Efficient Retrieval by Shape Content

Euripides G.M. Petrakis*
Dept. of Electr. and Comp. Engineering
Technical University of Crete, Chanea, Greece
petrakis@ced.tuc.gr

Evangelos Milios†
Department of Computer Science
York University, Toronto Canada
eem@cs.yorku.ca

Abstract

We propose an approach for image similarity retrieval based on shape information. The heart of our methodology is a dynamic programming shape matching algorithm which detects similarities between shapes at various levels of shape detail (resolution). Our method provides for indexing of a data set, achieving up to three orders of magnitude speed-up over sequential scanning. We illustrate the application of our method to real two-dimensional static hand gesture data. We also demonstrate the superiority of our approach over traditional methods of shape matching and retrieval, such as Fourier descriptors, Geometric and Sequential moments.

1. Introduction

Object recognition is an important problem in computer vision and has received considerable attention in the literature. Most approaches to object recognition in the Computer Vision research are *model-based* [1] emphasizing on the accuracy of recognition. However, the increasing amounts of image data in many application domains has generated additional interest for real-time management and retrieval of shapes [7]. There, the emphasis is not only on accuracy, but also on efficiency (i.e., speed) of retrieval.

In this work, we deal with the following general problem:

- We have a collection of N two-dimensional closed shapes (i.e., hand gestures in our case).
- Given a query shape, we want to find the most similar shapes or all shapes below a distance threshold t.
- We need to respond to such queries *fast* that is, faster that sequential scanning.

We summarize the contributions of this work in the following:

- We propose a shape matching algorithm which detects similarities between shapes at various levels of detail.
- We establish the superiority of our algorithm over traditional methods to shape matching and retrieval [7].
- Our method allows for indexing, achieving up to three orders of magnitude speed-up over sequential scanning.

The rest of this work is organized as follows: The proposed matching algorithm is discussed in Section 2. Our approach to indexing and retrieval is presented in Section 3 followed by experimental results in Section 4 and conclusions in Section 5.

2. Shape matching

The shape matching algorithm that lies at the core of our methodology takes in two shapes and computes (a) Their distance and (b) The correspondences between similar parts of the two shapes.

2.1. Definitions

Let A and B be the two shapes to be matched and let $A = a_0, a_1, \ldots a_{N-1}$ and $B = b_0, b_1, \ldots b_{M-1}$ be the convex/concave segment sequences of the two shapes, with a_i being the segment between two consecutive inflection points (i.e., points of change of curvature) p_i and p_{i+1}; similarly for b_j. Then, $a(i-n|i)$, $n \geq 0$, is the sequence of segments $a_{i-n}, a_{i-n+1}, ..., a_i$; similarly for $b(j-m|j)$, $m \geq 0$.

The algorithm searches for segment correspondences at various levels of shape detail by allowing matching of merged sequences of consecutive segments, if this leads to the minimization of a cost function. Each merging is a recursive application of the grammar rules $CVC \rightarrow C$ and $VCV \rightarrow V$, where C and V denote convex and concave segments respectively [8]. The number of merged segments is always odd. A *complete match* is a correspondence between groups of consecutive segments in order, such that no segments are left unassociated.

*This work was carried-out while the author was visiting York Univ.

†This work was supported by a grant from the Natural Sciences and Engineering Research Council of Canada.

The goal is to find the *best* association of segments in shape A to segments in shape B. This is formulated as a minimization problem which is solved efficiently by Dynamic Programming (DP): A table of partial costs is built and the optimal matching is searched in the form of a path in the DP table that minimizes a dissimilarity cost.

The DP table has $2N$ columns and $2M$ rows, corresponding to segments of shape A and shape B respectively repeated twice (to force the algorithm to consider all possible relative rotations between the two shapes). A link between cells (i_{w-1}, j_{w-1}) and (i_w, j_w) denotes the matching of segments $a(i_{w-1}+1|i_w)$ with $b(j_{w-1}+1|j_w)$. A *path* is a linked sequence of cells $(i_0, j_0), (i_1, j_1), ..., (i_w, j_w)$, not necessarily adjacent, indicating a partial match. A complete match has $(i_0, j_0) = (i_w, j_w)$. Function $\psi\left(a(i_{w-1}+1|i_w), b(j_{w-1}+1|j_w)\right)$ represents the dissimilarity cost between its two arguments and is defined in Section 2.3.

Each cell $cell(i, j)$ in the DP table contains the cost array $g[k_w]$ and associated bookkeeping data $t_1[k_w]$, $t_2[k_w]$, $index[k_w]$, $g_n[k_w]$, $g_m[k_w]$, where k_w varies from 0 to $K-1$ and refers to the k_w-th best path (k_w-th option), or partial match, up to and including $cell(i_w, j_w)$. Each cell keeps up to K best paths. Specifically, $g[k_w]$ holds the cost of the path, $t_1[k_w]$ and $t_2[k_w]$ hold the number of unmatched segments in shapes A and B respectively for the path and $index[k_w]$, $g_n[k_w]$ and $g_m[k_w]$ hold the back links for the path and allow the backward tracing of the path.

If $g_n[k_w] = n_w$ and $g_m[k_w] = m_w$ for $cell(i_w, j_w)$, then the previous cell and option in the path is $cell(i_{w-1}, j_{w-1}) = cell(i_w - 2n_w - 1, j_w - 2m_w - 1)$ and $index[k_w]$ respectively, where n_w and m_w are nonnegative integers. Notice that, $i_{w-1} = i_w - 2n_w - 1$ and $j_{w-1} = j_w - 2m_w - 1$, since the number of segments which are merged is always odd.

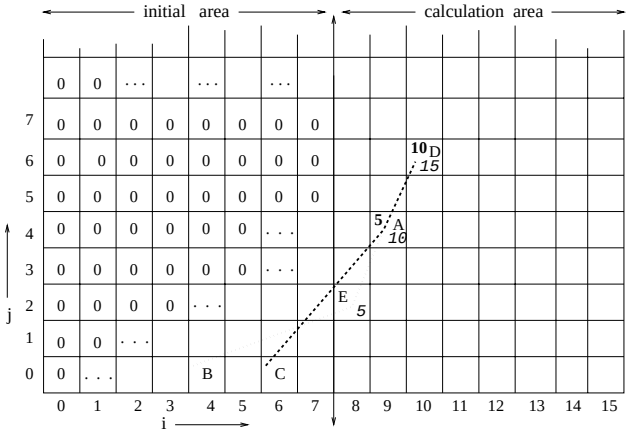

Figure 1. Example of a DP table.

Figure 1 illustrates an example of a DP table computed for the matching of two shapes with 8 and 4 segments respectively. Two incomplete paths are shown ending at $cell(10, 6)$. Numbers in cells indicate accumulated costs. The DP table consists of the initial value area (left half) and the calculation area (right half). In the initial value area all g terms are initialized to zero, t_1 to N and t_2 to M, implying that each of these cells can act as the first cell in a path. The calculation area is computed and finally the optimal path is searched. A path is complete when its corresponding t_1 and t_2 at the final cell of the path, both become zero simultaneously.

2.2. The algorithm

The main idea in the algorithm is to fill the DP table cells and then search for and trace back the optimal complete path. We outline the algorithm below.

for $i_w = N, N+1, \ldots 2N-1$ **do**
  for $j_w = 0, 1, \ldots 2M-1$ **do**
    fill the K options in $cell(i_w, j_w)$;
    check if a complete path has been found;
  end for
end for
find the complete path with the lowest cost;
retrace its shape match by following backward links;

The *for* loop for j_w does not run over all the indicated values, but only over those values that do not involve convex to concave segment associations, which are not possible. To describe the filling of $cell(i_w, j_w)$ of the DP table with values, we need the following auxiliary functions:

- $xmin(list, x)$ extracts the x smallest terms from $list$.
- $listat(i_w, j_w, n_w, m_w)$ is the list of the costs of all options at $cell(i_{w-1}, j_{w-1})$, augmented by the dissimilarity cost $\psi(a(i_{w-1}+1|i_w), b(j_{w-1}+1|j_w))$ of extending them to $cell(i_w, j_w)$.
- $fmin(f, i)$ returns the integer pair (n, m) that leads to the i-th smallest result in the evaluation of a given function $f(n, m)$.

The function f required for filling the k_w-th option of $cell(i_w, j_w)$ is

$$f(n,m) = g(i_w - 2n - 1, j_w - 2m - 1, k_w) + \psi\left(a(i_w - 2n|i_w), b(j_w - 2m|j_w)\right), \quad n, m \geq 0. \tag{1}$$

With the above definitions, we fill the cost array $g[k_w]$ at $cell(i_w, j_w)$ with the K values computed by the following loop:

for $x = 0, 1, \ldots K-1$ **do**
  collect $xmin(listat(i_w, j_w, fmin(f, x)), K)$;
end for

Function $fmin(f,i)$ searches for n's over the range $[0, \frac{N-1}{2}]$ and for m's over the range $[0, \frac{M-1}{2}]$ to form acceptable pairs (n,m).

2.3. Cost components

The cost term ψ in Equation 1 can be rewritten as $\psi(a(i_{w-1}+1|i_w), b(j_{w-1}+1|j_w))$. Following the notations of [10], this cost term consists of three additive components:

$$\begin{aligned}\psi\left(a(i_{w-1}+1|i_w), b(j_{w-1}+1|j_w)\right) = \\ MergingCost(a(i_{w-1}+1|i_w)) + \\ MergingCost(b(j_{w-1}+1|j_w)) + \\ \lambda\, DissimCost\left(a(i_{w-1}+1|i_w), b(j_{w-1}+1|j_w)\right),\end{aligned} \quad (2)$$

where λ represents the relative importance of the merging and dissimilarity costs. In this work λ was set to 1.

The first two terms in Equation 2 represent the cost of merging segments $a(i_{w-1}+1|i_w)$ in shape A and segments $b(j_{w-1}+1|j_w)$ in shape B respectively while, the last term is the cost of associating the merged $a(i_{w-1}+1|i_w)$ with the merged $b(j_{w-1}+1|j_w)$.

Requirements for reliable cost computation are the following:

- Merging should follow the process grammar rules [8] (i.e., each allowable merging should be a recursive application of the grammar rules $CVC \Rightarrow C$ and $VCV \Rightarrow V$). This is enforced by the DP algorithm.
- Merging a *"visually prominent"* segment (i.e., a large segment with high curvature) into a merged segment of the opposite type (convex or concave) should incur a high cost. To specify this requirement, we need to define *visual prominence* in geometric terms.
- The partial cost components arising from different features of the shape should be combined into a total cost in a meaningful way.

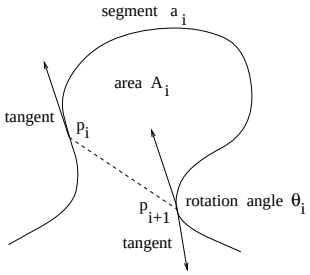

Figure 2. Geometric quantities for defining the prominence of a segment

The heuristic cost computations that follow attempt to satisfy the above requirements. Firt, we define geometric quantities (features) needed in the specification of visual prominence of a segment according to Figure 2.

Rotation Angle θ_i is the angle traversed by the tangent to the segment from inflection point p_i to inflection point p_{i+1} and shows how much a segment is curved.

Length l_i is the length of segment a_i.

Area A_i is the area enclosed between the chord and the arc between the inflection points p_i and p_{i+1}.

Dissimilarity cost computation: Assigns a higher cost to segments (or groups of segments) with large differences in more than one feature:

$$DissimCost = W \max_{all\ features\ f}\{d_f\}, \quad (3)$$

where $f = l, \theta$ or a. Factor W equals the number of features for which d_f is greater than $0.75 \times \max\{d_f\}$, where $f = l, \theta, a$. Thus, if all three features have uniformly large d_f, then the dissimilarity cost is multiplied by 3.

The term d_f, for $f = \theta$ is defined as

$$d_f = \left|\frac{\Theta_A - \Theta_B}{\Theta_A + \Theta_B}\right|, \quad (4)$$

where $\Theta_A = \sum_{s=i_{w-1}+1}^{i_w} \theta_s$, and $\Theta_B = \sum_{s=j_{w-1}+1}^{j_w} \theta_s$, and θ_s being the rotation angle of segment with index s of shape A and shape B respectively.

The term d_f, for f being l (length) or area (a), is defined as

$$d_f = \left|\frac{f_A}{F_A} - \frac{f_B}{F_B}\right|, \quad (5)$$

where $F_A = \sum_{s=0}^{N-1} f_s$, $f_A = \sum_{s=i_{w-1}+1}^{i_w} f_s$ of shape A and similarly for F_B, f_B of shape B.

Merging cost computation: Let the types of the segments being merged be $CVC \ldots VC$. The opposite case is obtained by switching C and V. The merging cost is defined as follows:

$$MergingCost = \max_{all\ features\ f}\{w_f c_f\}, \quad (6)$$

where subscript f refers to a feature (length, area or rotation angle).

For f being length or area:

$$c_f = 1 - \frac{\sum_{C\ segs\ of\ group} f - \sum_{V\ segs\ of\ group} f}{\sum_{all\ segs\ of\ shape} f}. \quad (7)$$

For f being rotation angle:

$$c_f = 1 - \frac{\sum_{C\ segs\ of\ group} f - \sum_{V\ segs\ of\ group} f}{\sum_{C\ segs\ of\ group} f + \sum_{V\ segs\ of\ group} f}. \quad (8)$$

The intuition behind these formulae is that they measure the visual prominence of the features of the absorbed segments (of type V) relative to the absorbing segments (of type C). All costs c_f are within the interval $[0, 2]$. Cost c_f is close to 0 if the convex segments visually dominate the concave ones (hence it is plausible to absorb the concave ones), while it is close to 2 if the concave segments visually dominate the convex ones (hence it is not plausible to perform the merge, therefore the merging cost should be high).

For f being any feature (length, area, rotation angle) the weight term is

$$w_f = \frac{N}{2} \frac{\sum_{V\ segs\ of\ group} f}{\sum_{V\ segs\ of\ shape} f}. \quad (9)$$

The intuition behind the weight term is to measure the visual prominence of the absorbed segments within the shape as a whole. Factor $\frac{N}{2}$ is heuristic.

2.4. A matching example

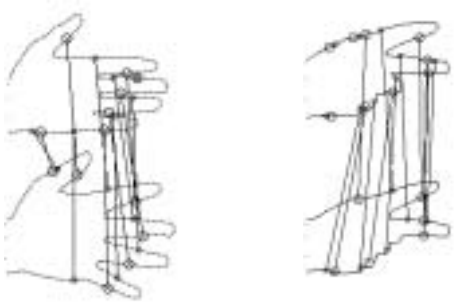

Figure 3. Segment associations reported by the matching algorithm.

Figure 3 illustrates the segments correspondences obtained by matching hand silhouettes obtained from different persons. The large circles indicate inflection points that are transitions from convex to concave and the small circles from concave to convex, when the shapes are traversed in a clockwise fashion.

3. Shape retrieval

The main idea behind our approach is to transform each shape to a point in an f-dimensional space (f to be determined). The mapping of shapes to f-dimensional points is achieved through FastMap [2]: The FastMap algorithm accepts as input N shapes, our distance function and f, the desired number of dimensions, and maps the above shapes to N points in an f-dimensional space. The complexity of this mapping is $\mathcal{O}(Nf)$ distance computations. Notice that this operation could (and should) be off-line. Similarly, when a query is given, it is quickly mapped to a point into the above f-dimensional space requiring only $\theta(f)$ distance computations. Then, the problem of IDB search is transformed into one of spatial search. To speed-up retrievals, the f-dimensional points are indexed using an R-tree [4]. Figure 4 illustrates the above sequence of operations.

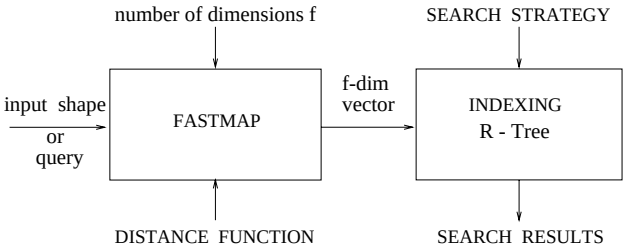

Figure 4. Mapping of shapes to f-dimensional points.

3.1. Shape indexing

Figure 5 demonstrates the file structure of the data on the disk. The file structure consists of the following parts:

- The *"shape file"* holding the original shapes along with their boundary contours.
- The R-tree holding an f-dimensional vector for each stored shape.

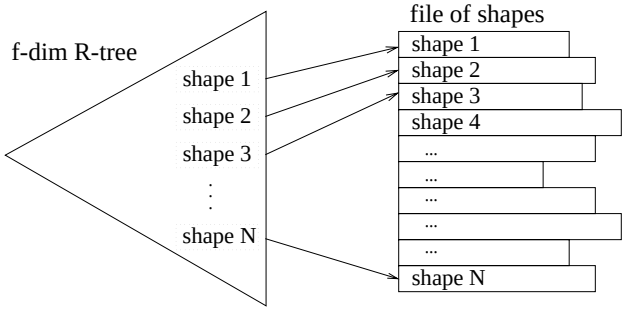

Figure 5. File structure.

3.2. Search strategy

The user specifies a query shape and a tolerance t and asks for all shapes within that tolerance. The f-dimensional vector

of the query is computed first using FastMap. Then, all vectors within distance t (range query) are retrieved from the R-tree. The R-tree may return *false alarms* (i.e., non qualifying shapes). A post-processing step is required to clean-up the false alarms. The generic search algorithm is as follows:

R-tree search: Issue a range query on the R-tree to obtain a list of promising shapes (their identifiers).

Clean-up: For each of the above obtained shapes, retrieve its corresponding contour from the shape file and compute the distance between this shape and the query. If the distance is less than the threshold t, the shape is included in the response set.

Nearest-neighbor queries, such as *"give me the 10 most similar shapes"*, can also be answered using the algorithm in [6].

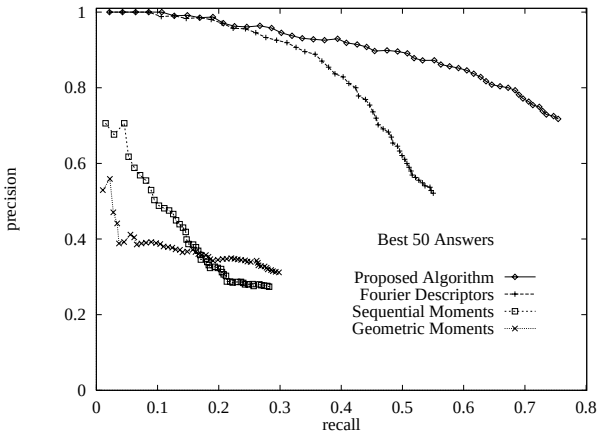

Figure 6. Precision-Recall diagram for (a) Our shape matching algorithm, (b) Fourier descriptors, (c) Sequential moments and (d) Geometric moments.

4. Experimental results

The experiments were designed to illustrate the:

Superiority of our shape matching algorithm over traditional algorithms such as those examined in [7].

Speed-up of our method over sequential scan searching.

Accuracy of our indexing method compared to sequential scanning. Our methodology exchanged a very small loss in accuracy (i.e., 2 - 10%) for much faster retrievals.

To test the efficiency of our methodology we used a data set consisting of 980 synthetic shapes which are generated from 17 original hand gestures using the shape morphing algorithm of [9]. Then, we took the above 17 shapes as queries. All measurements below correspond to averages over 17 queries.

4.1. Comparison with other methods

The competitors to our method are: (a) Fourier descriptors [11] (b) Sequential moments [3] and Geometric moments [5]. We used human relevance judgements to compute the effectiveness of each method. Two shapes (i.e., a query and a model shape) are considered similar if a human judges that they represent the same figure. To measure effectiveness, for each candidate method we computed:

Precision defined as the percentage of similar shapes retrieved with respect to the total number of retrieved shapes.

Recall defined as the percentage of similar shapes retrieved with respect to the total number of similar shapes in the database. Because we don't have the resources to compare every query with each database shape (i.e., this would require, for each method, $17 \times 980 = 16,660$ visual judgements!), for each query, we merged the answers obtained by all candidate methods and we considered this as the database which is manually inspected for relevant entries. This method allows for judgements such as *"method A returns 5% fewer correct answers than method B"*.

Each one of the 17 queries retrieves the best 50 shapes. For answers containing between 1 and 50 entries, we computed the average values of precision and recall. These values are represented in a *precision-recall plot*: The horizontal axis corresponds to recall and the vertical axis corresponds to precision. The total number of points in each curve is 50. The top-left point of the diagram corresponds to the first best match while, the bottom right point corresponds to the entire answer set. Each point is the average over 17 queries. Each method is represented by a curve.

Figure 6 illustrates the precision-recall diagram for this experiment. Our method performs clearly better than any other method, achieving up to 20% better recall and 20% better precision than the second best method (Fourier descriptors) for answers with 50 shapes.

4.2. Response time

In the following we study the speed-up effect of our indexing scheme over sequential scanning. Times are reported in number of distance computations that is, in number of calls

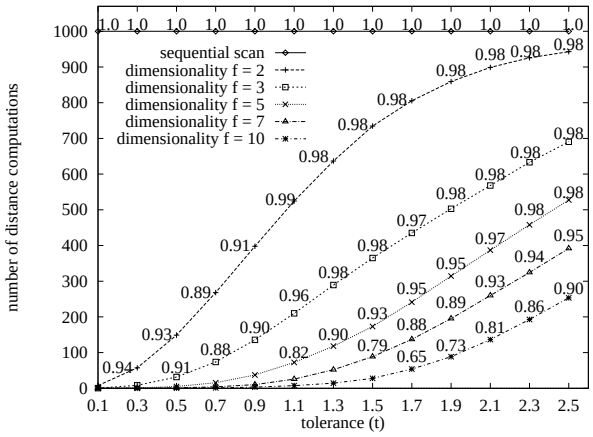

Figure 7. Average number of distance computations as a function of the tolerance t, corresponding to sequential scanning and dimensionality f = 2, 3, 5, 7 and 10. The labels denote average values of accuracy.

of the matching algorithm. The reasons of this choice are: (a) Response time varies depending on the implementation of the matching algorithm and of the access structures and (b) The time for distance computations (i.e., 1-2 seconds per matching on a PENTIUM PC 200Mhz) clearly dominates R-tree search time (i.e., less than 1 second per query!).

Figure 7 shows the average number of distance computations plotted against the tolerance t for f = 2, 3, 5, 7 and 10 dimensions. For $t < 0.5$, the 1-3 best matches are retrieved. Indexed search in this case is up to three orders of magnitude faster than sequential scanning. For $t \leq 2.5$, 52 shapes are retrieved on the average and our method is still much faster than sequential scanning for most f's.

The shape of the curves is justified as follows: Sequential scan, performs always the same number of distance computations. For the proposed method, the number of shapes which are retrieved from the R-tree increases exponentially with the tolerance t and each retrieved shape yields a shape distance calculation during clean-up. Moreover, the speed-up decreases with the dimensionality: As the dimensionality decreases, more points (shapes) are projected from a higher dimensionality space within t distance from the query (false drops) in the lower dimensionality space. The above false drops account for distance computations during clean-up and time responses are slowed-down.

4.3. Accuracy

The labels in Figure 7 denote average values of accuracy compared to sequential scanning (e.g., a value 0.95 denotes that 5% of the answers are missed). Presumably, a typical user retrieves about 50 shapes. In this case, $f = 5$ assures maximum speed-up with minimum information loss (i.e., 2%).

5. Conclusions

We propose a novel approach for handling similarity retrieval in a shape database. The heart of our approach is a shape matching algorithm which identifies similar shapes regardless of view-scale (resolution). With respect to retrieval, we propose an indexing method which achieves up to three orders of magnitude speed-up over sequential scanning with very high accuracy. Our method performs better than traditional methods to shape matching and retrieval [7].

Acknowledgements

The authors are grateful to Prof. Christos Faloutsos, Pantelis Elinas, Zusheng Rao, Marita Roussou and Panos Economopoulos for valuable contributions in this work.

References

[1] R. T. Chin and C. R. Dyer. Model-Based Recognition in Robot Vision. *ACM Computing Surveys*, 18(1):67–108, 1986.

[2] C. Faloutsos and K.-I. Lin. FastMap: A Fast Algorithm for Indexing, Data Mining and Visualization of Traditional and Multimedia Datasets. In *SIGMOD RECORD, Proceedings of the 95 ACM SIGMOD International Conference on Management of Data*, pages 163–174, San Jose, California, June 1995.

[3] L. Gurta and M. D. Srinath. Contour Sequence Moments for the Classifiaction of Closed Planar Shapes. *PR*, 20(3):267–271, 1987.

[4] A. Guttman. R-trees: A Dynamic Index Structure for Spatial Searching. In *Proceedings of ACM SIGMOD*, pages 47–57, June 1984.

[5] M.-K. Hu. Visual Pattern Recognition by Moment Invariants. *IRE Transactions in Information Theory*, IT-8:179–187, 1962.

[6] P. F. Korn, N. Sidiropulos, C. Faloutsos, E. Siegel, and Z. Protopapas. Fast and Effective Retrieval of Medical Tumor Shapes. *IEEE Transactions on Knowledge and Data Engineering*, 10(6):889–904, November/December 1998.

[7] B. M. Mehtre, M. S. Kankanhalli, and W. F. Lee. Shape Measures for Content Based Image Retrieval: A Comparison. *Information Processing and Management*, 33(3):319–337, 1997.

[8] E. Milios. Shape Matching using Curvature Processes. *Computer Vision, Graphics and Image Processing*, 47:203–226, 1989.

[9] T. W. Sederberg and E. Greenwood. A Physical Approach to 2-D Shape Bending. *Computer Grphics*, 26(2):25–34, 1992.

[10] N. Ueda and S. Suzuki. Learning Visual Models from Shape Contours Using Multiscale Convex/Concave Structure Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):337–352, April 1993.

[11] T. P. Wallace and P. A. Wintz. An Efficient Three-Dimensional Aircraft Recognition Algorithm Using Normalized Fourier Descriptors. *Computer Graphics and Image Processing*, 13:99–126, 1980.