, unraveling its redundancies is an important and challenging task. One could use this knowledge to assist in image querying, and to construct more efficient and compact image representations. In order to identify redundancy in the database, we propose the following approach: First, design algorithms to measure the similarity between images. Second, given pairwise image similarity, use similarity-based clustering to reveal the structure in the data by hierarchically dividing the images into distinct clusters. Third, identify redundancy in each cluster and use it to prune the database and pick cluster representatives; this would allow for efficient indexing into the database.

It is important to distinguish between our approach, where the clustering of N images uses only their $N \times N$ similarity matrix, and the more typical approach where images are first embedded in some D -dimensional vector space, whose dimension should be significantly reduced using such methods as PCA. Mapping an image into such a space in effect requires the identification of D measurements (or “features”) that completely describe the image. This has proven to be an elusive task. The task of image comparison, on the other hand, seems more within our reach: rather than look for an explicit representation of images as vectors, we seek an algorithm (as complex as necessary) which receives as input two images, and returns as output the similarity between them.

In this paper we focus on the design of similarity measures, limiting ourselves to the *shape* dimension of similarity and ignoring other dimensions (e.g., color, motion, context). The similarity is therefore defined as similarity between silhouettes. We describe our silhouette matching algorithm and show results of extensive experiments with real images. We then outline a stochastic clustering

algorithm (described at length in [14]), and argue that the good clustering results we show give objective evidence to the quality of our silhouette matching algorithm. The issue of database pruning, and how to identify within-cluster redundancies to allow for efficient indexing into the database, is discussed in later work [21].

More specifically, we describe a novel flexible curve matching algorithm to relate between feature points that are automatically extracted on the boundaries of objects. Unlike most pattern recognition applications of clustering, we use real images of three dimensional objects, basing our similarity measure on the shape of their occluding contours. Our algorithm is designed to give a graded similarity value, where low values reflect similarity between weakly similar curves, and higher values indicate strong similarity. To illustrate, given two curves describing the shape of two different mammals, we consider our algorithm to be “successful” if it matches their limbs and head correspondingly. The matched pairs of feature points are then aligned using an optimal $2D$ similarity transformation (translation, rotation and scale). From the residual distances between corresponding features we compute a robust dissimilarity measure between silhouettes. The matching algorithm is outlined in Section 3, and the resulting dissimilarity values are compared with those reported in the literature.

According to the general approach adopted in this paper, our next step is to feed the computed dissimilarities into a pairwise clustering algorithm, to obtain hierarchical clusters of similar images. A pairwise clustering algorithm exploits only proximity information, and is therefore suitable when vectorial representation of images is not available. Instead, the images are represented as nodes in a graph, with edges whose weight reflect the similarity between every image pair. In [14] we describe our stochastic clustering algorithm, which is outlined in Section 4. Our algorithm determines the number of clusters in a (true) hierarchical manner, and tolerates to some extent violations of metric properties (i.e., violation of the triangle inequality). We demonstrate perceptually veridical results using a database of 121 images of 12 different objects, which are hierarchically classified. The useful clustering results illustrate the quality of our matching algorithm, and the usefulness of our general approach.

2 Curve matching: problem and related work

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 some scaling or $2D$ rigid transformation 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. The organization of silhouettes into shape categories (like tools, cars, etc.) necessitates *flexible* matching, which can support graded similarity estimation.

While our approach focuses on the silhouette boundary, a dual approach is based on its medial axis. Specifically, a medial axis together with singularities labeling form a shock graph representation, and matching shock graphs is an isomorphism problem. The methods for solving it includes semi-definite programming [37], replicator dynamics [31], graduated assignment [35] and syntactic graph matching [39]. In some of these cases the matching is only structural, while in others two levels of matching (structural and metrical) are supported. The methods based on shock graphs succeed to define a graded similarity measure, and may be combined with suitable database indexing [36]. In this paper we show, however, that our results are of the same quality in spite of using boundary representation, which is inherently less sensitive to occlusion, and which does not involve

the NP-complete graph isomorphism problem.

To put our method in the context of existing work on boundary matching, we first distinguish between dense matching and feature matching. Dense matching is usually formulated as a parameterization problem, with some cost function to be minimized. The cost might be defined as the “elastic energy” needed to transform one curve to the other [5, 8, 10], but other alternatives exist [2, 15, 12, 29]. 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 they are usually not invariant under both 2D rotation and scaling. In addition, the computation of elastic energy (which is defined in terms of curvature) is scale dependent, and requires 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 [4, 20, 22, 44]. Consequently these methods are rather slow; moreover, if scaling is permitted an erroneous shrinking of one feature set may result, followed by the matching of the entire 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 points order, but then the method becomes syntactic (see below). Moreover, we illustrate in section 5.7.3 why proximity matching is not adequate for weakly similar curves. As an alternative to the alignment transformation, features may be mapped to an intrinsic invariant coordinate frame [27, 33, 34]; the drawback of this approach is that it is global, as the entire curve is needed to correctly compute the mapping.

Features can be used to divide the curves into shape elements, or primitives. If a single curve can be decomposed into shape primitives, the matching algorithm should be constrained to preserve their order. But 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 [6, 25, 9], 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., clustering in Hough space [38], geometrical hashing [24], and clique finding in an association graph [7, 11, 18, 23]). Note that at the application level, finding isomorphism between attributed relational graphs is the same problem as finding isomorphism between shock graphs (discussed above), although in the last case an additional constraint may apply [31].

For our purpose of matching complex outlines, it is advantageous to use the natural order of primitives. This results in a great simplification, and there is no need to solve the difficult graph isomorphism problem. Moreover, the relations encoded by the attributed relational graphs need to be invariant with respect to 2D image transformations, and as a result they are usually non local.

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 [45, 28, 17]. The vision problem is different from the string matching problem in two major aspects, however: first, in vision 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, 16, 26] or may be addressed by constructing for each

curve a cascade of representations at different scales [3, 30, 43]. 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 5.7). Using *variant* attributes is less efficient, but provides the possibility to define a merge operator which can handle noise [32, 40, 41], and might be useful (if correctly defined) in handling resolution change. However, the methods using variant attributes could not ensure rotation and scale invariance.

3 Flexible syntactic curve matching: algorithm

In this section we present a local syntactic matching method which can cope with both occlusion and irrelevant changes due to image transformation, while using variant attributes. These attributes support a simple smoothing mechanism, hence we can handle true scale (resolution) changes. The algorithm is outlined in Section 3.1, while the missing details are given in Section 5. We are primarily concerned with the amount of flexibility that our method achieves, since we aim to apply it to weakly similar curves. Section 3.2 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.

3.1 The proposed matching method

The occluding contours of objects are first extracted from the image and a syntactic representation is constructed, whose primitives are line segments, and whose attributes are length and absolute orientation. Our algorithm then uses a variant of the edit matching procedure combined with heuristic search. Thus we define a novel similarity measure between primitives to assign cost to each edit operation, a novel merge operation, and introduce penalty for interrupting a contour (in addition to the regular deletion/insertion penalty).

More specifically, let A and A' be two syntactic representations of two contours; $A = \{a_1, a_2, \dots, a_N\}$ is a cyclically ordered list of N line segments, and $A' = \{a'_1, a'_2, \dots, a'_{N'}\}$ is another cyclic list of N' segments.

Let a_i be a segment of A and a'_j be a segment of A' . Matching these segments uniquely determines the relative global rotation and scale (2D alignment transformation) between the curves. We assume that the optimal alignment is well approximated by at least one of the NN' possible selections of a_i and a'_j . In fact, we will discuss in Section 5.2 a method to prune many of them, leaving us with a set $\Psi \subseteq A \times A'$ of candidate global alignments, such that usually $|\Psi| \ll NN'$.

A member $\{a_i, a'_j\}$ of Ψ (abbreviated $\{i, j\}$ for convenience) denotes a starting point for our syntactic matching algorithm. The algorithm uses edit operations to extend the correspondence between the remaining unmatched segments, preserving their cyclic order. Total cost is minimized using dynamic programming, where the cost of a match between every two segments depends on their attributes, as well as on the attributes of the initial chosen pair (a_i and a'_j). This implicitly takes into account the global alignment operation. The various edit operations and their cost functions are described in Section 5.3; the cost of the edit operations can be either negative or positive.¹

¹Negative cost can be interpreted as positive “gain” or “reward”. Every matching prefix has a total (accumulated) cost, which should be as negative as possible. A prefix is never extended by a suffix with positive cost, since this would increase the cost; hence *partial* matching is achieved by leaving the last segments unmatched. Note that if all costs are negative then the minimal cost must be obtained by matching all the segments of the shorter sequence, and

```

procedure CURVE-MATCHING:
  input: two curve representations  $A$  and  $A'$ .
  output: correspondence between line segments and dissimilarity value.
   $\Psi, cost^*, \{i^*, j^*\} \leftarrow \text{INITIALIZE}(A, A')$ 
   $\{i, j\}, potential \leftarrow \text{PICK-CANDIDATE}(\Psi)$ 
  While  $cost^* > potential$ 
     $cost^*, \{i^*, j^*\} \leftarrow \text{SYNTACTIC-STEP}(A, A', \{i, j\})$ 
     $\Psi \leftarrow \text{PREDICT-COST}(\{i, j\})$ 
     $\{i, j\}, potential \leftarrow \text{PICK-CANDIDATE}(\Psi)$ 
  end-loop
   $correspondence \leftarrow \text{TRACE}(\{i^*, j^*\})$ 
   $dissim \leftarrow \text{DISSIMILARITY}(A, A', correspondence)$ 
  return  $correspondence$  and  $dissim$ .

```

Figure 1: Pseudo code for CURVE-MATCHING procedure

We are searching for the member $\{i, j\} \in \Psi$ for which the extended correspondence list has minimal edit cost. Brute force implementation of this search is computationally infeasible. The syntactic matching process is therefore interlaced with heuristic search for the best initial pair in Ψ . Namely, a single dynamic programming extension step is performed for the best candidate in Ψ (possibly a different candidate in each extension matching step), while maintaining the lowest cost achieved by any of the sequences. When no candidate in Ψ has the potential to achieve a lower cost, the search is stopped (see below).

The pseudo-code in Figure 1 integrates the components of our matching algorithm into a procedure which gets two syntactic representations and returns a segment correspondence and a dissimilarity value. The arrays which support the dynamic programming are not referenced in this pseudo-code, to increase its readability. We note that the procedure CURVE-MATCHING minimizes the edit cost which is typically negative, thus, in effect, CURVE-MATCHING is maximizing the “gain” of matching.

The procedure INITIALIZE performs the initial pruning of pairs of starting points, and returns the set of candidates Ψ sorted by increasing potential values (see below). Full description is given in Section 5.2. Its implementation performs a few syntactic matching steps for all NN' possible pairs, and computes the intermediate edit cost corresponding to this partial matching. The minimal intermediate edit cost achieved by any of the candidates is returned as $cost^*$, and the candidate pair which achieves $cost^*$ is returned as $\{i^*, j^*\}$.

The procedure PICK-CANDIDATE selects a particular member of Ψ , to be fed into the syntactic algorithm. To understand its operation, we need to define the concept of potential: For each candidate $\{i, j\}$ that has been extended to a correspondence list of some length, we compute a lower bound on its final edit cost. This bound is based on the intermediate edit cost which has already been achieved, and the cost of the best (lowest cost) possible matching of the remaining unmatched segments. We call this bound the *potential* of the candidate $\{i, j\}$. The procedure PICK-CANDIDATE returns the member of Ψ which has best (minimal) potential.

Technically, we store Ψ as an ordered list, sorted by increasing potential value. Each member of Ψ is a candidate $\{i, j\}$ and its current potential. The procedure PICK-CANDIDATE then returns the first member of Ψ . The list Ψ is initially sorted by INITIALIZE, and its order is maintained by the

if all costs are positive then the minimal cost is trivially obtained by leaving all segments unmatched. The average cost value determines the asymptotic matching length for random sequences [13].

procedure **PREDICT-COST** discussed below.

The search for the best candidate $\{i^*, j^*\}$ continues as long as there exists a candidate whose potential is lower than the best cost achieved so far ($cost^*$). It is implemented by the loop which iterates as long as $cost^* > potential$. Note that $cost^*$ cannot increase and $potential$ cannot decrease during the search.

The procedure **SYNTACTIC-STEP** is the core of our algorithm. It is given as input two cyclically ordered sequences $A = \{a_1, \dots, a_N\}$ and $A' = \{a'_1, \dots, a'_{N'}\}$, which are partially matched from position i of A and onward, and from position j of A' and onward. It uses dynamic programming to extend the edit transformation between A and A' by one step. Since our editing cost operation typically takes negative values, the edit cost of $\{i, j\}$ could become better (lower) than $cost^*$. In this case $\{i^*, j^*\}$ is set equal to $\{i, j\}$, and $cost^*$ is set equal to the newly achieved edit cost (otherwise $\{i^*, j^*\}$ and $cost^*$ remain unchanged). Sections 5.3 and 5.4 give the full description of the procedure **SYNTACTIC-STEP**.

Extending the editing sequence of $\{i, j\}$ is likely to increase its potential, making it a less attractive candidate. This is because the potential of $\{i, j\}$ is partially determined by a lower bound on the final edit distance between the yet unmatched segments, and the edit operation just added can only tighten this bound by decreasing the number of unmatched segments. The procedure **PREDICT-COST** re-estimates the final cost, and corrects the potential of $\{i, j\}$. Since Ψ is kept as an ordered list, **PREDICT-COST** pushes the candidate $\{i, j\}$ down to maintain the order of the list. Section 5.5 gives full details of the potential estimation.

Assuming that the reader is familiar with conventional dynamic programming implementations, it is sufficient to describe the procedure **TRACE** as the procedure which reads the lowest cost path from the dynamic programming array². When this procedure is applied to the array associated with the best candidate $\{i^*, j^*\}$, the lowest cost editing sequence is obtained. In our implementation, in order to keep space complexity low, we keep just the last few rows and columns for each array, hence the procedure **TRACE** needs to repeat the syntactic matching for the best pair $\{i^*, j^*\}$. See Section 5.4 for a description of the dynamic programming implementation.

Finally, using the correspondence we found, we refine the global $2D$ alignment by minimizing the sum of residual distances between matched segments endpoints. The procedure **DISSIMILARITY** performs the minimization, and uses the residual distances to define a robust measure of dissimilarity between the curves (details in Section 5.6).

Our approach thus combines syntactic matching with a proximity measure (in this sense our method resembles that of [1]). That is, we establish feature correspondence using syntactic matching, and then evaluate the dissimilarity according to the residual distances between matched points. We *do not* use the edit distance as a measure of dissimilarity, mainly due to the fact that this quantity depends on the somewhat arbitrary parameters of the edit operation and segment similarity, whereas typically the best *matching* result is not sensitive to these exact parameters. That is, the same matching is obtained for a range of edit parameter values, although the edit distance may be different. Another advantage to combining syntactic and proximity criteria is that in many cases the combination provides a mechanism for outliers removal, as is demonstrated in Section 5.6.

3.2 Matching results

We now present a few image pairs and triplets together with the matching results, which demonstrate perceptually appealing matching. In Section 3.3 below we apply our matching algorithm to a database of 31 silhouettes given to us by Sharvit & Kimia, and compare our dissimilarity values

²Note, however, that using both positive and negative costs allows for partial matching, hence the path can terminate at any entry of the dynamical programming array and not necessarily at the last row or column.

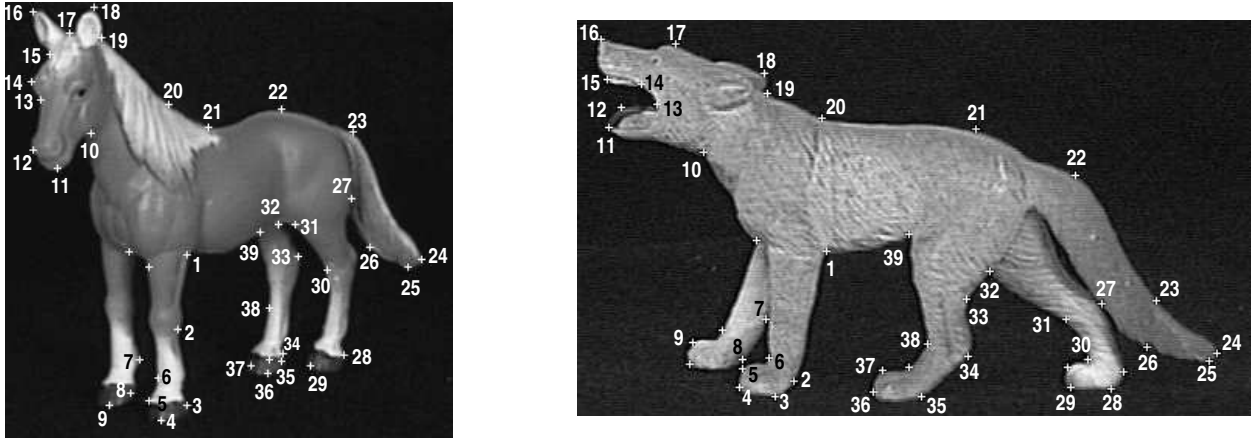


Figure 2: Qualitative matching between pictures of toy models of a horse and a wolf. Note the correct correspondence between the feet of the wolf to those of the horse, and the correspondence between the tails. The results are shown without outliers pruning. In this example, all the features to which 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.

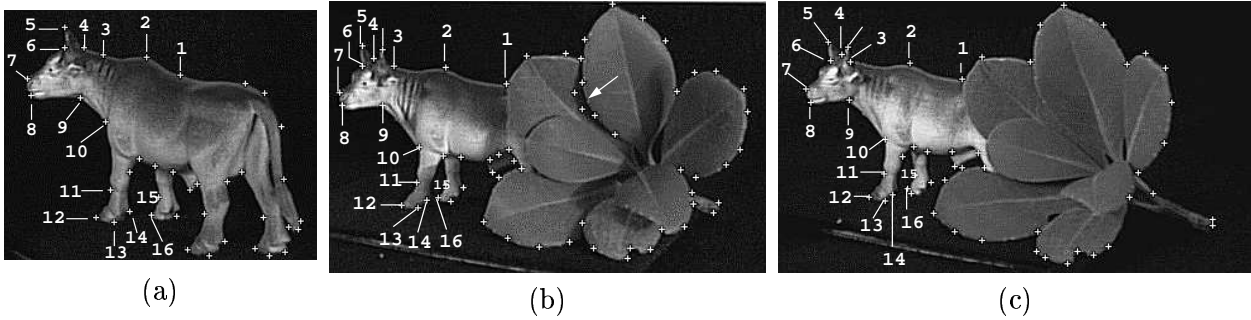


Figure 3: Dealing with occlusion: partial matching between three images. Points are mapped from (a) to (b) to (c) and back to (a). Only points which are mapped back to themselves are accepted (order is not important). The points on the tail in (a) are matched with the shadow (pointed by the arrow) in (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.

to those reported in [35]. Additional classification results will be presented in Section 4.2, using the matching of a few thousands image pairs, to provide indirect objective examination of the matching quality.

In all the experiments reported in this paper we use the same parameter values (defined in Section 5): $w_1 = 1$, $w_2 = 0.8$, $w_3 = 8.0$ and $K = 4$ (with the exception of Figure 6, where $K = 5$). Each matching of an image pair took only a few seconds (see section 4.2).

Figure 2 shows two images of different objects. There is a 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 animals and the feature points were automatically extracted in the preprocessing stage. Corresponding points are marked in Figure 2 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 3 demonstrates the local nature of our algorithm, namely, that partial matching can be

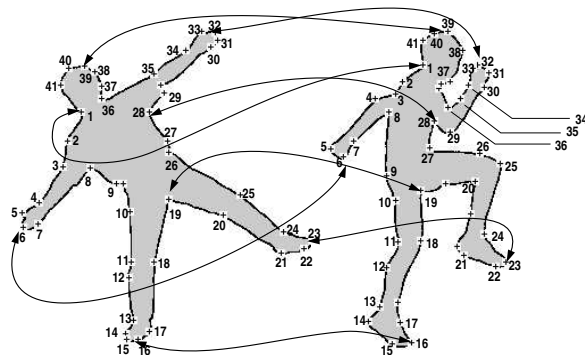
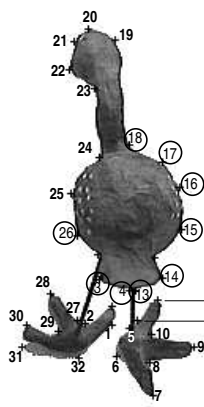
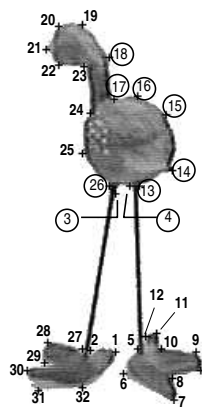


Figure 4: Matching two views of an object subjected to large foreshortening. Rejected pairs (in circles) were detected in four iterations of eliminating the (10%) most distant pairs and re-aligning the others. 33 and 35 features were extracted on the two outlines; 32 pairs were initially matched, and 9 pairs were rejected (28%).

Figure 5: 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.



Figure 6: Combination of various sources of difficulty: different models, different viewpoints and occlusion. The merging utility is used to overcome the different number of feature points around the wheels; gap insertion is utilized to ignore the large irrelevant part.

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.

Figure 3 also demonstrates outliers pruning using three images. In image 3b 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 3a, since the two parts are locally similar. However, we use the procedure of mapping the points of 3a to 3b, then to 3c and back to 3a. Only points which are mapped back to themselves are accepted as correct matches; these matched are marked by common numbers in Figure 3.

Figures 4 and 6 show results when matching images taken from very different points of view. In Figure 4 two different views of the same object are matched, and the method of iterative elimination of distances is demonstrated (see Section 5.6). Figure 6 shows matching between three different cars, viewed from very different viewpoints and subjected to occlusion. Matching under a large perturbation of viewpoint 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 [46].

The last example (Figure 5) shows results with an articulated object, matching human limbs at different body configurations.

3.3 Dissimilarity measurements: comparison


















































	   	   	   	   	   	   
	0 30 59 37	77 183 118 103	164 124 116 142	120 109 113 145	116 111 114 93	125 144 172 143 120
	30 0 50 25	123 84 142 146	111 137 109 115	111 112 109 113	82 107 153 137	106 110 117 132 123
	59 50 0 47	117 85 122 142	110 200 73 142	108 111 109 112	91 102 132 113	134 141 124 140 135
	37 25 47 0	106 112 144 108	125 120 104 137	109 122 126 140	106 111 109 104	143 169 163 148 111
	77 123 117 106	0 21 59 47	105 127 163 139	105 140 143 123	150 145 112 101	144 187 163 134 115
	183 84 85 112	21 0 89 78	124 116 134 84	157 125 108 128	64 132 140 129	177 142 171 140 150
	118 142 122 144	59 89 0 65	103 140 116 179	122 141 132 122	85 107 127 95	134 117 133 162 139
	103 146 142 108	47 78 65 0	114 92 127 82	97 136 118 95	81 180 128 131	128 91 116 149 117
	164 111 110 125	105 124 103 114	0 76 22 61	58 68 67 73	107 75 77 95	87 93 68 87 69
	124 137 200 120	127 116 140 92	76 0 28 78	74 82 63 62	72 115 74 91	91 115 99 92 83
	116 109 73 104	163 134 116 127	22 28 0 86	80 72 81 127	92 77 94 113	93 113 82 114 74
	142 115 142 137	139 84 179 82	61 78 86 0	94 65 71 85	63 85 80 107	123 127 98 93 88
	120 111 108 109	105 157 122 97	58 74 80 94	0 24 26 51	77 92 74 83	91 141 66 108 93
	109 112 111 122	140 125 141 136	68 82 72 65	24 0 45 67	64 46 66 122	106 77 92 92 117
	113 109 109 126	143 108 132 118	67 63 81 71	26 45 0 53	96 65 80 65	101 106 101 97 83
	145 113 112 140	123 128 122 95	73 62 127 85	51 67 53 0	91 83 67 81	70 72 62 98 55
	116 82 91 106	150 64 85 81	107 72 92 63	77 64 96 91	0 21 28 20	143 140 132 124 86
	111 107 102 111	145 132 107 180	75 115 77 85	92 46 65 83	21 0 15 29	125 122 117 134 119
	114 153 132 109	112 140 127 128	77 74 94 80	74 66 80 67	28 15 0 21	121 174 141 158 124
	93 137 113 104	101 129 95 131	95 91 113 107	83 122 65 81	20 29 21 0	123 90 123 138 110
	125 106 134 143	144 177 134 128	87 91 93 123	91 106 101 70	143 125 121 123	0 14 20 20 19
	144 110 141 169	187 142 117 91	93 115 113 127	141 77 106 72	140 122 174 90	14 0 16 19 15
	172 117 124 163	163 171 133 116	68 99 82 98	66 92 101 62	132 117 141 123	20 16 0 30 25
	143 132 140 148	134 140 162 149	87 92 114 93	108 92 97 98	124 134 158 138	20 19 30 0 34
	120 123 135 111	115 150 139 117	69 83 74 88	93 117 83 55	86 119 124 110	19 15 25 34 0

Table 1: The dissimilarity values computed between 25 silhouettes (multiplied by 1000 and rounded). For each line, the columns that correspond to the three nearest neighbors (and the self zero distance) are highlighted. The first, second and third nearest neighbor are in the same class a fraction of 25/25, 21/25 and 19/25 of the times respectively.

In this section we use our matching algorithm to compute a dissimilarity value, as will be explained in Section 5.6. Good matching is essential for correct dissimilarity estimation, whose values we use for quantitative comparisons with other methods. We use the image database created by Sharvit & Kimia (see [35]), which consists of 31 silhouettes of 6 classes of objects (including fish, airplanes, tools).

In [35], 25 images were selected out of this database, and their pairwise similarities were computed. The measure of quality was the number of instances (out of 25) in which the first, second

and third nearest neighbor of an image was found in its own class. We follow the same procedure, and show in Table 1 the dissimilarity values which we obtain for 25 silhouettes.³

We find that the fraction of times (out of 25) that the first nearest neighbor of an image belongs to its own class is 25/25, namely, it is always the case. For the second and third nearest neighbors the results are 21/25 and 19/25. In comparison, the results reported in [35] are 23/25, 21/25 and 20/25 respectively, whereas our results for their choice of 25 images are the fractions 25/25, 20/25 and 17/25 respectively. It is to be noted, however, that in the framework of [35] one can use additional information, specifically, whether the two graphs that represent a pair of shapes have similar topology.

We conclude that the two methods are comparable in quality when isolated silhouettes are matched. This is in spite of our using boundary representation, whereas symmetry representation (shock graph) is used in [35]. The shock graph representation is inherently more sensitive to occlusions, while shock graph matching requires solving the difficult NP-complete graph isomorphism problem (see Section 1). Moreover, our method can easily be adjusted to handle open curves, by avoiding the assumption that the syntactic representation is cyclic. On the other hand, shock graphs must distinguish between interior and exterior.

Recently, progress has been made toward computing the edit distance between shock graphs [39] using a polynomial time algorithm that exploits their special structure. So far, however, the algorithm is not capable of dealing with invariance to image transformations, and no quantitative measures have been reported.

4 Clustering of silhouettes

The next step in our approach involves feeding a graph of image similarity values, computed by the silhouette matching algorithm, to a similarity-based clustering algorithm. By hierarchically dividing the images into distinct clusters we discover structure in the data. For this end we developed a stochastic clustering algorithm [14], whose full description is beyond the scope of this paper. Instead, we give below a brief review of the algorithm, and show clustering results which demonstrate the usefulness of our approach, and the quality of the matching results.

4.1 Stochastic clustering algorithm

We represent the image database as a graph, with nodes representing the images and similarity values assigning weights to the edges. Every partition of the nodes into r disjoint sets is an r -way cut in the graph, and the edges which connect between different sets of nodes are said to *cross* the cut. The value c of a cut is the sum of the weights of all crossing edges.

Our clustering method induces a probability distribution over the set of all r -way cuts in the graph, with a known lower bound on the probability of the minimal cut. Under this distribution over cuts, we compute for every two nodes u and v in the graph their probability p_{uv}^r of being in the same component of a random r -way cut.

The partition of the nodes into disjoint sets, which satisfies $p_{uv}^r < 0.5$ for every crossing edge (u, v) , is the output of our clustering algorithm for scale level r ($r = 1 \dots N$). At level $r=1$ all the nodes must be in one cluster, and as r is increased the partitions undergo a series of bifurcations. The “interesting” bifurcation points, which account for meaningful data clustering, can be clearly

³As table 1 shows, we have at least 4 images in each class. In [35] the same images were chosen, with the exception that one of the classes consisted of only 3 images, while the fish class contained 5 images. However, for members of a class consisting of only 3 images, the three nearest neighbors can no longer be all in the same class. Hence we modified slightly the choice of selected images.

distinguished from the others. This is a major advantage of our algorithm over deterministic agglomerative methods.

Our clustering method is robust and efficient, running in $O(N \log^2(N))$ time for sparse graphs, and $O(N^2 \log(N))$ for complete graphs. It does not require that the similarity values obey metric relations, hence it is suitable for the present application, where the similarity provided by our matching algorithm does not necessarily satisfy the triangle inequality.⁴

4.2 Clustering results

We now integrate the two steps of similarity estimation and pairwise clustering into one experiment of image database categorization. The database contains 121 images of 12 different objects; 90 of the images were collected by placing 6 toy models on a turntable, so that the objects could be viewed from different viewpoints. The other images are the 31 silhouettes discussed in Section 3.3. For each of the 6 toy models we collected 15 images, by rotating them in azimuth ($\vartheta = -20^\circ, -10^\circ, 0^\circ, 10^\circ, 20^\circ$) and elevation ($\varphi = -10^\circ, 0^\circ, 10^\circ$). We used models of a cow, wolf, hippopotamus, two different cars and a child.

The central images ($\vartheta = \varphi = 0$) in each of the three groups of pictures of animal models are side views (i.e., four legs, head and tail are visible). All the different 15 images of each animal model are somewhat similar in that the same parts are visible (though in some pictures some parts, such as 2 legs or a leg and a tail, are merged into one in the silhouette). Thus, there is weak geometrical similarity between all the 45 silhouettes of the three mammals, and there is weak geometrical similarity between the 30 different silhouettes of the two cars. A desirable shape categorization procedure should reveal this hidden hierarchical structure.

All the images were automatically preprocessed, to extract the silhouettes of the objects and represent them syntactically (see Section 5.1). The dissimilarities between the silhouettes are estimated using the algorithm described in section 3.1. In order to compare all the image pairs in our database of 121 images, we performed 7260 matching assignments; this took about 10 hours on an INDY R4400 175Mhz workstation (about 5 seconds per image pair, on average).

The dissimilarity matrix constitutes the input to the clustering algorithm outlined above. When the scale parameter r is varied, the hierarchical classification shown in Figure 7 is obtained. At the highest level ($r=1$) all the images belong to a single cluster. As r is increased, finer structure emerges. Note that related clusters (like the two car clusters) split at higher r values, which means that our dissimilarity measure is continuous, assigning low (but reliable) values to weakly similar shapes.

Since humans can do so, we assume that an ideal shape classifier can put the images of every object in a different class. It is hard to test this hypothesis, since as humans we cannot ignore the semantic meaning of the shapes. Nevertheless, comparing with the ideal human perceptual classification, our finest resolution level is almost perfect, with only two classification error (in the boxes marked by *) and the undesirable split of the fish cluster.

Our categorization is obtained using only intrinsic shape information. The relative size, orientation and position of the silhouettes within each category is arbitrary. Moreover, global information like the length of the occluding contour or the area it encloses are not used. Hence we expect that moderate occlusion will not affect the classification.

⁴This would also be the case had we used the generalized Hausdorff distance or the normalized edit distance [28]. The violation of metric properties is also known to exist in the function underlying our human notion of similarity between both semantic and perceptual stimuli [42].

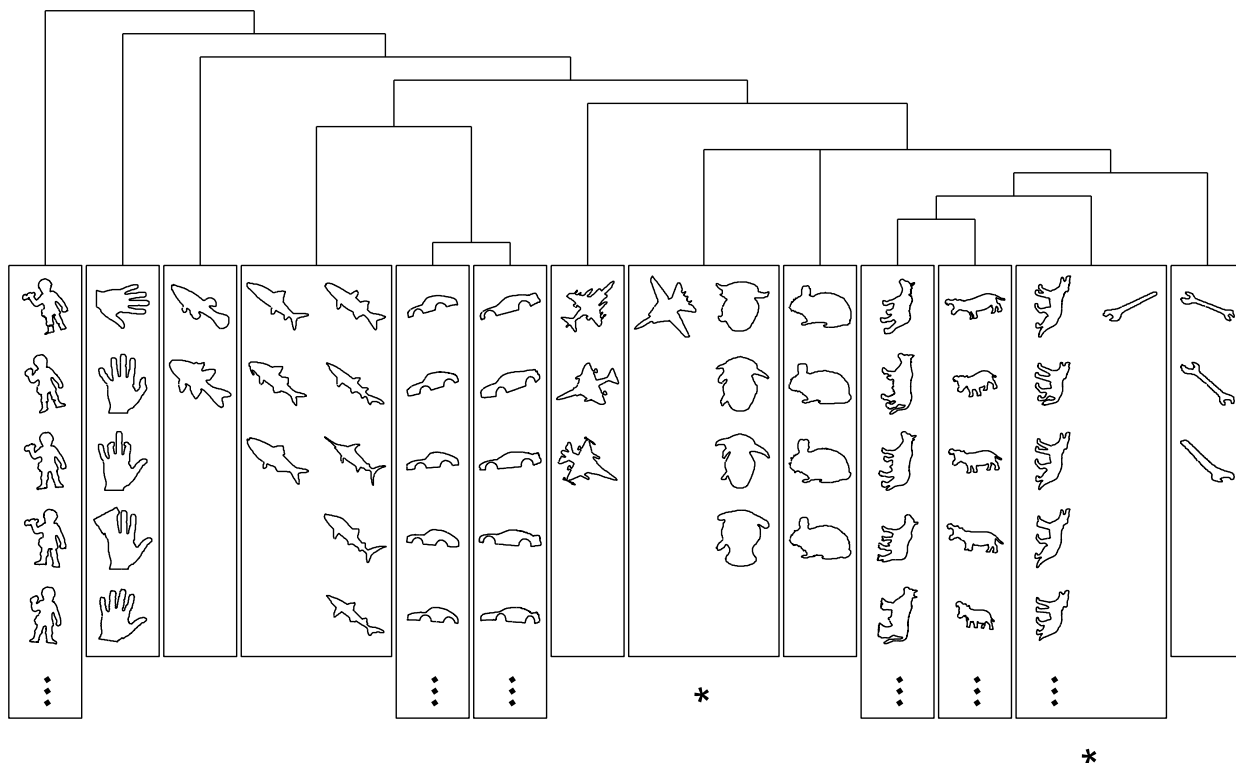


Figure 7: The classification tree (dendrogram) obtained for an image database consisting of 121 images. The finest classification level is shown by putting each cluster of silhouettes in a box. For the large clusters representing our own toy models (see text) the figure shows only 5 exemplars, but the other 10 are classified correctly as well. Note that the lower levels of the tree correspond to meaningful hierarchies, where similar classes (like the two cars or the three sets of mammals) are grouped together. The vertical axis is not in scale.

5 Flexible syntactic matching: details

In this section we give the details of the various procedures and steps involved in the curve matching algorithm, outlined in Section 3.1. Contour representation is discussed in Section 5.1. The initial pruning of candidate global alignments is discussed in Section 5.2, where we describe the procedure **INITIALIZE**. The syntactic edit operations which are used by **SYNTACTIC-STEP**, and their respective costs, are discussed in Section 5.3. The details of the dynamic programming procedure, which **SYNTACTIC-STEP** uses to minimize the edit distance, are given in Section 5.4. In Section 5.5 we define the potentials which are used to guide the search for best starting point and describe the procedure **PREDICT-COST**. Finally, the procedure **DISSIMILARITY** is discussed in Section 5.6.

5.1 Preprocessing and contour representation

In the examples shown in this paper, objects appear on dark background, and segmentation is successfully accomplished by a commercial k-means segmentation tool. A syntactic representation of the occluding contour is then automatically extracted: it is a polygon whose vertices are either points of extreme curvature, or points which are added to refine the polygonal approximation. Thus the primitives of our syntactic representation 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 it is around 50. Coarser scale descriptions may be obtained using merge

operations.

Feature points (vertices) are initially identified at points of high curvature, according to the following procedure: at every contour pixel p an angle ρ is computed between two vectors u and v . The vector u is the vectorial sum of m vectors connecting p to its m neighboring pixels on the left, and the vector v is similarly defined to the right. Points where ρ is locally minimal are defined as feature points. The polygonal approximation (obtained by connecting these points by straight lines) is compared with the original outline. If the distance between the contour points to the polygonal segments is larger than a few pixels, more feature points are added to refine the approximation.

5.2 Global alignment: pruning the starting points

The procedure INITIALIZE receives two cyclic sequences A and A' of lengths N and N' respectively, as defined in Section 3.1. Initially there are NN' possible starting points for the syntactic matching procedure; recall that each starting point corresponds to the matching of one segment a_i in A to another segment a_j in A' , thus defining the global 2D alignment between the two curves. However, the number of successful starting points is much smaller than NN' , and they tend to correspond to similar global 2D alignment transformations for two reasons: (i) Low cost transformations tend to be similar since any pair of segments $\{a_i, a'_j\}$, which belongs to a good correspondence list, is likely to be a good starting point for the syntactic algorithm. (ii) The overall number of good starting points tends to be small since most of the pairs in $A \times A'$ cannot be extended to a low cost correspondence sequence; the reason is that it might be possible to find a random match of short length, but it is very unlikely to successfully match long random sequences.

These observations are used by the procedure INITIALIZE to reduce significantly the number of candidate starting points. The procedure uses as a parameter the number t of edit operations which are performed for every one of the NN' possible starting points (in our experiments we use $t=5$ or 10). The pruning proceeds using the relation between starting points and global 2D alignments, as follows.

Every starting point $\{a_i, a'_j\}$ is associated with a global 2D alignment, and in particular with a certain rotation angle that maps the direction of a_i to that of a'_j . Let n be $\min(N, N')$, and observe the distribution of the n rotation angles which achieved the best n edit distances after t steps. If these angles are distributed sharply enough around some central value c , we conclude that c is a good estimator for the global rotation. Then, we discard every starting point in $A \times A'$ whose associated rotation is too far from c . The remaining set of candidates is the set Ψ (see Figure 8).

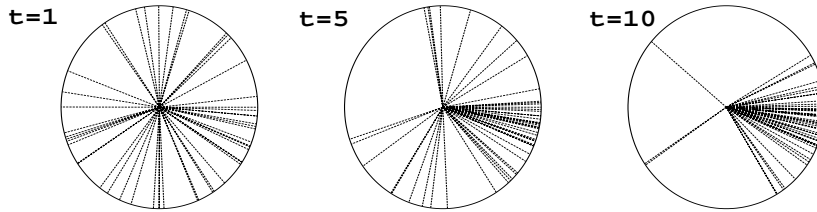


Figure 8: An example of initial pruning. Two curves with $n = \min(N, N') = 50$ are matched syntactically using t edit steps (see text for details). The 50 candidates which achieve best (minimal) edit cost are examined, to see whether the distribution of their associated rotations shows central tendency. Here we sample the distribution after 1, 5 and 10 syntactic steps. In this example 5-10 steps are sufficient for a reliable estimation of the global rotation angle c . We proceed by eliminating the candidates whose associated global rotation is too far from c .

For each candidate that remains in Ψ the procedure **INITIALIZE** computes its future potential, as discussed in Section 5.5, and sorts the list Ψ by increasing potential values. The minimal edit distance (cost) that has been achieved during the first t steps is returned as $cost^*$, and the candidate possessing this cost is returned as $\{i^*, j^*\}$.

5.3 Syntactic operations which determine the edit distance

The goal of classical string edit algorithms is to find a sequence of elementary edit operations, which transform one string into another at a minimal cost. The elementary operations are substitution, deletion and insertion of string symbols. Converting the algorithm to the domain of vision, symbol substitution is interpreted as matching two shape primitives, and the substitution cost is replaced by the dissimilarity between the matched primitives. The dissimilarity measure is discussed in Section 5.3.1. Novel operations involving gap insertion and the merging of primitives are discussed in Sections 5.3.2 and 5.3.3.

5.3.1 Similarity between primitives

We define now the similarity between line segments a_k and a_l' . The cost of a substitution operation is this value with a minus sign, hence the more similar the segments are, the lower their substitution cost is. We denote the attributes of a_k, a_l' - orientation and length - by (θ, ℓ) and (θ', ℓ') respectively. The ratio between the length attributes is denoted relative scale $c = \ell/\ell'$.

The term “reference segments” refers to the starting point segments, which implicitly determine the global rotation and scale that aligns the two curves (as discussed above). The reference segments are specified by the argument $\{i, j\}$ in the call to the procedure **SYNTACTIC-STEP**, and are denoted here a_0, a_0' . The segment similarity also depends on the corresponding attributes - orientation, length and relative scale - of the reference segments: (θ_0, ℓ_0) , (θ_0', ℓ_0') and $c_0 = \ell_0/\ell_0'$.

We first define the component of similarity which is determined by the length (or relative scale) attribute of two matching segments. We map the two matched pairs of length values $\{\ell, \ell'\}$ and $\{\ell_0, \ell_0'\}$ to two corresponding directions in the (ℓ, ℓ') -plane, and measure the angle between these two directions. The cosine of twice this angle is the length-dependent component of our measure of segment similarity (Figure 9). This measure is numerically stable. It is not sensitive to small scale changes, nor does it diverge when $c_0 = \ell_0/\ell_0'$ is small. It is measured in intrinsic units between -1 and 1 . The measure is symmetric, so that the labeling of the contours as “first” and “second” has no effect.

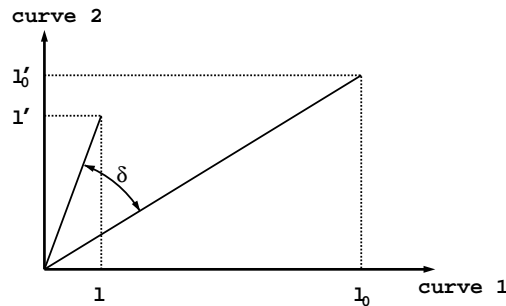


Figure 9: Length similarity is measured by comparing the corresponding reference length values $[\ell_0, \ell_0']$ with the corresponding length values of the current segment $[\ell, \ell']$. Each length pair is mapped to a direction in the plane, and similarity is defined as $\cos(2\delta)$. This value is bounded between -1 and 1 .

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

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

S_ℓ thus depends explicitly on the scale values c and ℓ_0 rather than on their ratio, hence it cannot be computed from the invariant attributes ℓ/ℓ_0 and ℓ'/ℓ'_0 . The irrelevance of labeling can be readily verified, since $S_\ell(c, \ell_0) = S_\ell(c^{-1}, \ell_0^{-1})$.

We next define 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 ($\theta_0 \rightarrow \theta'_0$):

$$S_\theta(\theta, \theta' | \theta_0, \theta'_0) = \cos[(\theta \rightarrow \theta') - (\theta_0 \rightarrow \theta'_0)] \quad (2)$$

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.

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

$$S = w_1 S_\ell + S_\theta \quad (3)$$

The positive weight w_1 (which equals 1 in all our experiments) controls the coupling of scale and orientation similarity.

5.3.2 Gap opening

In string matching, the null symbol λ serves to define the deletion and insertion operations using $a \rightarrow \lambda$ and $\lambda \rightarrow a$ respectively, where a denotes a symbol. In our case, a denotes a line segment and λ is interpreted as a “gap element”. Thus $a \rightarrow \lambda$ means that the second curve is interrupted and a gap element λ is inserted into it, to be matched with a . We define, customarily, the same cost for both operations, making the insertion of a into one sequence equivalent to its deletion from the other.

The cost of interrupting a contour and inserting ξ connected gap elements into it (that are matched with ξ consecutive segments on the other curve) is defined as $w_3 - w_2 \cdot \xi$, where w_2, w_3 are positive parameters. Thus we assign a penalty of magnitude w_3 for each single interruption, which is discounted by w_2 for every individual element insertion or deletion. This predefined quantity competes with the lowest cost (or best reward) that can be achieved by ξ substitutions. A match of ξ segments whose cost is higher (less negative) than $w_3 - w_2 \cdot \xi$ is considered to be poor, and in this case the interruption and gap insertion is preferred.

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 parameter estimation 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. Our algorithm therefore uses deletions in cases of occlusion, while for local mismatches it uses the merging operation, which is described next.

5.3.3 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 representation by being able to locally and adaptively change its scale from fine to coarse.

We define the following 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.⁵ The cost of a merge operation is defined as the dissimilarity between the merged segment and the one it is matched with. A comparison of this rule with the literature is given in the appendix.

5.4 Minimizing edit cost via dynamic programming

Procedure **SYNTACTIC-STEP** is given as input two cyclically ordered sequences $A = \{a_1, \dots, a_N\}$ and $A' = \{a'_1, \dots, a'_{N'}\}$, which are partially matched from position i of A and onward, and from position j of A' and onward. It uses dynamic programming to extend the edit transformation between A and A' by one step. The edit operations and their cost functions were discussed above in Section 5.3.

The notation used in Section 3.1 hides, for simplicity, the workspace arrays which are used for path planning. Let us assume that the procedure **SYNTACTIC-STEP** can access an array R which corresponds with the starting point $\{i, j\}$. The entry $R[\mu, \nu]$ holds the minimal cost that can be achieved when the μ segments of A following a_i are matched with the ν segments of A' following a'_j . We will see later that it is not necessary to keep the complete array in memory.

A syntactic step is the operation of getting an array R which is partially filled, and extending it by filling some of the missing entries. Our choice of extension is called “block completion”. Let us assume for simplicity that the reference segments $\{i, j\}$ are the first ones. There is no loss of generality here, since the order in A, A' is cyclic. Initially the computed portion of the array R corresponds to a diagonal block, whose two corners are $R[1, 1]$ and $R[\mu, \nu]$. When the procedure **SYNTACTIC-STEP** is applied to R , it extends R by filling in another row (of length ν), and/or another column (of length μ).

The decision whether a row or a column is to be added depends on the previous values of R , and is related to the potential computation that is discussed below in Section 5.5. Roughly speaking, we add a row (column) if the minimum of R is in the last computed row (column), and we add both if the minimum is obtained in the last computed corner.

When extending the block of size $\mu \times \nu$, every new entry $R[k, l]$ is computed according to the following rule:

$$R[k, l] = \min\{r_1, r_2, r_3\} \quad (4)$$

where

$$r_1 = \min_{\alpha, \beta \in \Omega} \{R[\alpha, \beta] - S(\overline{\alpha k}, \overline{\beta l})\}$$

⁵Implementation note: it turns out to be more convenient to keep for every line segment the attributes $(\ell, \sin \theta, \cos \theta)$ instead of (ℓ, θ) . It is straightforward to express the orientation similarity S_θ using the new attributes (and S_ℓ is, of course, unaffected). The advantage is that merging segments does not involve the computation of any trigonometric functions. Thus after merge, using the input attributes $(\ell_i, \sin \theta_i, \cos \theta_i)$ we define $x = \sum_i \ell_i (\cos \theta_i)$ and $y = \sum_i \ell_i (\sin \theta_i)$, and then $\ell = \sqrt{x^2 + y^2}$, $\sin \theta = x/\ell$ and $\cos \theta = y/\ell$.

$$\begin{aligned}
 r_2 &= \min_{0 < \alpha < k} \{R[\alpha, l] + w_3 - w_2 \cdot (k - \alpha)\} \\
 r_3 &= \min_{0 < \beta < l} \{R[k, \beta] + w_3 - w_2 \cdot (l - \beta)\}
 \end{aligned}$$

\overline{xy} denotes the vectorial sum of the segments $(x + 1), \dots, y$, and the domain Ω is defined below.

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. We define Ω as a triangular subregion of R which contains $K(K - 1)/2$ entries (Figure 10). Namely:

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

and the computation of r_1 involves $K(K - 1)/2$ evaluations of alternatives merges. The minimal value of K is 2, which is the “classical” substitution (diagonal) step.

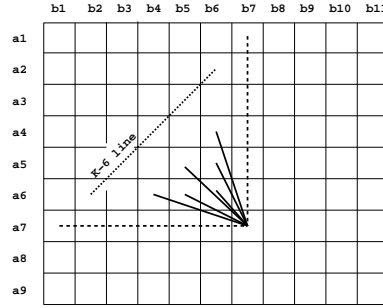


Figure 10: The entry $R[k, l]$ is updated according to Equation (4). Solid lines represent merge steps (with the “classical” substitution as a special case). Dashed lines represent the interruption and deletion of ξ connected elements.

The minimization domains for r_2 and r_3 are illustrated in Figure 10 by the horizontal and vertical dashed lines, generalizing the single element deletion operation to the deletion of ξ elements. This operation is associated with an interruption penalty (w_3) and a guaranteed reward ($w_2\xi$), which competes with the reward of substitutions.

We note that in order to find the minimal value for r_2 along the vertical line, it is not necessary to compare all its k possible values. It is sufficient to keep one index (α_0) for each column l , such that $r_2 = R[\alpha_0, l] + w_3 - w_2(k - \alpha_0)$. The initial value of α_0 is 1, and after entry $R[k, l]$ has been evaluated, the value of α_0 should be set to k iff $R[k, l] \leq R[\alpha_0, l] - w_2(k - \alpha_0)$. A similar argument applies to the computation of r_3 . Hence both the computation of r_2 and r_3 have complexity $O(1)$.

Finally, according to the block completion scheme of the array R defined above, it is easily verified that only the last $\lfloor K/2 \rfloor$ rows and columns of the block need to be stored in memory. On the other hand, if only these entries are kept, the complete path cannot be restored after the minimal edit cost is found. Hence in our implementation the procedure **TRACE** (see Section 3.1) repeats the syntactic procedure for the array possessing the best result, while keeping all of it in memory. It then follows the best path, which ends at the entry that achieved minimal edit cost, and returns it as the final matching.

5.5 Potential: bounding the edit distance

In order to guide the search for the best starting point, which is interlaced with the syntactic steps, it is necessary to bound the final minimal cost of a partially computed matching. We use a tight lower bound on the edit cost estimation, termed “potential”. The optimality of the search is thus

guaranteed. In this section we define the potential function, and explain how it is used by the procedure PREDICT-COST.

Let us consider an entry $R[k, l]$ in the dynamic programming array R . Without loss of generality, due to the cyclic segment order, we can assume that the forward path leaving this entry lies in the rectangular block defined by the corners $R[k, l]$ and $R[N, N']$. We define η and ζ to be the dimensions of this rectangle: $\eta = \min(N - k, N' - l)$, and $\zeta = \max(N - k, N' - l)$. We also define ϵ to be the maximal similarity between segments ($\epsilon = 1 + w_1$ from Equation (3)). The lowest cost substitution thus has the negative cost $-\epsilon$, which acts as a reward.

The values of the parameters w_1, w_2, w_3 are chosen to guarantee that a diagonal path consisting of the lowest cost substitutions has the minimal possible cost⁶. Hence the lowest cost forward path originating from $R[k, l]$ must contain a diagonal segment of maximal length, namely of length η . The cost assigned with this part is $-\epsilon\eta$. In addition to the diagonal part, the forward path may contain a vertical or horizontal part of length $\zeta - \eta$. In general, this part can decrease the cost only if $w_3 - w_2(\zeta - \eta)$ is negative. However, in the case where the value of $R[k, l]$ is set equal to r_2 or r_3 in Equation (4), it is the case that the entry $[k, l]$ is already considered to be inside a gap. In this case we can first extend the gap by $\zeta - \eta$ moves, and only then continue with η diagonal steps. Thus the interruption penalty is avoided, and the forward path cost becomes $-\epsilon\eta - w_2(\zeta - \eta)$. Combining this bound with the value of $R[k, l]$, we get a lower bound f_{kl} on the minimal edit cost which can be achieved passing through $R[k, l]$:

$$f_{kl} = \begin{cases} R[k, l] - \epsilon\eta + \min\{0, w_3 - w_2(\zeta - \eta)\} & \text{if } R[k, l] = r_1 \text{ in Equation (4)} \\ R[k, l] - \epsilon\eta - w_2(\zeta - \eta) & \text{if } R[k, l] = r_2, r_3 \text{ in Equation (4)} \end{cases}$$

We refer to f_{kl} as an entry potential, and define the potential of the array R as the minimum over active entry potential. An entry is active if there is a future path computation that will use its value. Hence the entries occupying the last $\lfloor K/2 \rfloor$ rows and columns of the evaluated block are active.

The procedure PREDICT-COST receives as input a candidate pair of starting points segments denoted $\{i, j\}$. It computes its potential, which is the potential of the array R associated with this pair (evidently the procedure must access the dynamic programming array R). The procedure PREDICT-COST then updates the list Ψ with the new potential value. In our implementation we keep the list Ψ sorted by increasing potentials, and the procedure PREDICT-COST places the candidate $\{i, j\}$ in its proper position, after computing its potential.

5.6 Computing curve similarity after matching

When the line segments are syntactically matched, the procedure DISSIMILARITY transforms the residual distances between their endpoints into a robust dissimilarity value. We may use the edit distance itself, but this quantity has two drawbacks. It depends on the somewhat arbitrary parameters of the edit operations (whereas typically the best *matching* result is not sensitive to these parameters), and it decreases without bound with increasing edited length. Normalizing the edit distance with respect to the curve length is not a trivial task [28].

Instead of the minimal edit distance we use its corresponding matching sequence to compute a robust proximity measure. A closed form expression for the similarity transformation (rotation, translation and scale) that minimizes the sum of squared distances between matched point sets is given in [47]. Applying this transformation to our matched points refines the alignment that was

⁶A sufficient condition is that $w_2 < \epsilon/2 = \frac{1+w_1}{2}$. If w_3 is nonzero, it is sufficient that $w_2 - \epsilon/2 < w_3/x$, where x is the largest possible diagonal path, namely $x = \min(N, N')$.

assumed during matching, since at that time the alignment was chosen from a finite set of NN' candidates. After optimal alignment we define the dissimilarity d to be equal to the k^{th} shortest residual distance, where $k = \frac{1}{2} \min(N, N')$.

In the case of complete matching, the number of matched features is $\min(N, N')$, and d becomes the median of the residual distances. The median is known to be a robust estimator, since it is not affected by extreme outliers. When many feature points are left unmatched, as happens when the two curves are not similar, d becomes larger than the median (since k is defined by the total numbers of points, N and N'). This increases our dissimilarity measure, reflecting the fact that the matching is partial. If the number of matched features is smaller than k , the dissimilarity is defined to be ∞ .

We also report on two other techniques which we have found useful for identifying and removing outliers. The first is iterative elimination: in every iteration the matched pairs that are most distant (after optimal 2D alignment) 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.

The other pruning technique can be used when three related images are available (rather than two). Assume that a feature point p on contour 1 is matched with 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 correspondence list ($p \leftrightarrow p'$, $p' \leftrightarrow p''$, $p \leftrightarrow p''$) is accepted.

5.7 Discussion and comparison to other methods

Below we discuss some issues relating to complexity, invariance and direct proximity computation. A detailed comparison of our syntactic operations to other methods is given in the appendix.

5.7.1 Complexity

The algorithm develops $|\Psi|$ dynamic programming arrays, and when it terminates each array has been completed up to a block of some size. Let n^2 be the average number of entries in a completed block. The total number of computed entries is therefore $|\Psi|n^2$. Clearly n is a fraction of N . From the considerations discussed in Section 5.2 it follows that $|\Psi|$ is typically of the order of N as well. It is possible, although not used here, to constrain the procedure INITIALIZE to return a set Ψ of size $\min(N, N')$ exactly. The overall number of computed entries is therefore $O(N^3)$.

Every single entry computation is of complexity $O(K^2)$, since $K(K-1)/2$ alternative evaluations are required to compute r_1 in Equation (4). Note that K is usually a small constant (we used $K=4$). An entry computation is followed by updating the array potential. We maintain for each row and column the best score achieved there, hence the array potential is computed in $O(K)$ time after a block is completed.

5.7.2 How to achieve invariance

Available syntactic matching methods usually achieve scale and rotation invariance (if at all) by using invariant attributes. The benefit of using invariant attributes is efficiency. The drawback is that invariant attributes cannot be smoothed by merging, and they are either non local or non interruptible. For example, in [26] 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 11). In [16] the attributes which describe curve fragments are Fourier coefficients, and in [1] an attribute called “sphericity” is defined. Both are invariant attributes, but non-interruptible⁷.

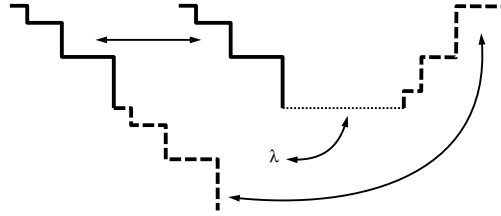


Figure 11: When the primitive attribute is measured relative to a preceding primitive, interrupting the sequence creates problems. Here, for example, the orientation information is lost when the dotted segment is matched with a gap element. As a result, the two contours may be matched almost perfectly to each other and considered as very similar.

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 [43], 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.

In contrast, our algorithm is invariant with respect to scaling, rotation and translation without relying on invariant attributes, while remaining efficient and 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.

5.7.3 Direct proximity minimization

The conversion of residual distances into a dissimilarity measure is explained in section 5.6. The reader may wonder why we can’t choose a proximity criterion, like the average distance between matched points or the Hausdorff distance, and minimize it directly. The Hausdorff distance may appear to be especially attractive, since it is not based on any prior feature pairing [2, 19].

We claim that direct minimization, which is heavily studied in the literature, is not adequate when the curves are only weakly similar. The reason is that proximity methods treat curves as two sets of points, and ignore more qualitative “structural” information.

An example is shown in Figure 12. In this example we compare two weakly similar curves, where a permissible alignment transformation includes translation, rotation and *uniform* scaling. Assuming that two sets of feature points had been extracted from the two curves, we investigate the proximity measure which is defined as the average distance between matched points. This measure is larger for the alignment shown in (c) than for the one shown in (d). Hence a proximity algorithm, which seeks the optimal alignment that achieves minimal residual distances, will consider (d) as a better alignment than (c)⁸.

⁷The 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 11) is small.

⁸Note that the lower proximity value in (d) is not the result of the use of different global scaling, since the shorter curve appears in (c) and (d) at the same size. Moreover, the lower proximity measure of (d) is obtained even though many-to-one matches are avoided, and the order of points is kept. Without these constraints, it is easy to

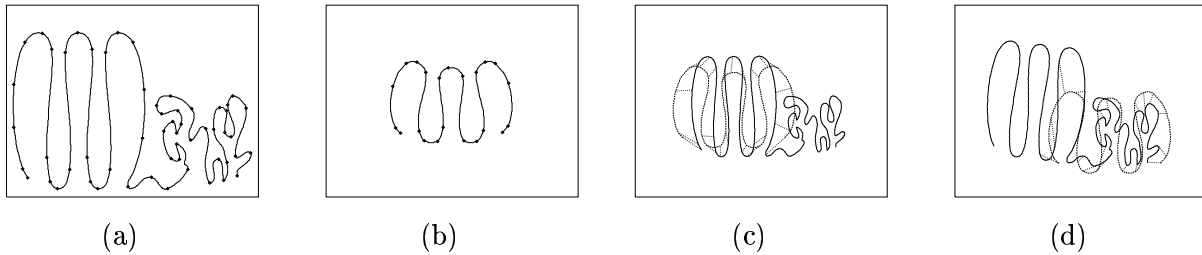


Figure 12: (a),(b) - Two weakly similar curves. The points are extracted automatically from polygonal approximations that do not depart from the original curves by more than 8 pixels. (c) The desirable matching result, as obtained by our matching algorithm. The average distance between matched feature points is 25 pixels. The directed Hausdorff distance is 34 pixels. (d) A non desirable pairing of points which yields better proximity value. The average distance between matched feature points is only 23 pixels. The directed Hausdorff distance is only 29 pixels.

We continue with the previous example and investigate instead the use of the directed Hausdorff distance as the proximity criterion. The directed Hausdorff distance from a point set P to a point set Q is defined (with the Euclidean norm $\|\cdot\|$) as: $h(P, Q) = \max_{p \in P} \min_{q \in Q} \|p - q\|$; it is equal to the largest distance from some point in P to its nearest neighbor in Q . This measure is asymmetric, and therefore the symmetric expression $\max\{h(P, Q), h(Q, P)\}$ is often preferred. However, when there is a large image clutter, as in our case, the symmetric distance is not useful since its value is determined by the irrelevant part of the curve. Thus the directed distance, measured from the shorter curve to the longer one, is larger for the alignment shown in (c) than for the one shown in (d). A proximity algorithm that is based on minimizing the directed Hausdorff distance will consider (d) as a better alignment than (c).

In contrast, because it uses local structure, our syntactic matching algorithm provides the correct matching of the curves in Figure 12 (shown in part c).

6 Discussion

This paper deals with the inherently ill-posed problem of matching weakly similar curves for the purpose of similarity estimation. Our flexible syntactic matching is based on a simple heuristic, guided by the principle that matched features should lie on locally similar pieces of curve. Naturally, this principle cannot guarantee that the results would always agree with our human intuition for “good” matching, but our examples demonstrate that satisfactory and intuitive results are usually obtained. In addition, we passed successfully two more objective tests: (i) in a database of 31 images we showed that nearest neighbors, computed according to our distance, are always of the same type; (ii) in a database of 121 images, clustering based on our similarity results gave perceptually veridical hierarchical structure. Note that “successful” matching depends on the application at hand. Our method is not suitable for recovering depth from stereo, but it is well suited for more qualitative tasks, such as the organization of an 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 is 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, add up to a robust and successful algorithm. The most important properties of our algorithm, which make it advantageous over other successful matching

get arbitrarily small proximity values for an arbitrary correspondence by shrinking one set of points and matching it with only a few points (or even a single point) from the other set.

algorithms, are its relatively low complexity, its locality which allows us to deal with occlusions, and its invariance to 2D image transformations.

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, in a classification task using our algorithm to pre-compute image pair similarity, the clustering algorithm detected all the relevant partitions. 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 for the quality of our matching method.

Note that according to the appearance based approach to object recognition, an object is represented by a collection of images, which in some sense span the image space of that object. Our method can be used to divide the appearances of an object into clusters of similar views, in order to assist the construction of an appearance based representation.

A Syntactic operations: comparison

How to measure similarity

Local and scale invariant matching methods usually use the normalized length ℓ/ℓ_0 . For example, the ratio between normalized lengths $\frac{\ell/\ell_0}{\ell'/\ell'_0}$ is used in [26, 25] (with global normalization the difference $|\ell/L - \ell'/L'|$ can be used [43, 40]). The ratio between normalized lengths may be viewed as the ratio between the relative scale $c = \ell/\ell'$ and the reference relative scale $c_0 = \ell_0/\ell'_0$. While this scale ratio is invariant, unlike our measure it is not bounded and is thus less stable.

We are familiar with only one other definition of a symmetric, bounded and scale invariant measure for segment length similarity [25]. However, their matching algorithm is not syntactic and is very different from ours. In addition, there is an important qualitative difference between the two definitions (see Figure 13), where our measure is more suitable for flexible matching.

As for our measure of orientation difference, we note that a *linear* measure of orientation differences has been widely used by others [40, 41, 26, 9]. The non linear measure used by [25] differs from ours in exactly the same way as discussed above concerning length.

Reminiscent of our combined similarity measure (3), in [32] 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 [32] it is assumed that characters are of the same scale and properly aligned).

How to open gaps

All the syntactical shape matching algorithms that we are familiar with make use of deletions and insertions as purely local operations, as 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 any event of contour interruption, in addition to the (negative) cost from deletion/insertion of any single element.

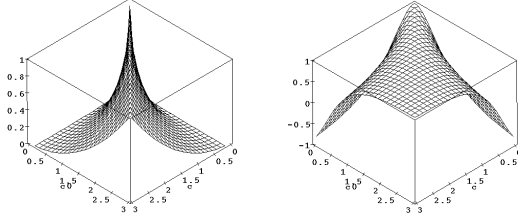


Figure 13: Scale similarity: the scale $c = \ell/\ell'$ is compared with the reference scale $c_0 = \ell_0/\ell'_0$. Left: The binary relation used by Li [25] to measure scale similarity is $\exp(-|\log(c/c_0)|/\sigma)$ with $\sigma = 0.5$. Right: Our measure function (Equation (1)) is not sensitive to small scale changes, since it is flat near the line $c = c_0$.

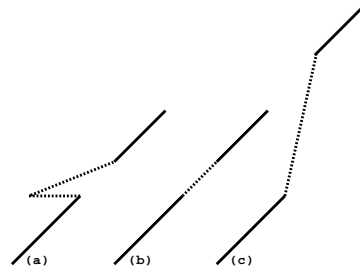


Figure 14: Comparison between merging rules: (a) A polygonal approximation of a curve, with two dotted segments which are to be merged. (b) Merging result according to our scheme. A coarser approximation is obtained. (c) Merging according to Tsai and Yu [40]. The new polygon does not appear to give a good approximation.

How to merge segments

A similar approach to segment merging (Section 5.3.3) was taken in [43], 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 new primitive. Instead, a cascade of alternative representations was 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 description⁹.

Compare the polygonal approximation after merging with the polygon that would have been obtained had the curve been 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 some feature at 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.

Merging was defined as an operation on attributes by [40], who also applied the technique to Chinese character recognition [41]. Their algorithm suffers from some drawbacks concerning invariance and locality¹⁰; below we concentrate on their merging mechanism, and compare it to our own.

Assume that two line segments characterized by (ℓ_1, θ_1) and (ℓ_2, θ_2) are to be merged into one segment (ℓ, θ) . In [40] $\ell = \ell_1 + \ell_2$, and θ is the weighted average between θ_1 and θ_2 , with weights

⁹The primitive elements used in [43] 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.

¹⁰The primitives used by [40] 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 rotation 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.

$\ell_1/(\ell_1 + \ell_2)$ and $\ell_2/(\ell_1 + \ell_2)$, and with the necessary cyclic correction.¹¹ 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 only achieved in one of the following 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 be very different from the original curve (see Figure 14). Hence, by performing segment merging on the polygonal approximation at fine scale, one typically does not obtain an acceptable coarse approximation of the shape.

Acknowledgments

We thank Ben Kimia and Daniel Sharvit for the 31 silhouette database, and Davi Geiger for the human limbs data. This research is partially founded by the Israeli ministry 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] Asada H. and Brady M., "The curvature primal sketch", PAMI 8, 2-14, 1986.
- [4] Ayach N. and Faugeras O., "HYPER: a new approach for the recognition and positioning of two dimensional objects", PAMI 8, 44-54, 1986.
- [5] 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.
- [6] Bhanu B. and Faugeras O., "Shape matching of two dimensional objects", PAMI 6, 137-155, 1984.
- [7] Bolles R. and Cain R., "Recognizing and locating partially visible objects: the focus feature method", *Int. J. Robotics Res.* 1, 57-81, 1982.
- [8] Brint A. and Brady M., "Stereo matching of curves", *Image and vision computing* 8, 50-56, 1990.
- [9] Christmas W., Kittler J. and Petrou M., "Structural matching in computer vision using probabilistic relaxation", PAMI 17, 749-764, 1995.
- [10] Cohen I., Ayache N. and Sulger P., "Tracking points on deformable objects using curvature information", *Proc. of ECCV*, 458-466, 1992.
- [11] Davis L., "Shape matching using relaxation techniques", PAMI 1, 60-72, 1979.
- [12] Del Bimbo A. and Pala P., "Visual image retrieval by elastic matching of user sketches", PAMI 19, 121-132, 1997.
- [13] Dembo A., Karlin S. and Zeitouni O., "Critical phenomena for sequence matching with scoring", *Annals of Probability* 22, 1993-2021, 1994.
- [14] Gdalyahu Y., Weinshall D. and Werman M., "Stochastic image segmentation by typical cuts", In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, Fort Collins, CO, 1999.
- [15] Geiger D., Gupta A., Costa L. and Vlontzos J., "Dynamic programming for detecting, tracking and matching deformable contours", PAMI 17, 294-302, 1995.
- [16] Gorman J., Mitchell O. and Kuhl F., "Partial shape recognition using Dynamic programming", PAMI 10, 257-266, 1988.

¹¹For example, an equal weight average between 0.9π (almost "west") and -0.9π (almost "west" as well) is π ("west") and not zero ("east").

- [17] Gregor J. and Thomason M., "Dynamic programming alignment of sequences representing cyclic patterns", PAMI 15, 129-135, 1993.
- [18] Horaud R. and Skordas T., "Stereo correspondence through feature grouping and maximal cliques", PAMI 11, 1168-1180, 1989.
- [19] Huttenlocher D., Klanderman G. and Rucklidge W., "Comparing images using the Hausdorff distance", PAMI 15, 850-863, 1993.
- [20] Huttenlocher D. and Ullman S., "Object recognition using alignment", Proc. ICCV (London), 102-111, 1987.
- [21] Jacobs D. W., Weinshall D. and Gdalyahu Y., "*Condensing image databases when retrieval is based on non-metric distances*", In Proceedings of the 6th International Conference on Computer Vision, Bombay, 1998.
- [22] Kamger-Parsi B., Margalit M. and Rozenfeld A., "Matching general polygonal arcs", CVGIP: *image understanding* 53, 227-234, 1991.
- [23] Koch M. and Kashyap R., "Using polygons to recognize and locate partially occluded objects", PAMI 9, 483-494, 1987.
- [24] Lamdan Y., Schwartz J. and Wolfson H., "Affine invariant model based object recognition", IEEE *trans. on Robotics and Automation* 6, 578-589, 1990.
- [25] Li S., "Matching: invariant to translations, rotations and scale changes", *Pattern Recognition* 25, 583-594, 1992.
- [26] Liu H. and Srinath M., "Partial shape classification using contour matching in distance transformation", PAMI 12, 1072-1079, 1990.
- [27] Lu C. and Dunham J., "Shape matching using polygon approximation and dynamic alignment", PRL 14, 945-949, 1993.
- [28] Marzal A. and Vidal E., "Computation of normalized edit distance and applications", PAMI 15, 926-932, 1993.
- [29] McConnell R., Kwok R., Curlander J., Kober W. and Pang S., " Ψ -S Correlation and dynamic time warping: two methods for tracking ice floes in SAR images", IEEE *trans. on geoscience and remote sensing* 29, 1004-1012, 1991.
- [30] Mokhtarian F. and Mackworth A., "Scale-based description and recognition of planar curves and two-dimensional shapes", PAMI 8, 34-44, 1986.
- [31] Pelillo M., Siddiqi K. and Zucker S., "Matching hierarchical structures using association graphs", Proc. of ECCV, 3-16, 1998.
- [32] Rocha J. and Pavlidis T. "A shape analysis model with applications to a character recognition system", PAMI 16, 393-404, 1994.
- [33] Sclaroff S. and Pentland A., "Modal matching for correspondence and recognition", PAMI 17, 545-561, 1995.
- [34] Shapiro L. and Brady M., "Feature based correspondence: an eigenvector approach", *Image and vision computing* 10, 283-288, 1992.
- [35] Sharvit D., Chan J., Tek H. and Kimia B., "Symmetry based indexing of image databases", *J. visual communication and image representation*, 1998.
- [36] Shokoufandeh A., Dickinson S., Siddiqi K. and Zucker S., "Indexing using a spectral encoding of topological structure", Proc. of CVPR, 491-497, 1999.
- [37] Siddiqi K., Shokoufandeh A., Dickinson S. and Zucker S., "Shock graphs and shape matching", Proc. of ICCV, 222-229, 1998.

- [38] Stockman G., Kopstein S. and Benett S., "Matching images to models for registration and object detection via clustering", *PAMI* 4, 229-241, 1982.
- [39] Tirthapura S., Sharvit D., Klein P. and Kimia B., "Indexing based on edit distance matching of shape graphs", *SPIE Proc. on multimedia storage and archiving systems III*, 25-36, 1998.
- [40] Tsai W. and Yu S., "Attributed string matching with merging for shape recognition", *PAMI* 7, 453-462, 1985.
- [41] Tsay Y. and Tsai W., "Attributed string matching by split and merge for on-line chinese character recognition", *PAMI* 15, 180-185, 1993.
- [42] Tversky A., "Features of similarity", *Psychological Review* 84, 327-352, 1977.
- [43] Ueda N. and Suzuki S., "Learning visual models from shape contours using multiscale convex/concave structure matching", *PAMI* 15, 337-352, 1993.
- [44] Umeyama S., "Parameterized point pattern matching and its application to recognition of object families", *PAMI* 15, 136-144, 1993.
- [45] Wang Y. and Pavlidis T., "Optimal correspondence of string subsequences", *PAMI* 12, 1080-1086, 1990.
- [46] Weinshall D. and Werman M., "On View Likelihood and Stability", *PAMI* 19, 97-108, 1997.
- [47] Werman M. and Weinshall D., "Similarity and Affine Invariant Distance Between Point Sets", *PAMI* 17, 810-814, 1995.