Hierarchical Matching of Deformable Shapes

Pedro Felzenszwalb
University of Chicago
pff@cs.uchicago.edu

Joshua Schwartz
University of Chicago
vegan@cs.uchicago.edu

Abstract

We describe a new hierarchical representation for two-dimensional objects that captures shape information at multiple levels of resolution. The representation is based on a hierarchical description of an object's boundary, and can be used in an elastic matching framework, both for comparing pairs of objects and for detecting objects in cluttered images. In contrast to classical elastic models, our representation explicitly captures global shape information. This leads to richer geometric models and more accurate recognition results. Our experiments demonstrate classification results that are significantly better than the current state-of-the-art in several shape datasets. We also show initial experiments in matching shapes to cluttered images.[1]

1. Introduction

Humans can often recognize objects using shape information alone. This has proven to be a difficult task for computer vision systems. One of the difficulties is in developing representations that can capture important shape variations. Moreover, we need to consider how a shape representation can be used computationally for solving vision problems. We want to be able to compare different objects and detect objects in cluttered images. The computational complexity of these tasks and the recognition accuracy obtained are highly dependent on the choice of the representation.

This paper describes an approach for matching shapes based on a hierarchical description of their boundaries. This approach can be used both for determining the similarity between two shapes and for matching a deformable shape model to a cluttered image. By using a hierarchical model, we are able to develop simple elastic matching algorithms that can take global geometric information into account.

Our matching algorithms are based on a compositional procedure. We combine matchings between adjacent segments on two curves to form matchings between longer segments. This approach makes it possible to consider the geometric arrangement among the endpoints of the matchings being combined. For long matchings, the endpoints are far away, which means that our measure of deformation captures global geometric properties. Figure 1 illustrates the procedure, where we combine a matching from A_1 to B_1 with a matching from A_2 to B_2 to obtain a longer matching between two curves. The quality of the combination depends on both the quality of the matchings being combined and the similarity between the geometric arrangements of points (p_1, p_2, p_3) and (q_1, q_2, q_3).

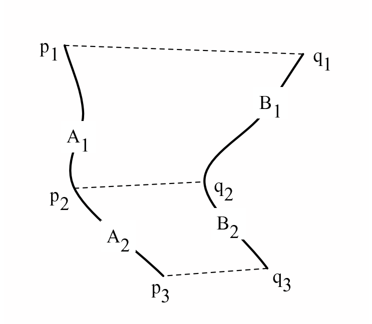


Figure 1. The composition of matchings between adjacent segments on two curves to form a matching between longer segments.

We have tested the hierarchical representation and compositional matching procedure in a variety of situations and obtained excellent performance. In classification tasks, we obtain better recognition results than other methods on several shape datasets, including the MPEG-7 shape dataset [15], a Swedish leaf dataset [25], and a silhouette dataset from Brown University [23]. We have also used the ETHZ dataset [11] to demonstrate how hierarchical matching can be used for matching shapes to real, cluttered images. These experiments illustrate how the approach is not restricted to matching pre-segmented shapes. Instead, we can match a model shape directly to an unorganized set of contours extracted from natural images.

Most of the previous elastic matching methods look for maps between two curves while minimizing a measure of local bending and stretching (see [2], [22] and references within). The methods in [6] and [12] use a similar idea

[1]This material is based upon work supported by the National Science Foundation under Grant No. 0534820.

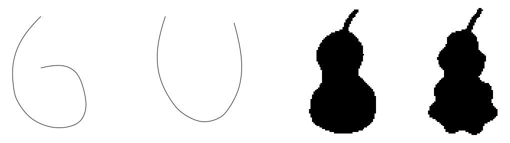


Figure 2. (a) Two curves that are almost indistinguishable by local properties alone. (b) Two objects that are similar at a coarse level but quite dissimilar at a local level.

to match a curve to cluttered images. Local deformation models are appealing from an algorithmic perspective — usually dynamic programming can be used to find optimal matchings. However, as described in [22] these methods can only address some aspects of shape similarity. Consider the curves in Figure 2(a). While they represent different characters (*6* and *U*) they can be transformed into each other without much bending and stretching. The two shapes are essentially indistinguishable if we focus on local properties alone. On the other hand, while the objects in Figure 2(b) are perceptually similar, they have completely different local boundary properties.

Our hierarchical representation captures geometric properties at different levels of resolution. At the finest level, these properties are related to standard local descriptions (capturing local curvature, for example). At coarser levels, the properties capture global shape aspects. As in classic elastic matching approaches, we use a dynamic programming algorithm for matching. But, as opposed to these other methods, ours does not solve a shortest path problem due to its compositional nature. Our compositional approach is related to the work in [4].

Hierarchical representations have proven to be useful in a variety of situations. The arc-tree in [13] gives a hierarchical description of a curve based on recursive selection of midpoints. This representation was used to perform geometric queries such as detecting intersections between two curves. Our representation can be thought of as a modified arc-tree in which the only information kept at each node is the relative position of the selected midpoint. Recursive midpoint selection is also a standard method used for polygon simplification in computer graphics [21].

In vision, multiscale representations such as the curvature scale-space (CSS) have been previously used for shape recognition [20, 19, 27]. The CSS captures critical curvature points of a contour at different levels of smoothing. Our representation is also based on a multiresolution approach, but we rely only on subsampling to define coarse geometric properties. The method in [27] uses dynamic programming for matching multiscale descriptions, but this method is not compositional in contrast to ours. Other hierarchical methods include the hierarchical graphical models in [8] and hierarchical procrustes matching [18].

The methods in [1] and [9] use triangulated graphs to represent shapes and to model deformations of objects. Our work is related since we use the geometric arrangement of sets of three points to capture shape information. Our algorithm for matching shapes to cluttered images, like that of [11], works by linking edge contours.

There are many other methods for representing, matching and recognizing shapes. These include methods based on the medial axis transform and the shock graph [5], [24], [23], procrustes analysis [7], shape contexts [3] and the inner distance [16]. We experimentally compare our algorithm to several of these approaches in Section 5.

2. The Shape-Tree

We start by describing our hierarchical representation for open curves. Let A be an open curve specified by a sequence of sample points $(a_1, \ldots, a_n)$. Let a_i be a midpoint on A. For example, we usually take $i = \lfloor n/2 \rfloor$. Another option is to choose the sample point such that the coarse curve (a_1, a_i, a_n) approximates A as well as possible. Let $L(a_i|a_1, a_n)$ denote the location of a_i relative to a_1 and a_n. The locations of the first and last sample points can be used to define a coordinate frame where we measure the location of the midpoint. The first and last sample points define a canonical scale and orientation, so the relative location $L(a_i|a_1, a_n)$ is invariant to similarity transformations.

The choice of a midpoint, a_i, breaks the original curve into two halves, $A_1 = (a_1, \ldots, a_i)$ and $A_2 = (a_i, \ldots, a_n)$. The hierarchical description of A is defined recursively, we keep track of $L(a_i|, a_1, a_n)$ and the hierarchical description of A_1 and A_2. This hierarchical description can be represented by a binary tree, as illustrated in Figure 3. We call this representation the *shape-tree* of a curve. Each node in the shape-tree stores the relative location of a midpoint with respect to the start and end point of a subcurve. The left child of a node describes the subcurve from the start to the midpoint while the right child describes the subcurve from the midpoint to the end. The leaves of this tree represent locations of sample points, a_i, relative to their neighboring points, a_{i-1} and a_{i+1}. Note that a subtree rooted at a node corresponds to the shape-tree of a subcurve.

Nodes in the bottom of the shape-tree represent relative locations of three sequential points along the curve. These nodes capture local geometric properties such as the angle formed at a point (which is a measure of curvature) and the relative distance between adjacent sample points. On the other hand, nodes near the root of the tree capture more global information encoded by the relative locations of points that are far from each other. This is a local property of a subsampled version of the original curve. The shape-tree contains only the locations of points relative to two other points. This makes the representation invariant to similarity transformations.

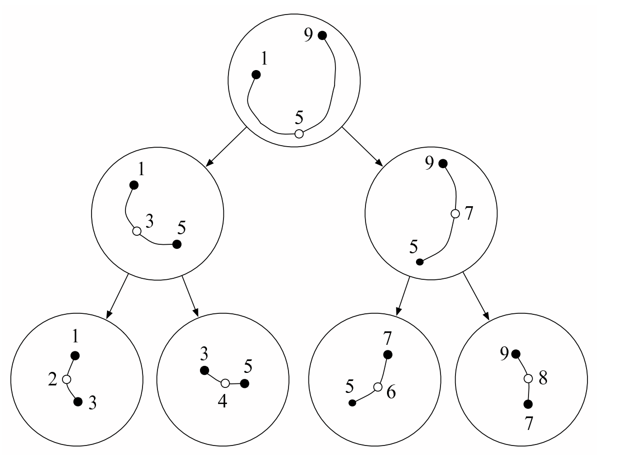


Figure 3. A shape-tree. The filled circles on each curve represent endpoints of subcurves and the unfilled circles represent midpoints. Each node stores the location of a midpoint relative to the endpoints. The midpoint becomes an endpoint when a subcurve is broken up at the level below.

Given the tree representation for A, along with the location of its start and end points a_1 and a_n, the curve can be recursively reconstructed. First, the start and end points of the curve are placed. Because the location of a midpoint of A relative to the start and the end is known, it can be placed. This process continues down the shape-tree until we have placed every sample point of A. By placing the initial points a_1 and a_n at arbitrary locations, a translated, rotated and scaled version of A can be obtained.

A closed curve can be represented in a similar fashion. Let B be a closed curve, specified by a sequence of sample points $(b_1, \ldots, b_n)$, where $b_n = b_1$. Now let b_i be a midpoint on B. The open curves $B_1 = (b_1, \ldots, b_i)$ and $B_2 = (b_i, \ldots, b_n)$ can each be represented by a shape-tree. Given a shape-tree representation of each side of a closed curve and a location for the start/end point and the first midpoint, the curve can be reconstructed at any location, orientation, and size. We simply reconstruct each side using the procedure outlined above.

We note that for a continuous curve it is possible to define an infinite shape-tree. This infinite tree gives a dense sampling of the points in the curve, fully capturing its geometry up to similarity transformations.

2.1. Deformations

We can deform a curve by perturbing the relative locations stored in its shape-tree representation. To explore this idea we need to pick a particular representation for the relative locations of the midpoints in a curve.

Bookstein coordinates [7] encode the relative locations of three points as a point in the plane. They give a simple way to represent the relative location, $L(a_i|a_1, a_n)$, of a midpoint in the shape-tree. Let v_1, v_2 and v_3 be three distinct points. There exists a unique similarity transformation that maps v_1 to $(-0.5, 0)$ and v_2 to $(0.5, 0)$. This transformation maps v_3 to a location that we call the *Bookstein coordinate* of v_3 with respect to v_1 and v_2.

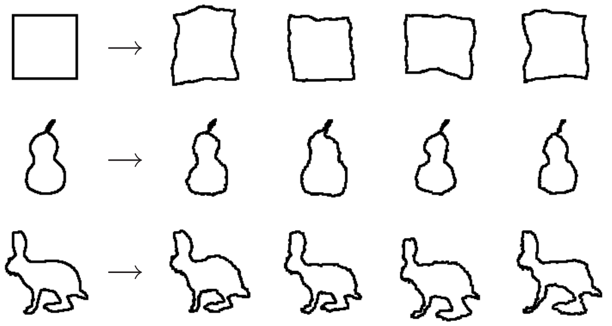


Figure 4. Random deformations obtained by adding independent noise to the nodes in a shape-tree representation of an object. The deformed squares illustrate how the method preserves important global properties while generating a wide range of variation.

Figure 4 shows some examples where we added independent noise to the Bookstein coordinates of each midpoint in a shape-tree before reconstructing a curve. The results are curves that are perceptually similar to the originals. Note that in the case of the square the deformed objects still seem to have four sides that meet at a right angle, even though the sides are quite deformed.

3. Elastic Matching

Let A and B be two open curves. When matching these curves, we build a shape-tree for A and look for a mapping from points in A to points in B such that the shape-tree of A is deformed as little as possible.[2] Here, we measure the total amount of deformation as a sum over deformations applied to each node in the shape-tree of A. The hierarchical nature of the shape-tree ensures that both local and global geometric properties are preserved by a good matching. In practice, we use use a non-uniform weighting over deformations applied to different nodes in the shape-tree. We allow larger deformations near the bottom of a shape-tree as these do not change the global appearance of an object.

Suppose $A = (a_1, \ldots, a_n)$ and $B = (b_1, \ldots, b_m)$. We assume that a_1 maps to b_1 while a_n maps to b_m. The shape-tree of A defines a midpoint a_i dividing the curve into two halves A_1 and A_2. The best match from A to B can be defined by a search for a point b_j on B where a_i maps to. This point is used to divide B into two halves B_1 and B_2 where A_1 and A_2 map to respectively. We say A and B are similar if we can find a midpoint on B such that A_1 is similar

[2]The method described here is not symmetric. Its possible to define a symmetric method but that leads to a more complex algorithm.

to B_1, A_2 is similar to B_2 and the relative locations of the midpoints $L(a_i|a_1, a_n)$ and $L(b_j|b_1, b_m)$ are similar. The similarity between subcurves is defined in the same manner. The cost of matching A to B can be expressed by a recursive equation,

$$\psi(A, B) = \min_{b_j \in B}(\psi(A_1, B_1) + \psi(A_2, B_2) + \lambda_A * \mathit{dif}(L(a_i|a_1, a_n), L(b_j|b_1, b_m))). \quad (1)$$

where *dif* measures the difference between the relative locations of the midpoints on A and B and λ_A is a weighting factor. For the experiments in this paper we used a weighting proportional on the length of A (the curve being deformed), giving a higher weights to deforming the relative locations of points that are far away. We use the full procrustes distance [7] for *dif*.

For the base case we need to define $\psi(A, B)$ when either A or B have two sample points. A curve with two sample points is just a line segment. We let the cost of matching one line segment with another be zero, while the cost of matching a line segment with a curve be exactly what it would be if the line segment was further subdivided to have the same number of sample points as the curve.

The recursive equation (1) can be solved using dynamic programming over the shape-tree of A. Let v be a node in the shape-tree of A. Consider the subcurve A' corresponding to the subtree rooted at v. Let $T(v)$ be a table of costs where $T(v)[s, e]$ is the cost of matching A' to the subcurve of B given by $(b_s, ..., b_e)$. The table $T(v)$ can be computed using equation (1) once the tables for the children of v have been computed. The algorithm computes all tables by starting at the leaves of the shape-tree and working in order of decreasing depth. The cost of matching A to B is $T(r)[1, m]$, where r is the root of the shape-tree.

There are $O(n)$ tables to be computed, and each table has $O(m^2)$ entries. To compute an entry, we have to search for an ideal midpoint on B. So, the dynamic programming procedure takes $O(nm^3)$ time overall. After all tables are computed, we can find the best matching from A to B by tracing back from the root of the shape-tree to the leaves, as in standard dynamic programming procedures.

When A and B are closed curves, we first break A in two halves, $A_1 = (a_1, \ldots, a_i)$ and $A_2 = (a_i, \ldots, a_n)$, where, as before, a_1 equals a_n. We match each node in the shape-trees of A_1 and A_2 to each subcurve of B. The cost of matching A to B, as a function of where a_1 and a_i map to, is given by $T_1(r_1)[s, e] + T_2(r_2)[e, s]$. Here r_1 and r_2 are the roots in the shape-trees of A_1 and A_2, while s and e are locations in B which a_1 and a_i map to. This leads to an $O(nm^3)$ algorithm for matching closed curves. In practice, we use between 50 and 100 sample points in each curve. Our current implementation takes about 0.5 seconds to compute a matching in a 3Ghz computer.

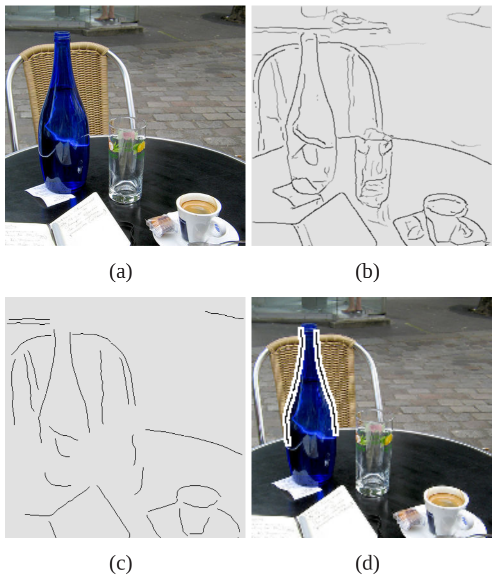


Figure 5. Detecting a bottle in an image. The input image is shown in (a). The soft edge map is shown in (b). In (c), we have the image contours extracted from (b). Our final detection is shown in (d).

The formulation above assumes that each part of A has a corresponding part on B. In many situations two curves are similar except that one of them has a missing or extra part. To make the matching robust to these transformations we bound $\psi(A', B')$ from above using a cost proportional to $\lambda_{A'} * (|A'| + |B'|)$, Intuitively this models a process where we replace a subcurve of A with a subcurve of B. Note that since the shape-tree of A is fixed in advance, this process can only replace certain parts of A. To allow for more flexibility in dealing with occlusions, we usually compute matchings using 2 to 4 different shape-trees and pick the best one. It is also possible to give a dynamic programming algorithm that allows arbitrary parts of A and B to be replaced, but that algorithm runs in $O(n^3m^3)$ time.

4. Matching to Cluttered Images

Generalizing the ideas from the last section, we can also match a model curve to a cluttered image. This algorithm proceeds in four stages. First, given a color image, we compute an edge strength map. Then, we extract a set of image contours from the edge map. After this, we match each image contour to all sub-countors of our model using dynamic programming. Finally, we use a second dynamic programming procedure to compose these matches together, forming an optimal matching between the model and a subset of the image contours. These stages are illustrated in Figure 5.

For the first stage, we use the PB edge operator [17] to compute an edge strength map. For the second stage, we trace smooth contours in the edge map using the method from [10]. The result is a set of salient contours in the image. An example can be found in Figure 5(c).

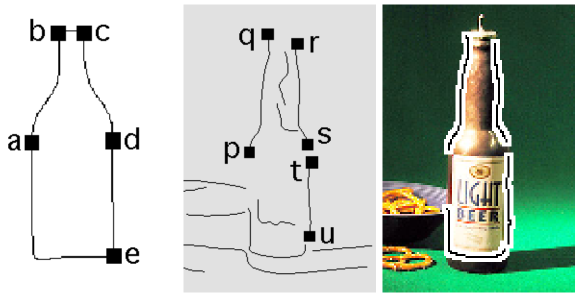
Let M be a model curve, $\mathcal{C}$ be the set of contours extracted from an image and P denote the set of endpoints of contours in $\mathcal{C}$. Our goal is to find a matching between M and a subset of $\mathcal{C}$. Let a and b be sample points in M, while p and q are points in P. We use *Match*(a, b, p, q) to denote a matching from the subcurve of M from a to b to a subset of the contours in $\mathcal{C}$ such that a maps to p and b maps to q.

In the third stage of the algorithm, we compute the best matching between each contour in $\mathcal{C}$ and each subcurve of M. This is done using the method from the last section. It takes $O(nm^3)$ time to compute a table giving the cost of deforming an image contour with n sample points to every possible subcurve in a model with m sample points. Thus, the overall running time of the third stage is linear in the total length of the contours $\mathcal{C}$ and cubic in the length of the model. This stage generates a set of matchings *Match*(a, b, p, q) that are stitched together to form larger matchings in the last stage.

We use the following compositional rule to stitch partial matchings together. Let q and r be two points in P such that $||q - r|| \leq \tau$, for some small threshold τ. If we have two matchings *Match*(a, b, p, q) and *Match*(b, c, r, s) then we can compose them to get a matching *Match*(a, c, p, s). We allow q and r to be different so that we can compose adjacent contours in the image even if their endpoints do not exactly align. Mismatches between endpoint locations can be caused by the edge detection or edge tracing procedure. In analogy to the expression in equation (1), the cost of the composed matching is the sum of the costs of the matchings being composed plus a measure of the differences between the relative locations of the midpoints in the model and the image. Here we take the “midpoint” in the image to be the average $(q + r)/2$.

Because of occlusions and missing edges we would like to allow a subcurve of the model to be left unmatched even though regions around it are matched. This is captured by considering “gap matches” *Match*(a, b, p, q) for every pair of sample points a and b in the model and points p and q in P. In these matchings a is mapped to p and b is mapped to q while the subcurve between a and b is lefted unmatched. The cost of a gap match is proportional to the arclength of the subcurve from a to b.

A complete match between M and a subset of the contours is given by a pair of matchings *Match*(a, b, p, q) and *Match*(b, a, q', p'), where both $||p' - p||$ and $||q' - q||$ are at most τ. Figure 6 illustrates the stitching procedure. We can find the best complete matching using a second dynamic programming step. We sequentially compute the cheapest matching of type *Match*(a, b, p, q) in order of increasing arclength of subcurves in the model. This stage of the algorithm runs in $O(m^3k^3)$ time, where m is the number of sample points in the model and k is the number of endpoints in P. In the future we plan to use the algorithm in [14] to compute optimal matches even faster. That method would compose matchings in order of their quality to avoid considering many possibilities that are considered by the dynamic programming procedure.



(a) Model (b) Image contours (c) Final result

Figure 6. The initial matching *Match*(a, b, p, q) can be composed with the gap match *Match*(b, c, q, r) to form a matching *Match*(a, c, p, r). Because s and t are close, the initial matchings *Match*(c, d, r, s) and *Match*(d, e, t, u) can be composed to form a matching *Match*(c, e, r, u). At this point, matchings *Match*(a, c, p, r) and *Match*(c, e, r, u) could be composed. Continuing in this way, we stitch together the boundary of the object.

5. Experiments

5.1. Shape Classification

MPEG-7 Shape Database

The MPEG-7 shape database [15] is a widely used dataset for testing shape recognition methods. The database has 1400 silhouette images, with 20 images per object class from a total of 70 different classes. Figure 7 shows some of the images in the database. The standard method for measuring the recognition rate of an algorithm in this dataset is as follows. For every image in the database, we look at the 40 most similar images and count how many of those are in the same class as the query image. The final score of the test is the ratio of the overall number of correct hits obtained to the best possible number of correct hits. The best possible number is 1400 * 20 since there are 1400 query images and 20 images per class. This is a hard dataset due to the large intraclass variability in each category. Table 1 lists the recognition rate we obtained using the shape-tree deformation method, together with results from other algorithms. Note that our method outperforms all previous systems.

Swedish Leaf Database

The Swedish leaf dataset [25] has pictures of 15 species of leaves, with 75 images per species for a total of 1125 images. Figure 8 shows some example images from this dataset. Note that some species are indistinguishable to the untrained eye. Similar to the methods in [25] and [16], we

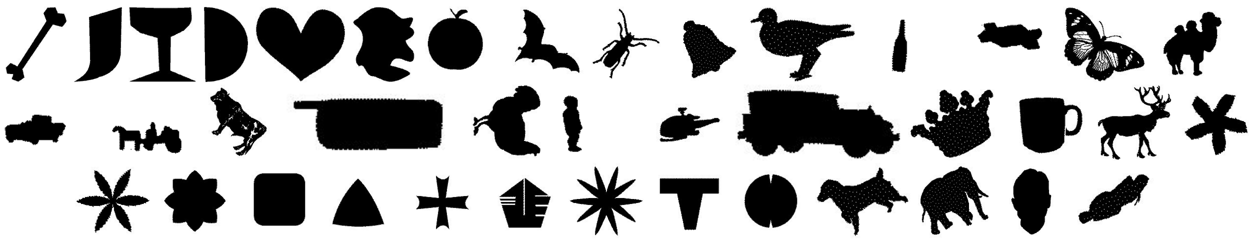


Figure 7. Some of the objects in the MPEG-7 dataset. One image per class for the first 40 classes (the database has 70 classes).

Method	Recognition rate
Shape-tree	**87.70%**
Hierarchical Procrustes [18]	86.35%
IDSC + DP [16]	85.40%
Generative Models [26]	80.03%
Curve Edit [22]	78.14%
SC + TPS [3]	76.51%
Visual Parts [15]	76.45%
CSS [19]	75.44%

Table 1. Classification results on the MPEG-7 dataset.

Method	Recognition rate
Shape-tree	**96.28%**
IDSC + DP [16]	94.13%
SC + DP [16]	88.12%
Fourier descriptors [16]	89.60%
Söderkvist [25]	82.40%

Table 2. Classification results on the Swedish leaf dataset.

randomly select 25 training images from each species and classify the remaining images using a nearest neighbor approach. Table 2 compares our classification rate to the other methods that have been tested on this dataset. The shape-tree matching algorithm outperforms the other methods by a significant amount.

Brown Database

We also tested the shape-tree matching algorithm on the silhouette database from [23]. The dataset has 11 examples from 9 different object categories for a total of 99 images. One interesting aspect of this dataset is that many of the shapes have missing parts and added clutter. Figure 9 shows some of the images. The recognition results in this dataset are measured as follows. For each shape in the database, we check if the 10 closest matches are in the same category as the query shape. Table 3 summarizes the results of different methods. Again, we see that our algorithm has a higher recognition rate than previous methods. With our method, all of the 7 best matches for each shape are in the correct category. In the best previous algorithms, only the top 3 matches were all correct.

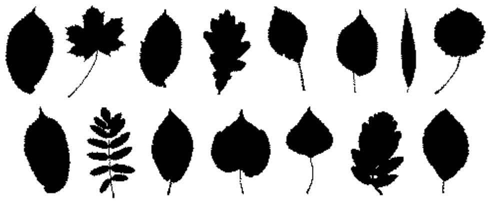


Figure 8. Leaves from the Swedish leaf dataset, one leaf per species. Note the similarity among some species.

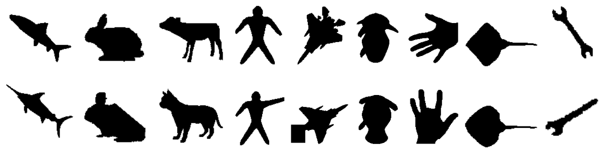


Figure 9. Images from the Brown dataset. Two per category.

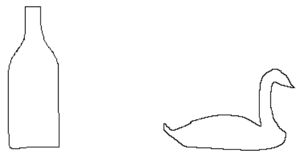


Figure 10. The models used for matching in the ETHZ dataset.

5.2. Matching in Cluttered Images

To test our matching algorithm on cluttered images, we ran experiments on a set of 80 images of swans and bottles from the ETHZ dataset [11]. Matching for each class is done with a single hand-drawn model shown in Figure 10. This makes this dataset a good test for elastic matching. The objects in each image often have substantially different shape from the model. Interestingly, several images in the dataset are paintings, drawings, or computerized renderings of scenes. Our algorithm performs very well on these images. A sampling of our results can be found in Figures 11 and 12. Note that our current implementation simply finds the best match in each picture.

6. Summary

We introduced a hierarchical shape representation with the goal of explicitly capturing both global and local geometric properties of an object. This representation is cap-

Method	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
Shape-tree	**99**	**99**	**99**	**99**	**99**	**99**	**99**	**97**	**93**	**86**
IDSC + DP [16]	99	99	99	98	98	97	97	98	94	79
Shock-Graph Edit [23]	99	99	99	98	98	97	96	95	93	82
Generative Models [26]	99	97	99	98	96	96	94	83	75	48

Table 3. Retrieval results on the dataset from [23]. Ideally the top 10 matches of each of the 99 shapes would be a shape in the same category. The table summarizes the number of correct matches in each rank.

tured by a tree, which we term the *shape-tree* of an object. We can define deformations of an object in terms of independent deformations applied to each node in its shape-tree. Since some of the nodes in the shape-tree capture global geometric information, the process of applying a small deformation to each node preserves perceptually important aspects of the object's shape.

We have used the shape-tree deformation model to develop a simple and efficient algorithm for matching curves. Our experimental results show that this method is very accurate when used for classifying objects from several large databases. Moreover, the matching algorithm can be extended to detecting deformable objects in cluttered images. Our future work will be directed towards refining and evaluating this process.

References

[1] Y. Amit and A. Kong. Graphical templates for model registration. *PAMI*, 18(3):225–236, 1996.

[2] R. Basri, L. Costa, D. Geiger, and D. Jacobs. Determining the similarity of deformable shapes. *Vision Research*, 38:2365–2385, 1998.

[3] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 24(4):509–522, April 2002.

[4] E. Bienenstock, S. Geman, and D. Potter. Compositionality, mdl priors, and object recognition. In *NIPS*, 1997.

[5] H. Blum. Biological shape and visual science. *Theoretical Biology*, 38:205–287, 1973.

[6] J. Coughlan, A. Yuille, C. English, and D. Snow. Efficient deformable template detection and localization without user initialization. *CVIU*, 78(3):303–319, June 2000.

[7] I. Dryden and K. Mardia. *Statistical Shape Analysis*. John Wiley and Sons, 1998.

[8] X. Fan, C. Qi, D. Liang, and H. Huang. Probabilistic contour extraction using hierarchical shape representation. In *ICCV*, pages I: 302–308, 2005.

[9] P. Felzenszwalb. Representation and detection of deformable shapes. *PAMI*, 27(2):208–220, February 2005.

[10] P. Felzenszwalb and D. McAllester. A min-cover approach for finding salient curves. In *IEEE Workshop on Percepeptual Organization*, 2006.

[11] V. Ferrari, T. Tuytelaars, and L. Van Gool. Object detection by contour segment networks. In *ECCV*, pages III: 14–28, 2006.

[12] U. Grenander, Y. Chow, and D. Keenan. *Hands: A Pattern Theoretic Study of Biological Shapes*. Springer-Verlag, 1991.

[13] O. Gunther and E. Wong. The arc tree: An approximation scheme to represent arbitrary curved shapes. *Computer Vision, Graphics, and Image Processing*, 51:313–337, 1990.

[14] D. Knuth. A generalization of Dijkstra's algorithm. *Information Processing Letters*, 6(1):1–5, February 1977.

[15] L. Latecki, R. Lakamper, and U. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *CVPR*, pages I: 424–429, 2000.

[16] H. Ling and D. Jacobs. Using the inner-distance for classification of articulated shapes. In *CVPR*, pages II: 719–726, 2005.

[17] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, May 2004.

[18] G. McNeill and S. Vijayakumar. Hierarchical procrustes matching for shape retrieval. In *CVPR*, pages I: 885–894, 2006.

[19] F. Mokhtarian, S. Abbasi, and J. Kittler. Efficient and robust retrieval by shape content through curvature scale space. In A. Smeulders and R. Jain, editors, *Image Databases and Multi-Media Search*, pages 51–58. World Scientific, 1997.

[20] F. Mokhtarian and A. Mackworth. A theory of multi-scale curvature-based shape representations for planar curves. *PAMI*, 14(8):789–805, 1992.

[21] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1:244–256, 1972.

[22] T. Sebastian, P. Klein, and B. Kimia. On aligning curves. *PAMI*, 25(1):116–124, January 2003.

[23] T. Sebastian, P. Klein, and B. Kimia. Recognition of shapes by editing their shock graphs. *PAMI*, 25(5):550–571, 2004.

[24] K. Siddiqi, A. Sokoufandeh, S. Dickinson, and S. Zucker. Shock graphs and shape matching. *IJCV*, 35(1):13–32, 1999.

[25] O. Soderkvist. Computer vision classification of leaves from swedish trees. Master's thesis, Linkoping University, 2001.

[26] Z. Tu and A. Yuille. Shape matching and recognition: Using generative models and informative features. In *ECCV*, pages Vol III: 195–209, 2004.

[27] N. Ueda and S. Suzuki. Learning visual models from shape contours using multiscale convex/concave structure matching. *PAMI*, 15(4):337–352, 1993.

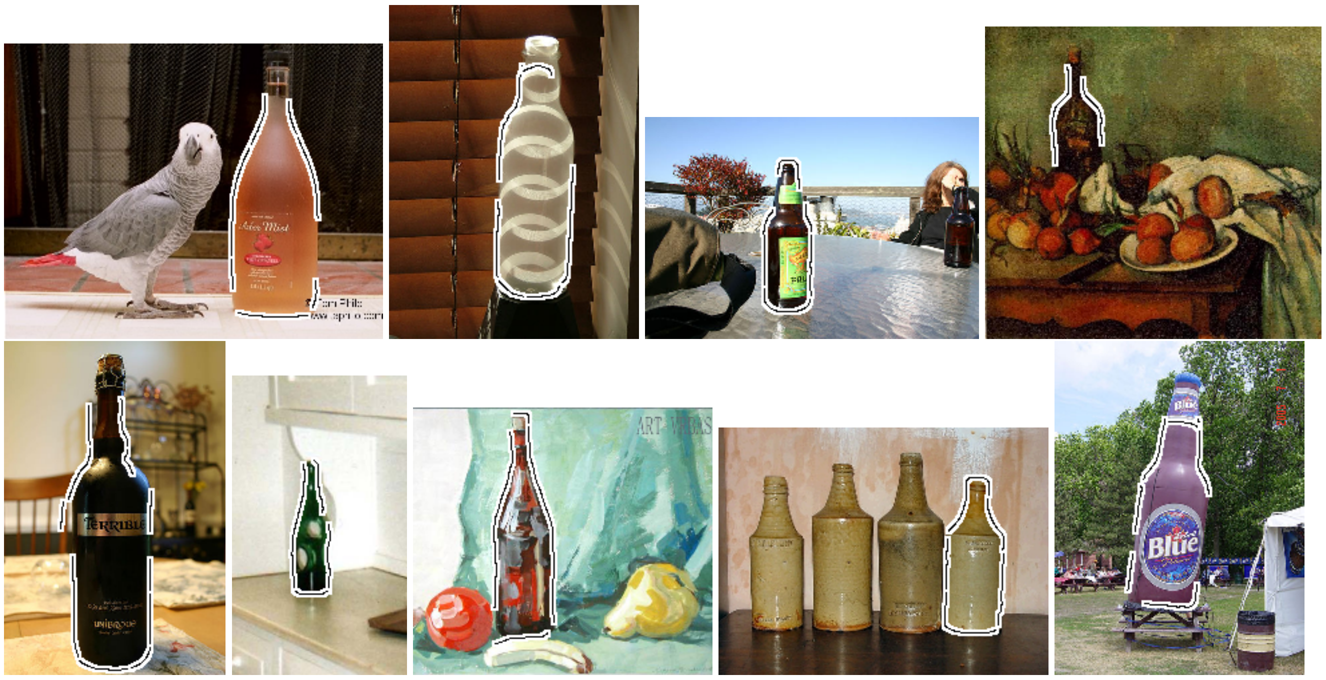


Figure 11. Some example results of matching a bottle to images in the ETHZ dataset. Only the best match in each image is shown. Most of the gaps in each matching are due to missing edges.

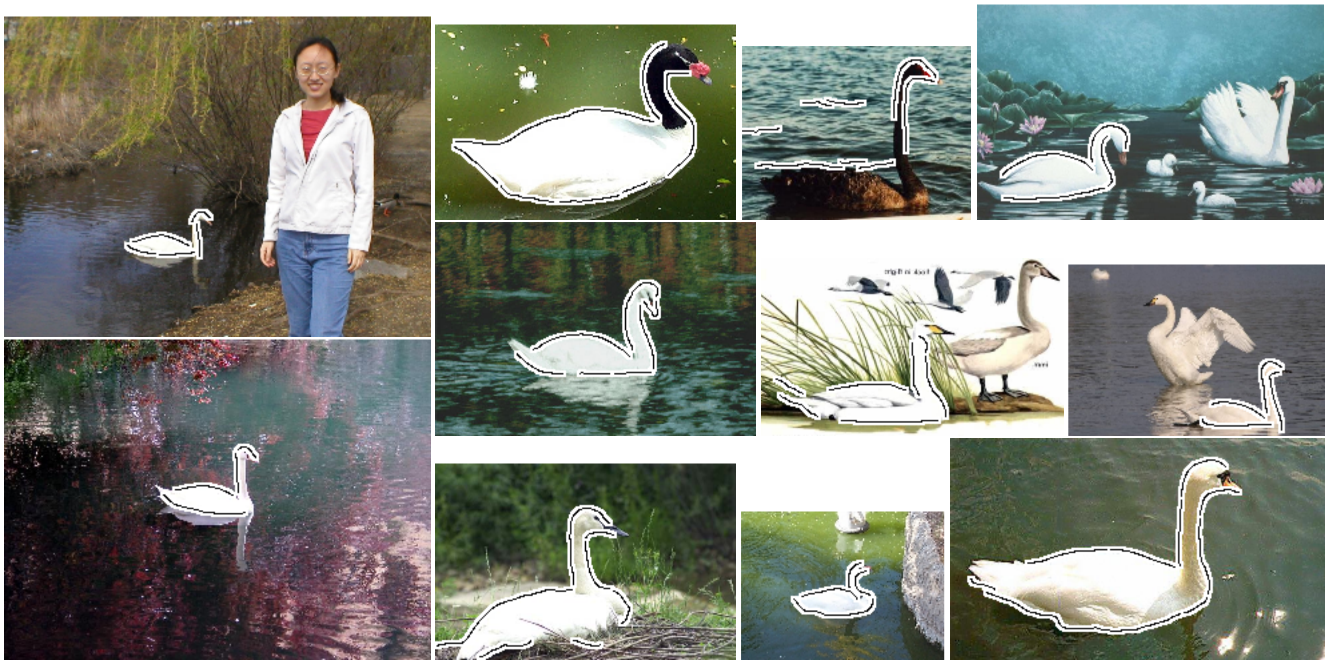


Figure 12. Some example results of matching a swan to images in the ETHZ dataset. Only the best match in each image is shown. The third image on the top shows a mistake, due to missing edges on the swan and extra edges on the water.