Shape-based Image Retrieval Using Geometric Hashing*

Scott D. Cohen **Leonidas J. Guibas**
{scohen,guibas}@cs.stanford.edu
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

We present a general strategy for *shape-based image retrieval* which considers similarity modulo a given transformation group $\mathcal{G}$. The shape content of an image is summarized by recording what geometric primitives, such as line segments and circular arcs, fit where in the image. Geometric hashing is used to compute a set of primitive features which are invariant under a $\mathcal{G}$-transformation of the image. Our search engine is *feature-based* in the sense that similarity is determined by looping over the features in the query and asking: Which database images have features that are close to a given query feature? The most similar database images are ones that have many features which are close to query features. We apply our approach to an example database of 500 chinese character bitmaps.

1 Introduction

The function of a content-based image retrieval system [Niblack *et al.*, 1993, Guibas and Tomasi, 1996] is typically to find database images that look similar to a given query image or drawing. Database and query images are usually summarized by their color, shape, and texture content. Here we use the term *images* in a very broad sense that includes any type of graphical information. Examples of images include color or grayscale pixel images, technical drawings of aircraft parts, architectural drawings, line art, and figures produced with standard drawing programs.

In this paper, we focus on *shape-based image retrieval.* We consider the *shape content* of an image to be a set of planar curves that help identify the image. A set of curves which summarize an image will be called an *illustration* of that image. For example, we might perform edgel dectection and linking to obtain an illustration of a grayscale pixel image. Of course, the database image itself may already be an illustration. This is the case for an image which is a technical drawing. The shape index of an image is derived from an illustration of that image.

This paper presents a general strategy for shape-based image retrieval in which the similarity of images is considered with respect to some given transformation group. Accounting for transformations is necessary to handle illustrations which are produced from a variety of sources. Such illustrations are likely to be expressed in coordinate systems with different units for scale and position. If we want to retrieve a portrait image from its landscape version, then our notion of similarity must also account for differences in orientation. If the illustrations are extracted from imaged objects, then allowing for a projective transformation in judging similarity is important.

There are several different approaches to building a search engine. A straightforward approach is to compare every database image to the query using some dissimilarity function defined on pairs of images. Such a retrieval strategy will eventually become too slow for interactive use as the number of images in the database grows. A related approach is to summarize image content by a point/vector in $\mathbf{R}^d$ in such a way that the L_2 distance between points is a measure of image dissimilarity. A Euclidean-space nearest-neighbor algorithm may then be used to avoid brute force search. The problem with this approach is that the dimension needed to capture differences in image content is likely to be quite high, perhaps even in the thousands, while current linear-space nearest-neighbor algorithms are limited in practice to maximum dimension $d_{\max} \approx 30$ due to "constant factors" which are exponential in the dimension. Yet another idea is to cluster database images so that when a query is far from a cluster representative, it will be far from all other images in

*Supported by ARPA grant DAAH01-95-C-R009 and by NSF grant CCR-9215219.

the cluster. This allows the search to eliminate all the images in a cluster by comparing the query to only the cluster representative. This pruning strategy can be made precise with the triangle inequality when the dissimilarity function is a metric. Unfortunately, the pruning power of the triangle inequality decreases as the dimension increases. All the previously mentioned retrieval strategies are *image-based* in the sense that direct comparisons are made between images using a dissimilarity function.

Our retrieval strategy is *feature-based* in the sense that the similarity is determined by looping over the features in the query and asking: Which database images have features that are close to a given query feature? The most similar database images are ones that have many features which are close to query features. We have traded one nearest-neighbor problem in a high-dimensional *image space* for many nearest-neighbor problems in a low-dimensional *feature space*. The challenge is to find a small feature set of an image which captures the content the image. This problem is even more difficult when the features are required to be invariant under some transformation(s) of the underlying image.

Our feature extraction approach starts with an illustration of the image. The illustration curves are projected onto a basis of *basic shapes* such as line segments, corners, circular arcs, etc.. More precisely, we record *what basic shapes fit where* in the illustration curves. This strategy was first suggested in [Cohen and Guibas, 1996]. The projection step is discussed further in section 2. An invariant set of geometric primitives (a.k.a. basic shapes) is then derived from the *projected illustration* using *geometric hashing* ([Lamdan and Wolfson, 1988]). The invariance is with respect to a transformation of the projected illustration. The geometric hashing step is the subject of section 3. Define the *features* of an image to be the geometric primitives in its *invariant projected illustration*. The final preprocessing step is to build a nearest-neighbor search structure on the set of all features of all database images. Section 4 is devoted to the creation and use of the feature space nearest-neighbor structure. Finally, we conclude in section 5 with some problems that need to be addressed in future work.

The strategy outlined in this paper will be applied to an example database of 500 chinese characters. A small sample of images in this database is shown in figure 1. The collection of chinese characters is an ideal database to test our ideas because there are many patterns which occur throughout the database at different scales, locations, and orientations. Our shape summary of a character is the *medial axis* of the set of black pixels which define the character. The results shown in figure 2 were computed using algorithms and software described in [Ogniewicz and Kübler, 1995]. The character skeletons are a very good one-dimensional summary of the characters.

2 Projecting an Illustration onto a Basis of Basic Shapes

If an illustration is not created by a drawing program with a palette of geometric primitives, then its curves are likely to be polylines with a large number of vertices. This is the respresentation for the medial axes shown in figure 2, as well as any illustration produced by linking edgels. Higher level descriptions of such curves will greatly simplify the indexing process. Therefore, we project the illustration onto a basis of basic shapes such as line segments and circular arcs. The projected illustration is a union of basic shapes which approximate the illustration curves. The basis of basic shapes is chosen so that as little information as possible is lost during projection. Different databases may call for different bases.

There are many methods for finding common geometric primitives in polylines. For example, the segmentation algorithm in [Lowe, 1987] uses a split-and-merge algorithm to divide an edgel chain into straight segments. The FEX algorithm in [Etemadi, 1992] finds straight segments and circular arcs, while the algorithm in [Rosin and West, 1995] identifies straight segments, circular arcs, and elliptical arcs. The algorithm in [Cohen and Guibas, 1997] locates any pattern shape described as a polyline within another polyline, allowing for a similarity transformation of the pattern.

There is a potential problem with separating the curve extraction and curve projection steps. The algorithms mentioned above operate on one polyline at a time with no regard for the union of polyline curves as a whole. If a long straight line segment is part of two different polylines in the illustration, then it will not be found. In the case when the underlying image is a color or grayscale pixel image, one could use an algorithm for finding geometric primitives that works directly on the pixel data.

The chinese character illustrations are well approximated using circular arcs and line segments, but we simplify the medial axis pixel chains into line segments only. The results are shown in figure 3. A naive polyline simplification algorithm was used to

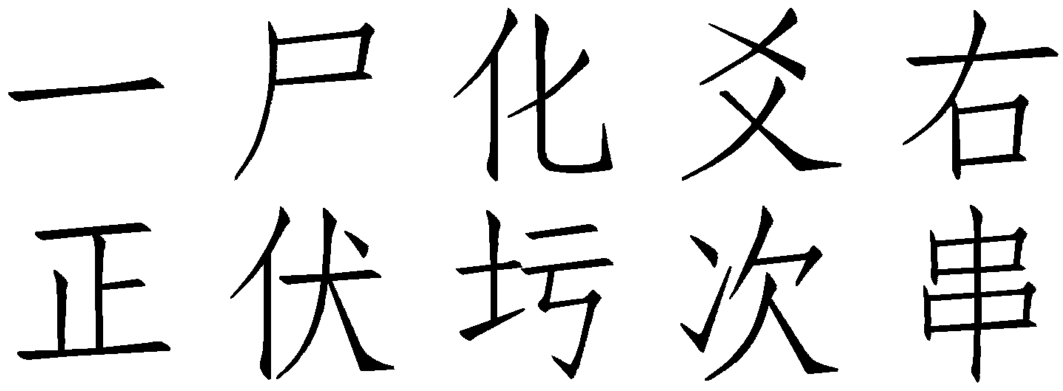

Figure 1: A small sample of images in a database of 500 chinese characters. The images are bitmaps.

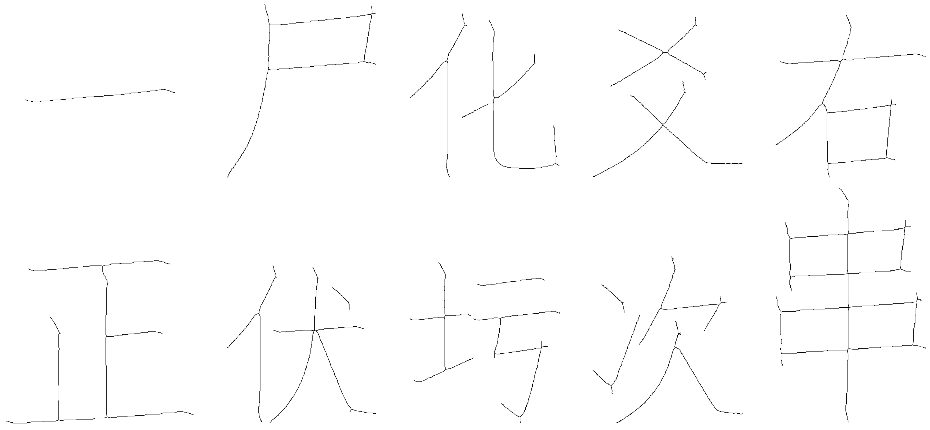

Figure 2: The shape summary of a chinese character bitmap is the medial axis of the set of black pixels which define the character. Here we show the summaries for the characters in figure 1.

approximate the medial axis pixel chains by straight segments. Consider the error in approximating the polygonal chain between start vertex u and end vertex w by the line segment $\overline{uw}$ connecting u and w. If this error is within a given bound, then reset w to the vertex right after w in the chain, and try the next segment. If the error exceeds the given tolerance, then approximate the chain from u to the vertex v just before w by the line segment $\overline{uv}$, reset the start vertex u to v, and try to find a line segment approximation starting from the new u. A reasonably high error bound was used in order to segment the medial axis chains into a small number line segments.

3 Accounting for Transformations Using Geometric Hashing

As mentioned in the introduction, similarity of images is considered with respect to some given transformation group $\mathcal{G}$. Ideally, a transformation of the underlying image will cause the same transformation of the corresponding illustration and projected illustration. We cannot directly compare two projected illustrations to judge image similarity. Instead, we derive an invariant feature set from the projected illustration using geometric hashing. This technique will produce the same feature set given projected illustrations S and $g(S)$, where $g \in \mathcal{G}$.

Geometric hashing is a method used to compare two point sets under some transformation group. Usually, the method is applied to finite point sets $P = \{\ p_1, \ldots, p_m\ \}$ and $Q = \{\ q_1, \ldots, q_n\ \}$. We illustrate the basic idea with the case of comparing P and Q under the group of translations. Consider the sets

$$\begin{aligned} I_i(P) &= \{\ p_k - p_i\ :\ 1 \leq k \leq m,\ k \neq i\ \} \qquad \text{and} \\ I_j(Q) &= \{\ q_l - q_j\ :\ 1 \leq l \leq n,\ l \neq j\ \}. \end{aligned}$$

Note that $I_i(P)$ and $I_j(Q)$ are invariant under translation of P and Q, respectively. If translating the set

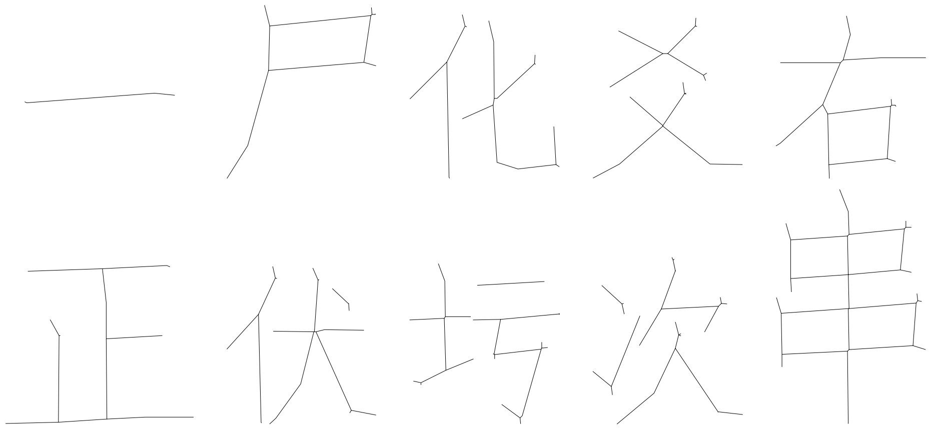

Figure 3: The chinese character illustrations are projected onto a basis with a line segment as the only basic shape. Here we show the projections for the character illustrations in figure 2.

P by $q_j - p_i$ produces a good match between P and Q, then the two sets $I_i(P)$ and $I_j(Q)$ will match well. The method can be made robust to missing data by comparing the translation invariant sets

$$I(P) = \bigcup_{i=1}^{m} I_i(P) \qquad \text{and} \qquad I(Q) = \bigcup_{j=1}^{n} I_j(Q).$$

In words, each of the points of P is recorded in $m-1$ different coordinate systems. The ith coordinate system has the same orientation and scale as the coordinate system of P, but its origin is at the point p_i. To compare P to Q, we compare $I(P)$ to $I(Q)$. Note that the sizes $|I(P)| = m(m-1) = O(m^2)$ and $|I(Q)| = n(n-1) = O(n^2)$ are quadratic in the original set sizes.

The details for the translation case can be generalized to other transformation groups. The general idea is to use subsets of P as bases in which to record all the other points in P. Ordered pairs of points (p_i, p_j) define the basis in the case of similarity transformations. The segment $\vec{p_i p_j}$ plays the role of the unit interval e_1 from $(0,0)$ to $(1,0)$ in recording the other points of P with respect to (p_i, p_j). More precisely, let T_{ij} be the transformation which maps $\vec{p_i p_j}$ to e_1. Then recording the point p_k with respect to (p_i, p_j) means recording $T_{ij}(p_k)$. The total number of points in the invariant set $I(P)$ is $O(m^3)$ in this case. For affine transformations, an ordered triple (p_i, p_j, p_k) defines the basis in which to record the other points in P. If T_{ijk} is the transformation that maps (p_i, p_j, p_k) to the vertices $(0,0)$, $(0,1)$, and $(1,0)$ of the right triangle Δ_1, then recording p_l with respect to (p_i, p_j, p_k) means recording $T_{ijk}(p_l)$. The total number of points in the invariant set $I(P)$ is $O(m^4)$ in this case.

The projected illustrations for our chinese character database are sets of segments. Although we do not have finite point sets, we can still apply the idea of geometric hashing to obtain a feature set which is invariant to a similarity transformation of the projected illustration. This is done by allowing each segment in the projected illustration P to play the role of the unit interval e_1. If P contains m segments, then we will have $2m$ different coordinate systems in which to record the segments in P (the factor of two is from considering both orderings of the segment endpoints). Therefore, using segment endpoints as basis point pairs leads to an invariant set $I(P)$ of $O(m^2)$ segments. The set $I(P)$ consists of m copies of P at varying scales, locations, and orientations. The overlap that occurs among these copies makes it very difficult to see the individual copies. A picture of some $I(P)$ from the chinese character database is not very informative, and hence no figure is provided.

4 Searching the Database

The final preprocessing step is to build a nearest-neighbor search structure on the feature space. When a query is given, its features are extracted in exactly the same manner as for the database images. For each query feature, we query the nearest-neighbor search structure for the k nearest database features to the query feature. Each time a database image has a feature which is close to a query feature, its similarity score is increased. As the similarity scores are updated, the R greatest image scores (and corresponding images) are tracked. Once all the query features have been processed, the R images with the highest similarity scores are returned.

An ideal situation for finding nearest database features is when the database features are points in a low-dimensional space, and the the L_2 distance between points measures feature dissimilarity. In this case, a standard Euclidean-space nearest-neighbor search strategy may by employed. The features in the chinese character database are the line segments in the invariant projected illustration. What is an appropriate distance measure between two line segments? We might, for example, use the Hausdorff distance between two line segments. There is work [Yianilos, 1993, Brin, 1995] on nearest-neighbor searching in general metric spaces (i.e. using only the distance between two objects). Here we opt for a simpler, ad hoc approach which embeds the segments as points in a four-dimensional Euclidean space. A directed line segment is specified by a quadruple

$$(l, \theta, a, b),$$

where l is length of the segment, θ is the angle the segment makes with the horizontal, and (a, b) is the position of the first endpoint. Note that the units of the components are different, so it does not make sense to use the L_2 distance unless we first normalize the components. Toward this end, we compute the standard deviations σ_l, σ_θ, σ_a, σ_b of the four component values over a large sample of database features. We use the L_2 distance between the normalized point features

$$\left(\frac{l}{\sigma_l}, \frac{\theta}{\sigma_\theta}, \frac{a}{\sigma_a}, \frac{b}{\sigma_b}\right)$$

as a measure of feature dissimilarity.

Our choice for a Euclidean nearest-neighbor searching algorithm is due to Arya, Mount, et.al. in [Arya *et al.*, 1994]. The algorithm preprocesses a set $S \subset \mathbf{R}^d$ of n points in $O(n \log n)$ time and $O(n)$ space, so that the k nearest neighbors to a given query point q can be computed in $O(k \log n)$ time. We apply the algorithm to find k nearest features to a given query feature, where $k = 32$ (in this setting, $d = 4$). If we let F denote the total number of database features and f_Q is the number of features in the query Q, then our query time is $O(f_Q \log F)$. For our 500 image database with the features extracted as previously described, a typical query takes roughly one second on an SGI Indy. Some sample queries are shown in figure 4.

5 Some Problems for Future Work

Our feature-based algorithm uses a very *one-way* notion of distance. A query and database image are similar whenever the database image has many of the same features as the query. There is no penalty for extra information in a database image which might cause it to look quite a bit different from the query. A possible solution to this problem involves tagging each feature point with both the image and basis points that produced it. This will allow us to estimate the transformation which makes the query match a particular database image, as well as the fraction of unmatched arclength in the database image.

Unfortunately, there is a more serious problem with our overall approach. The similarity score depends heavily on the segment decompositions of the projected illustrations – and these decompositions are not canonical. A high similarity score will be obtained iff there is a similarity transformation of Q that makes many segments in Q match well many segments in P, where two segments match well iff both pairs of endpoints are close. This fact is due to our use of pairs of segment endpoints as bases in the geometric hashing step. In essence, we are judging the similarity of the representations of the projected illustrations instead of the projected illustrations themselves. This *representation problem* will be the subject of future research.

A third problem is that our geometric hashing strategy produces too many features to index. If there are m segments in a projected illustration, the invariant projected illustration will have $O(m^2)$ segments. This brute force approach is motivated by the fact that we want to be able to match a subset of the query to a subset of a database illustration without making any *a priori* assumptions about features that are likely to appear in both the database illustration and a similar query illustration. Suppose, instead, that we record segments only with respect to the c longest segments in the set, where c is a small constant. This strategy produces an invariant projected illustration with only $O(m)$ segments, but it assumes that a long segment in a database image is likely to appear as a long segment in a similar query. Thus, the representation problem remains.

Acknowledgements

We would like to thank Guillermo Sapiro for recommending the medial axis code package based on [Ogniewicz and Kübler, 1995], and Carlo Tomasi for carefully reading the manuscript and providing useful comments.

References

[Arya *et al.*, 1994] S. Arya, D.M. Mount, N.S. Netanyahu, R. Silverman, and A.Y. Wu. An optimal algorithm for approximate nearest neighbor searching. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 573–582, 1994.

[Brin, 1995] S. Brin. Near neighbor search in large metric spaces. In *Proceedings of VLDB '95. 21st International Conference on Very Large Data Bases*, pages 574–584, 1995.

[Cohen and Guibas, 1996] Scott D. Cohen and Leonidas J. Guibas. Shape-based illustration indexing and retrieval - some first steps. In *Proceedings of the ARPA Image Understanding Workshop*, pages 1209–1212, February 1996.

[Cohen and Guibas, 1997] Scott D. Cohen and Leonidas J. Guibas. Partial matching of planar polylines under similarity transformations. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 777–786, January 1997.

[Etemadi, 1992] A. Etemadi. Robust segmentation of edge data. In *International Conference on Image Processing and its Applications*, pages 311–314, April 1992.

[Guibas and Tomasi, 1996] Leonidas J. Guibas and Carlo Tomasi. Image retrieval and robot vision research at Stanford. In *Proceedings of the ARPA Image Understanding Workshop*, pages 101–108, February 1996.

[Lamdan and Wolfson, 1988] Yehezkel Lamdan and Haim J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Second International Conference on Computer Vision*, pages 238–249, 1988.

[Lowe, 1987] D.G. Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31:355–395, 1987.

[Niblack *et al.*, 1993] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The QBIC project: querying images by content using color, texture, and shape. In *Proceedings of the SPIE*, volume 1908, pages 173–187, 1993.

[Ogniewicz and Kübler, 1995] R. L. Ogniewicz and O. Kübler. Hierarchic Voronoi skeletons. *Pattern Recognition*, 28(3):343–359, March 1995.

[Rosin and West, 1995] Paul L. Rosin and Geoff A.W. West. Nonparametric segmentation of curves into various representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(12):1140–1153, December 1995.

[Yianilos, 1993] P. Yianilos. Data structures and algorithms for nearest neighbor searching in general metric spaces. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, 1993.

Figure 4: Sample queries into the chinese character database. The left column shows the query image, while the right column shows the results window. Each query takes about one second on an SGI Indy.