Shape-based Illustration Indexing and Retrieval Some First Steps

Scott D. Cohen   *Leonidas J. Guibas*
Computer Science Department
Stanford University
Stanford, CA 94305

Abstract

We propose a general set of ideas for indexing technical illustrations based on the shapes present in them, so that they can be efficiently retrieved later using as the key other 'similar-looking' illustrations (either pre-existing, or interactively drawn by the user). A very simple prototype system demonstrating these ideas was implemented and is described in this note. The general scheme is to select a class of *basic shapes* and record in the index where (more precisely, via what homothetic map) these basic shapes match well into the illustration. The current implementation indexes using line segments as the only basic shape. A Hausdorff matcher is then used to compute the best alignment of the basic shape matches between a query and an illustration, thus giving us a measure of the distance between the two. Currently every illustration in the data-base is matched individually against the query, though sublinear algorithms are under investigation. A library of approximately two-hundred illustrations from a geometry textbook was indexed using this scheme and then used for retrieval experiments. An interactive interface was provided for specifying the data-base to be searched and the query illustration, for setting various parameters regarding the match, and for displaying the best matches found in the data-base.

1 Background

As the amount of pictorial information available digitally increases, it becomes ever more important to develop techniques for allowing users to index and browse through such data. The pictorial information can be images or video sequences acquired through cameras, as well as synthetic illustrations, diagrams, charts, or graphics created with the aid of the computer. Pictorial data in all these forms can appear in documents and it is clearly important to develop methods for navigating, searching, and browsing through images and illustrations as effectively as one can now do with text.

There are numerous modalities of pictorial data that can help in this task of indexing and retrieval. Such are, for example, color, shape, and texture. Depending on the type of data, the significance of these modalities can vary, as well as the ease of extracting the parameters of the various indices we may select. In general this task will be easier for synthetic images where we have a model of the objects being represented and of the imaging process that was used. But even for synthetic images the selection of suitable indices can be a challenging problem, as most of the time we are trying to search for and retrieve pictorial data that is 'visually similar' to other pictorial data we may have. Our measure of similarity may have to allow for changes in object pose and illumination, in location and scale, still work despite partial occlusions and missing data, etc.

In the work reported here we have chosen to focus on the use of *shape information* for the tasks of indexing and retrieval. Clearly shape is a universally useful modality for describing, and therefore indexing, pictorial data. We have restricted out attention to the domain of computer-generated technical illustrations, where shape information is both precisely available and the main way in which pictorial meaning is conveyed. We are confident that after we have techniques that can operate successfully in this domain, we will be able to port them to other kinds of pictorial data as well by applying shape extraction techniques from computer vision.

2 Shape Indexing for Illustrations

There is a vast literature in computer vision on measures of shape similarity and algorithms for computing the distance between shapes under various transformation groups. We will not attempt to survey this literature here, in part because our requirements are somewhat different. Our goal is, given an illustration P, to compute a compact index $\iota(P)$ which records the shapes present in P and their location. Given a collection of illustrations $P_1, P_2, \ldots, P_n$, we wish to compute a data-structure $\mathcal{D}$ for recording $\iota(P_1), \iota(P_2), \ldots, \iota(P_n)$ so that retrieval queries can be answered efficiently. At retrieval time we assume that we will be given another illustration Q, either preexisting or interactively drawn by the user. In our scheme we do retrieval by computing $\iota(Q)$ and then searching $\mathcal{D}$ for the illustrations P_i whose index $\iota(P_i)$ is 'similar' to $\iota(Q)$.

An illustration P for us is a collection on instanced graphics primitives (lines or polylines, circular arcs, Bézier cubic or B-spline arcs, marks, etc.), as is almost universally the case with the illustrators in common use today (e.g., Adobe Illustrator, Aldus Freehand, Xfig, etc.). Usually these graphics primitives are grouped together into 'objects' which can then be manipulated by the illustrator as a single graphical unit. We assume that we have access to all this information for computing our index $\iota(P)$, though at the present time we do not make any use of the grouping or hierarchical information present. We also assume that the query illustration Q is given to us in the same format as the data-base illustrations.

Our key notion for indexing an illustration P is to say 'which shape appears where' in P. We start with a a collection of *basic shapes* $S_1, S_2, \ldots, S_k$ – these may be built-in, or user-definable. Our index $\iota(P)$ records the significant matches of shapes S_i in P. In matching we allow each shape S_i to translate, rotate, and scale, though more general affine or projective mappings may also make sense in certain contexts. Because we regard basic shapes as simple, in our current design we match each basic shape S_i separately against each high-level graphics primitive present in P (e.g, each polyline or each B-spline). This might miss, for example, a match of a 'letter S'-like basic shape against a 'letter-S' figure formed by the juxtaposition of two separately instanced circular arcs. We have chosen to ignore such matches both because they would make our matching algorithms much more complex, but also because the high-level graphics primitives present in illustrators today makes it unlikely that such 'accidental' shapes in an illustration carry a lot of the user-intended meaning.

We will say that a basic shape S_i matches well into the graphic primitive element G_j of P if S_i can be translated, rotated, and scaled (above a certain minimum size σ) so that a large fraction α of it fits within G_j 'fattened' by a disk of radius ϵ, for some small $\epsilon > 0$ – this corresponds to a good one-directional Hausdorff match [1]. We are currently investigating algorithms for finding such good matches efficiently for different basic shapes S_i (line segments, circular arcs, corners, etc.).

Note that there still several unresolved issues:

1. The above discussion applies primarily to 'line-like' graphics primitives and basic shapes. Illustrators also provide filled areas, however, which have to be treated somewhat differently.
2. There is the issue of whether the shape library $S_1, S_2, \ldots, S_k$ tries to capture global (e.g., a large circular arc) or local (.e.g, a small cusp) features of P. Clearly matching

large features of P can be done more reliably. However, small features may be more useful in the index, as their presence can be detected even though the overall shape they come from is partially occluded, etc. A related issue is how to compare a match of basic shape S_i of large scale but possibly worse quality (larger ϵ) to one of small scale but better quality.

3. In general there will be many good matches of a given basic shape S_i into an illustration P and our goal is to choose a small representative subset. For example, near any very good match there will be other (less) good matches that are dominated by it. Clearly we do not want to record the latter in the index. There can also be situations, as when matching a circular arc onto a circle, where there is a continuum of equally good matches.

To summarize the key point of this section: for every illustration P we compute an index $\iota(P)$. This index is a list of the basic shapes — a subsequence of $\{S_1, S_2, \ldots, S_k\}$ — which match well into P. For each such matching basic shape S_i we record a list of transformations $T_1^{(i)}, T_2^{(i)}, \ldots, T_{i_t}^{(i)}$ that correspond to the significant matches of S_i into P. Each transformation is encoded by four parameters: two for a translation and one each for rotation and scaling. We typically expect that i_t will be a very small integer.

3 Shape Retrieval

Although a key goal of our work is to allow comparison of a query illustration Q against a data-base of illustrations $P_1, P_2, \ldots, P_n$, without explicitly comparing $\iota(Q)$ to each of $\iota(P_1), \iota(P_2), \ldots, \iota(P_n)$ separately, we have not achieved this yet. We are optimistic that we will be able to attain sublinear query-time algorithms by using computational geometric techniques on the set of indices — essentially by clustering illustrations whose indices have a small 'distance' from each other. But even the problem of comparing $\iota(Q)$ to $\iota(P)$ for a single P is sufficiently interesting that it has taken the bulk of our attention so far.

Recall that when the user specifies the query illustration Q, $\iota(Q)$ is computed by the same algorithm as $\iota(P)$. Nevertheless, we cannot expect in general that $\iota(Q)$ and $\iota(P)$ will be identical. For one, the user may draw in Q only certain key or dominant shapes he/she knows to be present in P, so $\iota(Q)$ will fequently contain less information than $\iota(P)$. In addition, we must allow for certain variability in the location, orientation, and scale of shapes between Q and P, and this will affect the way the corresponding matching transformations T for the same basic shape S_i are recorded in $\iota(Q)$ and $\iota(P)$. If T_Q and T_P denote two corresponding match transformation in $\iota(Q)$ and $\iota(P)$, then T_Q and T_P may themselves differ by a certain translation, rotation, and scaling, reflecting the above mentioned variability.

It is part of our design to provide an interface in which the user can indicate acceptable bounds on this translation, rotation, and scale variability between Q and P. When we start exploiting the hierarchical structure of illustrations, it will also be our goal to allow the user to specify different variability bounds at different levels of the hierarchy so that, for example, matches generated from graphic elements of the same object must all be variable by nearly the same amount, while only looser correspondences are required across objects. As an example, the illustration P may contain two objects, say, a house and a car. Each of those may consist of several graphic elements which are reproduced fairly accurately in the query for each object, but the relative position of the house and the car might be off by a fair amount.

To match $\iota(Q)$ and $\iota(P)$ we regard each transformation T in each of them as a 'colored' point in $\mathcal{R}^4$. The four dimensions correspond to x and y (the translation amount), θ (the rotation), and $\log s$ (logarithm of the scale). The 'color' is just the index i of the basic shape S_i for which T records a good match. The reason for recording the logarithm of the scale is that in this way a change of scale corresponds to a translation along the fourth coordinate, in a way exactly analogous to the other three. We can then define the 'distance' from $\iota(Q)$ to $\iota(P)$ in $\mathcal{R}^4$ to be the colored one-way Hausdorff distance under translation between the two sets, where the translation may be restricted to be inside the parameter ranges specified by the user (by 'colored distance' we mean that in the match only points of the same color can be matched). Note that in order to define the magnitude of a translation in $\mathcal{R}^4$ we will need to select some coefficients for the relative importance of translation, rotation, and scale variability.

4 The Implementation

We have implemented the simplest possible version of such an indexing scheme for illustrations – in fact the bulk of our programming effort was expended in developing Tcl/Tk widgets for the required user interface, and for adapting the illustrator we chose to use to our needs. We chose **Ipe** (Interactive Picture Environment), an illustrator developed by Otfried Schwarzkopf and others at the Informatics Department of the University of Utrecht in the Netherlands [4]. The reasons for selecting Ipe were that (1) the entire code for Ipe is publicly available, (2) Ipe has a nice extension mechanism built in, (3) the format in which Ipe files are stored is very close to Postscript, and in fact Ipe can read arbitrary Postscript files (thus in principle we have a path to try our techniques on arbitrary collections of Postscript illustrations), and (4) we had available a corpus of approximately two hundred illustrations from a new textbook in computational geometry authored by Mark de Berg, Mark Overmars, Otfried Schwarzkopf, and Marc van Kreveld in Utrecht.

Figure 1 below shows some typical illustrations from this textbook.

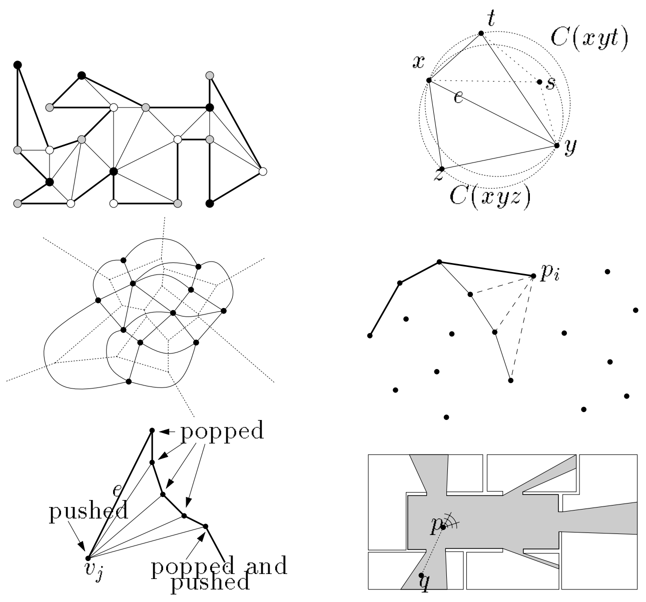


Figure 1: Example illustrations from the textbook

As these examples make clear, polygonal lines predominate in this corpus. We chose to do a very simple implementation, indexing illustrations using only a single basic shape, a line segment. Even the line segment matching algorithm we have used against the polyline graphics primitive in the Ipe files is rather unsophisticated, but due to lack of space we omit details. We did not have at our disposal a four-dimensional Hausdorff matcher, but we did have a two-dimensional matcher developed by William Rucklidge of PARC [2]. So when we match indices, we give the user the option of using scale and orientation only, or translation only.

An example of a very simple query is shown in Figure 2. Figure 3 shows the interactive widgets for the match parameter settings while Figure 4 shows the window where the best matching illustrations are returned.

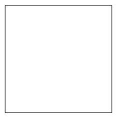
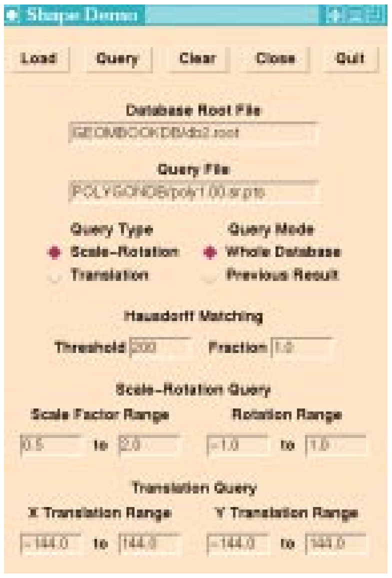


Figure 2: An example query drawn using Ipe



Database Root File: GEOMBOOKDB/db2.root

Query File: POLYGONDB/poly1.00.sr.pts

Query Type: [✓ Scale-Rotation] Translation

Query Mode: [✓ Whole Database] Previous Result

Hausdorff Matching

Threshold 200 Fraction 1.0

Scale-Rotation Query

Scale Factor Range 0.5 to 2.0 Rotation Range -1.0 to 1.0

Translation Query

X Translation Range -144.0 to 144.0 Y Translation Range -144.0 to 144.0

Figure 3: Example of the graphical display for setting match parameters

Note that the user can set the 'threshold' for the match (the ϵ of the previous discussion) and match 'fraction' (the α). Not surprisingly, matching with scale and orientation proved to be much more useful than with translation — we can think of our index as recording the size and orientation of the sufficiently long line segments we find in the illustrations. We use these 2-d distance estimates between the query and each of the data base images to select and order the set of candidate matches for display to the user.

5 Conclusions

Though the present software is only a rudimentary demonstration of the capabilities which a shape-based indexing system might provide, we feel that there is potential for some useful technology to be developed along the lines of this approach and we intend to pursue it further.

Acknowledgements: The original framework for the ideas presented below was developed at Stanford University under an ARPA-funded project titled 'Image Browsing and Retrieval' with Professors Leonidas Guibas and Carlo Tomasi as PIs. The implementation described above was carried out during the summer of 1995 at PARC by Scott Cohen, a graduate student working under the supervision of Leonidas Guibas. The authors grate-

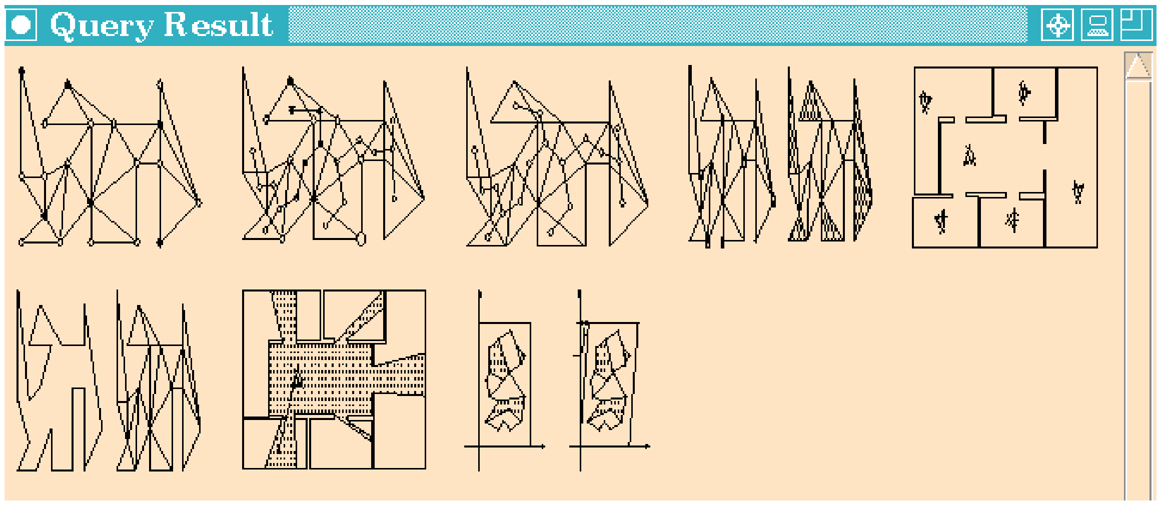


Figure 4: Example of the graphical display for returned matches

fully acknowledge the support provided by ARPA grant DAAH01-95-C-R009 and by Xerox PARC.

References

[1] L.P. Chew, M.T. Goodrich, D.P. Huttenlocher, K. Kedem, J.M. Kleinberg, and D. Kravets. Geometric Pattern Matching under Euclidean Motion. In *Proceedings of the Fifth Canadian Conference on Computational Geometry*, pages 151–156, 1993.

[2] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. Comparing Images Using the Hausdorff Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, September 1993. Code available from ftp://cs.cornell.edu/pub/wjr/Hausdorff.tar.gz.

[3] K. Mehlhorn. *Data Structures and Algorithms 3: Multi-dimensional Searching and Computational Geometry.* Springer-Verlag, 1984.

[4] O. Schwarzkopf. *A Manual of Ipe.* The Ipe package is available from ftp://ftp.cs.ruu.nl/pub/X11/Ipe/.