Electronic Letters on Computer Vision and Image Analysis 7(2):62-75, 2008

Multiresolution Shape Codes with Applications to Image Retrieval

Arindam Biswas*, Partha Bhowmick*, and Bhargab B Bhattacharya+

* *Computer Science and Technology Department, Bengal Engineering and Science University, Shibpur, INDIA*
+ *Advanced Computing & Microelectronics Unit, Indian Statistical Institute, Kolkata, INDIA*

Received 17th October 2007; revised th January 2008; accepted 22nd June 2008

Abstract

A novel technique on shape coding of an object in a binary digital image, based on the tight isothetic polygonal covers of the object in a multiresolution background, is proposed. This technique would be useful in various analyses and applications related with digital images. To demonstrate the power and usefulness of such shape codes, we have also proposed an image retrieval scheme based on shape codes. The elegance of the scheme on shape codes lies in capturing the shape of the object(s) present in an image from its gross appearance to its finer details by a set of isothetic polygons, in a hierarchical manner. The inherent tolerance present in such a cast of isothetic polygonal shape enables the shape codes of two objects resemble closely as the grid resolution becomes finer and finer, provided the objects are of similar shapes. The method is very fast because it does not involve any complex computations, and requires only integer-based comparisons and additions. Experimental results demonstrate the strength and efficiency of the proposed scheme.

1 Introduction

Encoding the shape information of an image is very fundamental to visualization and retrieval of digital images. An efficient encoding, both in terms of construction and storage, is of utmost importance in implementing object-based systems, especially in multimedia research. Encoded shape information plays different roles depending upon the target application, vis-a-vis localization and retrieval of a specific shape. Several algorithms have been proposed in the literature that deal with the localization and retrieval problems.

The methods, namely, context-based arithmetic encoder (CAE) [10], runlength encoding [16], and polygon-based encoding [14] are some of the methods that deal with the problem of localization. In CAE-based technique, adapted in MPEG-4, the binary shape information is coded utilizing the macroblock-based structure in which binary alpha data are grouped within 16×16 binary alpha blocks. In the runlength encoding, the vertices of a polygonal approximation of the object's shape are encoded by adapting the representation to a dynamic range of the relative locations of the object's vertices. The polygon-based encoding deals with the lossy encoding of object boundaries that are given as eight-connected chain codes. However, the shape specification for localization, e.g., shape coding based on rate-distorion [10], is not useful for the retrieval as it has to undergo complex decoding and contour extraction process. Similarly, shape descriptors for retrieval purpose using curvature scale space descriptors [12, 13], geometric moments [8], etc., are not amenable to localization because the computation of these descriptors involves nonreversible transformation. The shape encoding has typically evolved around two basic concepts, one of which being bitmap based [17, 18, 19] where an object is encoded by

Correspondence to: `barindam@gmail.com`, `abiswas@cs.becs.ac.in`

Recommended for acceptance by João Manuel R. S. Tavares and Renato Natal Jorge
A preliminary version of this paper appeared in [4].

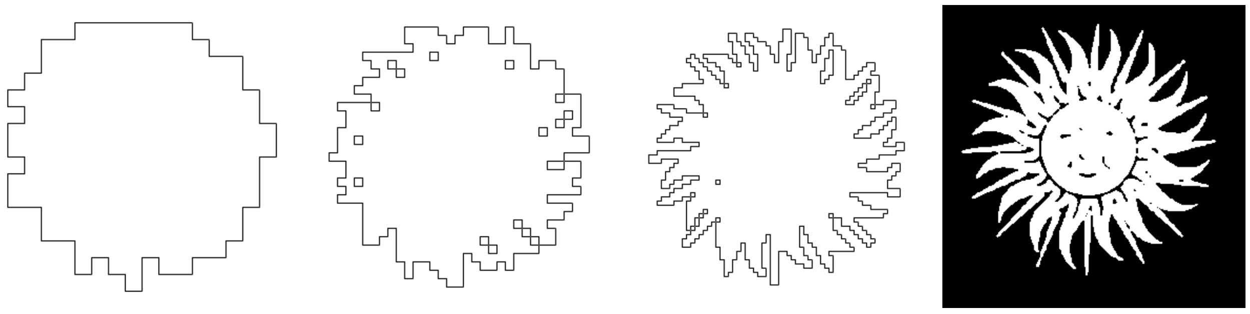


Figure 1: The isothetic polygons of an image are shown from left to right with increasingly finer resolution. The original image is shown in extreme right.

its support map, and the other being contour based (where the object boundary is represented efficiently), e.g., chain code [6, 9], polygonal approximation [11, 14, 16], high-order curve fittings viz. splines [10], combined polygon-spline approximation [7], approximation of polytopes [1], etc.

In this paper, a fast and efficient shape coding technique based on tight isothetic polygon(s) describing the binary object(s) on a multigrid background, is proposed. This method differs from other polygonal approximation methods [11, 14, 15, 16] in the fact that it does not require the contour to be extracted before constructing the isothetic (axis-parallel) polygons. It merges both the requirements of localization and retrieval with proper tuning. It stores the polygons for a given image imposed on coarse to finer background grids, with remarkably high storage efficiency. On finer grids, it serves as a good encoder for visualization, as illustrated in Fig. 1. For retrieval, at each finer level, the effective search database size gets diminished, eventually to report encouraging matching results. The rest of the paper is organized as follows. Sec. 2 explains the proposed work, which includes the construction of the isothetic polygon, the formulation of the shape code from the extracted isothetic polygons, and its usage in image retrieval. Sec. 3 presents the experimental results, and Sec. 4 makes the conclusion and a discussion about the future works related to the proposed work.

2 Proposed Work

A tight isothetic polygon covering a 2D object of any shape can be obtained using an earlier algorithm called TIPS [3]. It may be noted that, since the algorithm TIPS cannot find out multiple polygons (for multiple objects), it has been modified in the proposed work to obtain the complete set of isothetic polygons for multiple objects present in a binary image. Each polygon is uniquely described, starting from a suitable vertex (start vertex), and the corresponding higher level description of the polygon is coded properly to generate a shape code for the corresponding polygon. The shape code, thus obtained, would be always different for different polygons, and more importantly, the proximity between shape codes would depend on the degree of similarity between the two corresponding polygons. The shape codes of the polygons being unique, and the set of polygons being different for different images, the combined shape codes in a multiresolution environment turn out to be powerful features for the subsequent image retrieval process.

2.1 Tight Isothetic Polygons

Given a binary image $\mathcal{I}$ containing one or more objects (connected components[*]), and a set of uniformly spaced horizontal and vertical grid lines, $\mathcal{G} = (\mathcal{H}, \mathcal{V})$[†], the corresponding set of tight isothetic polygonal cover,

[*]We have considered 8-connectivity in this paper.

[†]In this paper, we have also used $\mathcal{G}$ to represent the set of grid points defined by $\mathcal{H}$ and $\mathcal{V}$; that is, the set of points (with integer coordinates) of intersection of the horizontal lines in $\mathcal{H}$ and the vertical lines in $\mathcal{V}$ is also denoted by $\mathcal{G}$, the meaning of the corresponding representation being evident from the context.

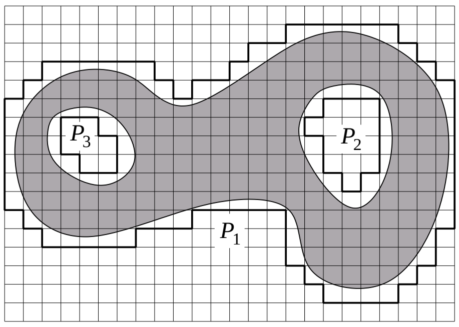


Figure 2: Isothetic polygonal cover consisting of an outer polygon, P_1, and two hole polygons, P_2 and P_3.

$\mathcal{P}(\mathcal{I},\mathcal{G}) = \{P_i\}_{i=1}^{n}$, consists of n isothetic polygons, such that the following conditions are satisfied:

(c1) $V_i \cap V_j = \emptyset, \forall i, \forall j, i \neq j$, where V_i and V_j represent the ordered set of vertices of P_i and P_j respectively.

(c2) P_i is either an *outer polygon* containing one connected component $\mathcal{C}$ of $\mathcal{I}$ or a *hole polygon* contained in a hole of $\mathcal{C}$. If P_i is an outer polygon, then each point $p \in \mathcal{C}$ lies inside P_i; if P_i is a hole polygon, then each point $p \in \mathcal{C}$ lies outside P_i.

(c3) Each vertex of $P_i, i = 1, 2, \ldots, n$, is an element of $\mathcal{G}$.

(c4) The sum of the areas of the outer polygons minus the sum of the areas of the hole polygons is minimized.

It may be noted that, property (c1) ensures that no two polygons in $\mathcal{P}(\mathcal{I},\mathcal{G})$ will intersect each other, although one polygon may contain another polygon, such that the latter (hole polygon) will lie completely inside the former (outer polygon), and the latter may again contain some other polygon (outer polygon), and so on. For example, in Fig. 2, the outer polygon P_1 contains the hole polygons P_2 and P_3, such that $P_1 - (P_2 + P_3)$ is the minimum isothetic region that contains the given object. Further, the (ordered) vertices of a polygon in $\mathcal{P}(\mathcal{I},\mathcal{G})$ may be such that the polygon may have its one horizontal edge intersecting with its one vertical edge, which implies that the concerned polygon may possess self-intersections. It may be also noted that, if an object in $\mathcal{I}$ is so small compared to the grid unit that the object does not intersect any horizontal or vertical grid line, which occurs when a tiny fragment occuring in an image is not perceivable in a coarse grid, then there would be no isothetic polygon in $\mathcal{P}(\mathcal{I},\mathcal{G})$ that would contain the object.

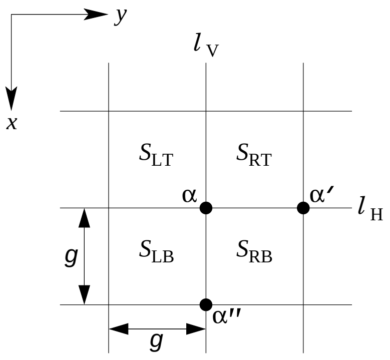
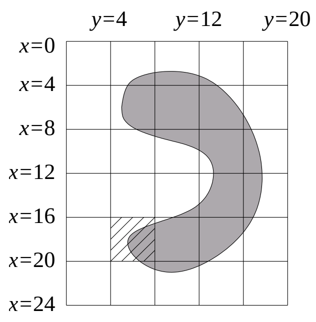


Figure 3: Four unit grid squares with common vertex α.



	0	1	2	3	4	5
0	0	0	0	0	0	-
1	0	0	1	1	0	0
2	0	1	1	1	0	-
3	0	0	1	1	1	0
4	0	1	1	1	1	-
5	0	0	1	1	1	0
6	0	0	0	1	1	-
7	0	0	0	1	1	0
8	0	0	1	1	1	-
9	0	0	1	1	1	0
10	0	1	1	1	0	-
11	0	0	1	1	0	0
12	0	0	0	0	0	-

Unit edge matrix, M_e.

	0	1	2	3	4
0	0	1	1	1	0
1	0	1	1	1	0
2	0	1	1	1	1
3	0	0	1	1	1
4	0	1	1	1	1
5	0	1	1	1	0

Unit square matrix, M_s.

Figure 4: Illustration of the formation of the unit edge matrix and the unit square matrix using the function φ (Eqn. 1).

2.2 Finding Tight Isothetic Polygons

Let $\mathcal{R}$ be the smallest two-dimensional image plane that contains the entire set of objects, denoted by $\mathcal{I}$. Henceforth in this paper, we use $\mathcal{I}$ to denote the binary image as well as the set of objects (connected components) present in the image, depending on the context. Let g be the underlying *grid size*, which is equal to the length (i.e., height or width) of each unit grid square in $\mathcal{G}$, defined over $\mathcal{R}$. It may be noted that the height h and width w of the plane $\mathcal{R}$ are chosen appropriately to suit the requirement that g divides both $h-1$ and $w-1$. Let $\alpha(i,j)$ be the point of intersection of the horizontal grid line $l_H : x = i$ and the vertical grid line $l_V : y = j$, where, $l_H \in \mathcal{H}$ and $l_V \in \mathcal{V}$. It may be observed that, since $\alpha(i,j)$ is a point on the grid with grid size g, g always divides i and j, i.e., $i \pmod g = 0$ and $j \pmod g = 0$. Let $\mathcal{S}_{\mathrm{LT}}(i,j)$, $\mathcal{S}_{\mathrm{RT}}(i,j)$, $\mathcal{S}_{\mathrm{LB}}(i,j)$, and $\mathcal{S}_{\mathrm{RB}}(i,j)$ represent the four unit grid sqaures surrounding the common grid node α, and lying at the top-left, right-top, left-bottom, and right-bottom w.r.t. α respectively, as shown in Fig. 3.

We define a function φ to construct a binary matrix M_e (*unit edge matrix*) that stores the information regarding the intersection of each of the unit grid edges (of length g) with the object $\mathcal{I}$. It may be noted that the total number of unit horizontal edges is $\frac{w-1}{g}(\frac{h-1}{g}+1)$, and total number of unit vertical edges is $\frac{h-1}{g}(\frac{w-1}{g}+1)$, which togetherly decide the size of the unit edge matrix, M_e. For example, in Fig. 4, $h = 25, w = 21$, and $g = 4$. The number of horizontal edges in each horizontal grid line is $(w-1)/g = 20/4 = 5$, and hence the total number of horizontal edges is $5 \times ((h-1)/g + 1) = 5 \times (24/4 + 1) = 35$. Similarly, the total number of vertical edges is $6 \times 6 = 36$. Let $\alpha'(i, j+g)$ be the grid point lying immediate right of $\alpha(i,j)$, and $e(\alpha, \alpha')$ be the unit horizontal grid edge connecting α and α' (Fig. 3). Similarly, let $\alpha''(i+g, j)$ be the grid point lying immediate below $\alpha(i,j)$, and $e(\alpha,\alpha'')$ be the unit vertical grid edge connecting α and α''. Then the function φ, defined as follows, indicates the entry place, $\langle i_{e(\alpha,\beta)}, j_{e(\alpha,\beta)}\rangle$, in M_e where the binary information about the intersection of the unit edge $e(\alpha,\beta)$ with the object $\mathcal{I}$ is stored:

$$\varphi : e(\alpha,\beta) \mapsto \begin{cases} \langle 2i/g, j/g\rangle, & \text{if } \beta = \alpha'; \\ \langle 2i/g+1, j/g\rangle, & \text{if } \beta = \alpha''. \end{cases} \tag{1}$$

The above function φ is defined in such a way that ensures adjacent entries in M_e corresponding to the four edges defining a grid square. Further, this helps the evaluation of the corresponding entry of the grid square in M_e, as shown in Eqn. 3. For example, in Fig. 4, the entries in M_e corresponding to the horizontal edges of the line-shaded grid square have been shown in blue, and those corresponding to the vertical edges have been shown in green. For the top horizontal edge of this square, we have $\alpha = (16,4)$ and $\alpha' = (16,8)$. The entry place in M_e is, therefore, given by $\langle 2\cdot 16/4, 4/4\rangle = \langle 8,1\rangle$. For the bottom horizontal edge, $\alpha = (20,4)$ and $\alpha' = (20,8)$, and so the entry place in M_e is $\langle 2\cdot 20/4, 4/4\rangle = \langle 10,1\rangle$. Similarly, the left vertical edge has the entry place $\langle 2{\cdot}16/4{+}1, 4/4\rangle = \langle 9,1\rangle$, and the right vertical edge has the entry place $\langle 2{\cdot}16/4{+}1, 8/4\rangle = \langle 9,2\rangle$. The number of rows M_e is given by $2 \times (h-1)/g + 1$ and the number of columns is $(w-1)/g+1$. In the above example, the size of M_e is $(2 \cdot 24/4 + 1) \times (20/4+1) = 13 \times 6$.

Depending on the intersection of the edge $e(\alpha,\beta)$ with the object $\mathcal{I}$, the corresponding entry $M_e[i_{e(\alpha,\beta)}][j_{e(\alpha,\beta)}]$ is decided as follows:

$$M_e[i_{e(\alpha,\beta)}][j_{e(\alpha,\beta)}] = \begin{cases} 1, & \text{if edge } e(\alpha,\beta) \text{ intersects } \mathcal{I}; \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Now from the unit edge matrix, M_e, we construct another binary matrix, called the *unit square matrix*, M_s, having $(h-1)/g$ rows and $(w-1)/g$ columns, as follows.

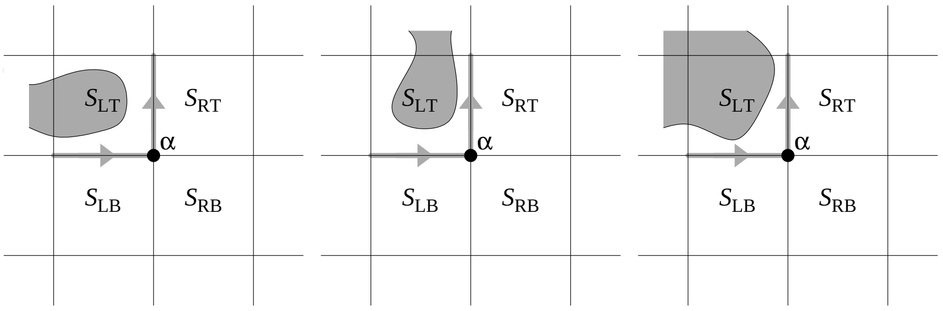
$$M_s[i_s][j_s] = \begin{cases} 1, & \text{if } ((\; M_e[2i_s][j_s] = 1 \text{ or} \\ & \qquad M_e[2i_s+1][j_s] = 1 \text{ or} \\ & \qquad M_e[2i_s+1][j_s+1] = 1 \text{ or} \\ & \qquad M_e[2i_s+2][j_s]) = 1) \\ 0, & \qquad \text{otherwise} \end{cases} \tag{3}$$

It may be observed that, if an entry in the unit square matrix, M_s, is unity, then the corresponding unit square grid in $\mathcal{R}$ contains some part of the object $\mathcal{I}$. More precisely, if $M_s[i_s][j_s] = 1$, then the unit grid square, formed by the four grid lines (two horizontal and two vertical), namely $x = gi_s$, $x = g(i_s+1)$, $y = gj_s$, and $y = g(j_s+1)$, contains some part of $\mathcal{I}$. Furthermore, as evident from Eqn. 3, $M_s[i_s][j_s] = 1$ if and only if at least one among $M_e[2i_s][j_s]$, $M_e[2i_s+1][j_s]$, $M_e[2i_s+1][j_s+1]$, and $M_e[2i_s+2][j_s]$ is unity, since the object $\mathcal{I}$ must penetrate at least one of the four edges of the unit grid square in order that it lies inside the concerned square. For example, in Fig. 4, the entry place $\langle 4,1\rangle$ in M_s, shown in red, corresponds to the hatched grid square. Here, $i_s = 4$ and $j_s = 1$. The entry $M_e[2i_s][j_s] = M_e[8][1]$ provides the information on intersection of the top horizontal edge defining the hatched grid square. Similarly, the entries $M_e[2i_s+1][j_s] = M_e[9][1]$, $M_e[2i_s+1][j_s+1] = M_e[9][2]$, and $M_e[2i_s+2][j_s] = M_e[10][1]$ provide similar information corresponding to the left vertical, right vertical, and bottom horizontal edges respectively. Since out of these four entries in M_e, the entries ($M_e[9][2]$ and $M_e[10][1]$) corresponding to the right vertical edge and the bottom horizontal edge are unity, we get $M_s[4][1] = 1$.

2.2.1 Finding the Vertices of $\mathcal{P}(\mathcal{I},\mathcal{G})$

In the proposed algorithm, the candidature of α as a vertex of $\mathcal{P}(\mathcal{I},\mathcal{G})$ is checked by looking at the combinatorial arrangements (w.r.t. object containments) of the four unit grid squares having common vertex α (Fig. 3). It may be noted that, each of these four unit squares has a binary entry at the corresponding locations in the *unit square matrix*, M_s, which, in turn, is derived from Eqns. 1–3, as discussed above. These four entries together form a 2×2 submatrix in M_s. Now, there exist $2^4 = 16$ different arrangements of these 4 unit grid squares, since each square have 2 possibilities (‘0’/‘1’). These 16 arrangements have been further reduced to 5 cases in this algorithm, where, a particular case $\mathbf{C}_q$, $q = 0,1,\ldots,4$, includes all the arrangements in which exactly q out of these 4 squares have object containments (i.e., contain parts of the object $\mathcal{I}$), and the remaining (i.e., $4-q$) ones have not. That is, the case in which the sum of the 4 bits in the corresponding entries in M_s is equal to q is represented by $\mathbf{C}_q$. Further, out of these 5 cases, only two cases, namely $\mathbf{C}_1$ and $\mathbf{C}_3$, always represent vertices of $\mathcal{P}(\mathcal{I},\mathcal{G})$, and one case, namely $\mathbf{C}_2$, may conditionally give rise to a vertex of $\mathcal{P}(\mathcal{I},\mathcal{G})$, as discussed below.

Case $\mathbf{C}_1$:
In this case, exactly one of the four unit grid squares contains some part of the object $\mathcal{I}$. W.l.g., let $\mathcal{S}_{\mathrm{LT}}(i,j)$ be the unit grid square that contains some part of $\mathcal{I}$, as shown in Fig. 5. Hence the isothetic cover, $\mathcal{P}(\mathcal{I},\mathcal{G})$, will have its one horizontal edge ending at α and the next vertical edge starting from α, if we consider traversal



(a) $\mathcal{I}$ intersects only the left edge of $\mathcal{S}_{\rm LT}$. (b) $\mathcal{I}$ intersects only the top edge of $\mathcal{S}_{\rm LT}$. (c) $\mathcal{I}$ intersects both the left edge and the top edge of $\mathcal{S}_{\rm LT}$.

Figure 5: Three possible instances for one of the 4 arrangements of case $\mathbf{C}_1$, where α is a 90^0 vertex. The edges (right edge and bottom edge of $\mathcal{S}_{\rm LT}$), which would belong to $\mathcal{P}(\mathcal{I},\mathcal{G})$, have been highlighted and directed to show their directions of traversal with $\mathcal{I}$ always lying at the left.

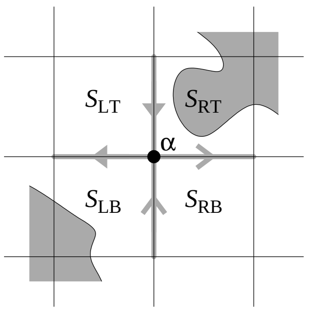


Figure 6: An instance of one of the 2 arrangements of case $\mathbf{C}_2$, where α is a 270^0 vertex.

along the edges of $\mathcal{P}(\mathcal{I},\mathcal{G})$ in a way such that the region/object $\mathcal{I}$ always lies to the left of each edge while being traversed (shown by the respective arrows in Fig. 5). This argument holds for each of the 4 arrangements where exactly one of the corresponding four binary entries in M_s is unity and the remaining three is zero. This observation leads to the fact that α is a 90^0 vertex (i.e. a vertex with 90^0 internal angle) of $\mathcal{P}(\mathcal{I},\mathcal{G})$, if and only if $q = 1$.

Case $\mathbf{C}_2$:
Here, exactly two of the four unit grid squares contain parts of $\mathcal{I}$. If the two unit grid squares, having object containments, do not have any unit grid edge in common, only then α will be vertex of $\mathcal{P}(\mathcal{I},\mathcal{G})$. It is easy to observe that, in case $\mathbf{C}_2$, only two different arrangements are possible for which α is a vertex; these are: (i) $\mathcal{S}_{\rm RT}(i,j)$ and $\mathcal{S}_{\rm LB}(i,j)$ contain object parts, and (ii) $\mathcal{S}_{\rm LT}(i,j)$ and $\mathcal{S}_{\rm RB}(i,j)$ contain object parts. An instance of arrangement (i) is shown in Fig. 6. It can be shown that, in each of these two arrangements, α becomes a 270^0 vertex, since $\mathcal{P}(\mathcal{I},\mathcal{G})$ is considered to be non-self-intersecting, as stated in condition **(c1)** in Sec. 2.1. It should be carefully observed in Fig. 6 that two different styles of arrow heads indicate the two possibilities of formation of edges of $\mathcal{P}(\mathcal{I},\mathcal{G})$, since α may be visited along either of the two possible paths during construction of $\mathcal{P}(\mathcal{I},\mathcal{G})$, and only the traced one is included in the final solution of $\mathcal{P}(\mathcal{I},\mathcal{G})$.

For all other (four) arrangements with $q = 2$, α is just an ordinary point lying on some edge of $\mathcal{P}(\mathcal{I},\mathcal{G})$.

Case $\mathbf{C}_3$:
If $q = 3$, then out of the four unit grid squares, only one square is free, which will have 4 different arrangements. In each of these arrangements, one of which is shown in Fig. 7, α would be a 270^0 vertex (i.e. a vertex with 270^0 internal angle).

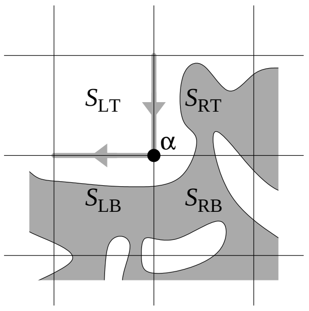


Figure 7: A typical instance of one of the 4 arrangements of case $\mathbf{C}_3$, where α is a 270^0 vertex.

For the two other cases, namely case $\mathbf{C}_0$ and case $\mathbf{C}_4$, it can be proved that α can never be a vertex of $\mathcal{P}(\mathcal{I},\mathcal{G})$. Case $\mathbf{C}_0$ implies that α is just an ordinary grid point that lies in $\mathcal{R} \setminus \mathcal{I}$, and can be shown to by lying outside $\mathcal{P}(\mathcal{I},\mathcal{G})$. Case $\mathbf{C}_4$ indicates that α is a grid point that either lies in $\mathcal{I}$, or lies in a hole of $\mathcal{I}$ and is surrounded by parts of $\mathcal{I}$ in all four unit grid squares with common vertex α, whence α can be shown to be a grid point lying inside $\mathcal{P}(\mathcal{I},\mathcal{G})$.

2.2.2 Storing the vertices of $\mathcal{P}(\mathcal{I},\mathcal{G})$

It is easy to see that there are two types of vertices of the isothetic polygon, $\mathcal{P}(\mathcal{I},\mathcal{G})$: 90^0 vertex (obtained from case $\mathbf{C}_1$), and 270^0 vertex (obtained from case $\mathbf{C}_2$ and case $\mathbf{C}_3$), whose nature are discussed in Sec. 2.2.1. During the process of extraction of these vertices, each of them is dynamically inserted simultaneously in two temporary link lists, L_x and L_y, such that L_x always remains lexicographically sorted in an increasing order w.r.t. x (primary key) and y (secondary key), and L_y always lexicographically sorted in an increasing order w.r.t. y (primary key) and x (secondary key). In addition to the grid coordinates of these vertices, one bit (*type-bit*) is stored for each vertex v in L_x and in L_y to denote its type ('0' denotes a 90^0 vertex and '1' denotes a 270^0 vertex). Furthermore, when the first 90^0 vertex ($v_1^{(0)}$) is detected, the way in which its two edges should be traversed, such that the object $\mathcal{I}$ lies left during traversal, is decided and stored accordingly. After extraction of all the vertices, the link lists L_x and L_y are processed, starting from any one of the 90^0 vertices, in order to construct the isothetic polygonal cover, $\mathcal{P}(\mathcal{I},\mathcal{G})$, as discussed in Sec. 2.2.3.

2.2.3 Construction of $\mathcal{P}(\mathcal{I},\mathcal{G})$

The construction of the isothetic polygonal cover, $\mathcal{P}(\mathcal{I},\mathcal{G})$, starts from $v_1^{(0)}$, which is the start vertex, as discussed in Sec. 2.2.2. It may be noted that, considering any 270^0 vertex as a start vertex is risky, since that vertex may be derived as a result of case $\mathbf{C}_2$, which has a dubious nature, as discussed in Sec. 2.2.1 and illustrated in Fig. 6. Further, if, for example, the isothetic cover is merely a rectangle, then there exist 90^0 vertices and there does not exist any 270^0 vertex. Hence a vertex with internal angle 90^0 should always be considered as a start vertex.

Now, if the outgoing edge from $v_1^{(0)}$ is vertical and directed upwards top (as shown in Fig. 5, then the preceding vertex in the list L_x is the next vertex, v_{next}, of $\mathcal{P}(\mathcal{I},\mathcal{G})$, since the points in L_x are ordered w.r.t. x. Similarly, if the outgoing edge from $v_1^{(0)}$ is vertical and directed downwards, then the succeeding vertex in the list L_x becomes the next vertex, v_{next}, of $\mathcal{P}(\mathcal{I},\mathcal{G})$. For the other two possible arrangements, the decisions are similarly taken, and v_{next} is obtained using L_y. Once v_{next} is found, then depending on the type-bit ('0' or '1') of v_{next}, the outgoing edge from v_{next} is decisively selected using the rule that $\mathcal{I}$ always lies left during traversal, and the process continues until the start vertex $v_1^{(0)}$ is reached (after traversing the incident edge of $v_1^{(0)}$ as the last edge of $\mathcal{P}(\mathcal{I},\mathcal{G})$). In this manner, the isothetic polygonal covers for all the connected components of $\mathcal{I}$ can be constructed.

2.3 Shape Code

Let V_{ri} be the ordered set of vertices for the ith polygon P_{ri} in $\mathcal{P}(\mathcal{I}_r, \mathcal{G}) = \{P_{ri}\}_{i=1}^{n_r}$ corresponding to the rth digital image $\mathcal{I}_r$ in a binary image database $\mathcal{D} = \{\mathcal{I}_r\}_{r=1}^{N}$ containing N images. Now, in $V_{ri} = \langle v_{ri}^{(1)}, v_{ri}^{(2)}, \ldots, v_{ri}^{(m_{ri})} \rangle$, consisting of m_{ri} vertices, each vertex $v_{ri}^{(k)}$ will be either a 90^0 vertex or a 270^0 vertex, depending on the object containments in the four unit grid quadrants incident at $v_{ri}^{(k)}$. Further, in the set V_{ri}, there will be always at least one 90^0 vertex that will have object containment in its bottom-right unit-grid-quadrant and the other three quadrants free. One such vertex is selected as the “start vertex” to generate the shape code of P_{ri}. W.l.o.g., let $v_{ri}^{(1)}$ be the start vertex in V_{ri}.

It may observed that, any two consecutive vertices in V_{ri} would form either a horizontal edge or a vertical edge of P_{ri}. Hence, if $(x[v_{ri}^{(j)}], y[v_{ri}^{(j)}])$ be the coordinates of the jth vertex in V_{ri}, then the distance (edge length) between two consecutive vertices $v_{ri}^{(k)}$ and $v_{ri}^{(k+1)}$ in V_{ri} is given by either $\left|x[v_{ri}^{(k)}] - x[v_{ri}^{(k+1)}]\right|$ or $\left|y[v_{ri}^{(k)}] - y[v_{ri}^{(k+1)}]\right|$, whichever is non-zero.

Now, in the counter-clockwise enumeration of the vertices of P_{ri}, starting from the start vertex $v_{ri}^{(1)}$, the length of each edge and the type of the destination vertex of the edge are stored in eight-bit (one byte) representation, where the two most significant bits are reserved for the two types of vertices (90^0 and 270^0) and a false vertex (180^0) which acts as a continuity flag for a long edge whose length has to be represented in more than 6 bits. In general, if $e(v_{ri}^{(k)}, v_{ri}^{(k+1)})$ denotes the edge from the vertex $v_{ri}^{(k)}$ to the next vertex $v_{ri}^{(k+1)}$, the latter being the destination vertex, then the two most significant bits are “`01`” if type$(v_{ri}^{(k+1)}) = 90^0$, or “`11`” if type$(v_{ri}^{(k+1)}) = 270^0$. Further, if $|e(v_{ri}^{(k)}, v_{ri}^{(k+1)})|$ denotes the edge length between $v_{ri}^{(k)}$ and $v_{ri}^{(k+1)}$, and $|(v_{ri}^{(k)}, v_{ri}^{(k+1)})| > 2^6 - 1 = 63$ grid units, then the problem of bit overflow is resolved by making the two most significant bits to be “`10`” (indicating 180^0) and the overflowing bits are put to the next byte until the entire length of the edge is stored in the corresponding bit representation.

As an example, if $|e(v_{ri}^{(k)}, v_{ri}^{(k+1)})| = 184$ and type$(v_{ri}^{(k+1)}) = 270^0$, then the code for $e(v_{ri}^{(k)}, v_{ri}^{(k+1)})$ needs 16 bits; and the 16-bit code for this edge would be “`11000010 10111000`”.

The shape code for the polygon P_{ri}, having the ordered set of vertices $V_{ri} = \langle v_{ri}^{(1)}, v_{ri}^{(2)}, \ldots, v_{ri}^{(m_{ri})} \rangle$, therefore, can be obtained as follows:

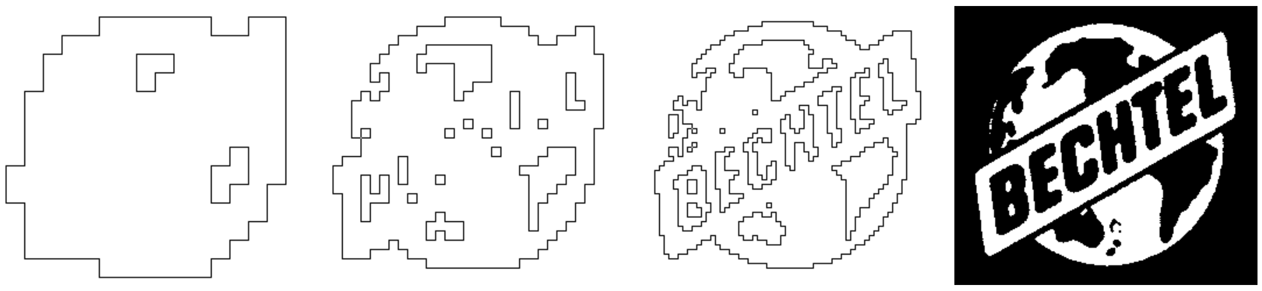
$$\begin{aligned} SC(P_{ri}, g) = x[v_{ri}^{(1)}]y[v_{ri}^{(1)}] \quad & (b_8 b_7)_{ri}^{(2)} |e(v_{ri}^{(1)}, v_{ri}^{(2)})| \\ & (b_8 b_7)_{ri}^{(3)} |e(v_{ri}^{(2)}, v_{ri}^{(3)})| \cdots \\ & (b_8 b_7)_{ri}^{(m_{ri})} |e(v_{ri}^{(m_{ri}-1)}, v_{ri}^{(m_{ri})})|, \end{aligned} \tag{4}$$

where, $(x[v_{ri}^{(1)}], y[v_{ri}^{(1)}])$ gives the start (first) vertex, $(b_8 b_7)_{ri}^{(2)}$ denotes the two (most significant) bits representing the vertex type of the second vertex $v_{ri}^{(2)}$, $|e(v_{ri}^{(1)}, v_{ri}^{(2)})|$ the length of the edge from the first vertex to the second vertex; and so on.

2.4 Multigrid Shape Code

The shape code of a tight isothetic polygon for an object (connected component) in a given uniform (square) grid $\mathcal{G} = (\mathcal{H}, \mathcal{V})$ depends not only on the shape of the object but also on the spacing of the grid lines. Higher the separation between the grid lines, lower will be the complexity (number of vertices) of the tight isothetic polygon (and its shape code, there of) containing the object, and also lower will be the visual perception about the underlying object from its isothetic polygon. On the contrary, as the grid lines become denser, the corresponding isothetic polygon becomes more meaningful and expressive, thereby facilitating the process of recognizing the underlying object.

In order to achieve a fast preliminary result on resemblance of object shapes, therefore, sparsely separated grid lines with large unit grid squares are used to generate the shape codes of objects. Shape codes for denser



(a) $g = 16$. (b) $g = 8$. (c) $g = 4$. (d) original.

MuSC:$\underline{5}$ $(\underline{0})$ (11000001)(01000010)$\cdots$ (01000110)(0^8) ▷ end of 1st polygon
$(\underline{7})$ $(\underline{2})$ (01000010)(01000001)$\cdots$ (01000010)(0^8) ▷ end of 2nd polygon
$(\underline{12})(\underline{7})$ (11000001)(01000001)$\cdots$ (01000001)(0^8)(0^8) ▷ end of $g_1 = 16$
$(\underline{13})(\underline{1})$ (11000001)(01000011)$\cdots$ (01000110)(0^8) ▷ end of 1st polygon
$(\underline{11})(\underline{4})$ (11000001)(01000001)$\cdots$ (01000111)(0^8) ▷ end of 2nd polygon
$\vdots$
$(\underline{12})(\underline{22})$(11000001)(01000001)$\cdots$ (01000001)(0^8)(0^8) ▷ end of $g_2 = 8$
$\vdots$
$(\underline{25})(\underline{44})$(11000001)(01000010)$\cdots$ (01000010)(0^8)(0^8) ▷ end of $g_3 = 4$

Figure 8: A sample logo image and its tight isothetic polygonal covers representing the corresponding MuSC. Each byte (8 bits) has been shown in parentheses for clarity, and the coordinates of start vertex of each polygon have been underlined. The $+x$-axis is considered from left to right, and the $+y$-axis from top to bottom.

grid lines with smaller unit grid squares, on the other hand, are required for finer and accurate checking between objects with similar shapes as certified from grid boxes with higher sizes. Hence shape codes of the tight isothetic polygons are generated for an image for different grid sizes. Isothetic polygons, as interpreted and realized from their shape codes, have been shown for a sample logo image in Fig. 8 for illustration. It may be observed in Fig. 8 how the underlying objects become more and more revealing as the grid size (separation of grid lines) goes on decreasing from 16 in Fig. 8(a) to 4 in Fig. 8(c).

Now, The **Mu**ltigrid **S**hape **C**ode (MuSC) for the image $\mathcal{I}_r$ is given by the concatenated ordered sequence of the shape codes (SC) of the objects (connected components) in the image, where a byte with zero value (i.e., each of its eight bits is 0) marks the end of the shape code of a single polygon, and two successive zero-valued bytes indicate the end of the shape codes of all the polygons for a particular grid configuration. Hence, MuSC for the image $\mathcal{I}_r$ is given by:

$$MuSC(\mathcal{I}_r) = \quad SC(\mathcal{I}_r, g_1)\, \mathtt{0}^{16}\, SC(\mathcal{I}_r, g_2)\, \mathtt{0}^{16} \cdots \mathtt{0}^{16}\, SC(\mathcal{I}_r, g_\alpha), \tag{5}$$

where, α numbers of different grid separations, ranging from g_1 to g_α, have been used to generate the above MuSC. The shape code $SC(\mathcal{I}_r, g)$ of the image $\mathcal{I}_r$ for a grid size g, $g_1 \leq g \leq g_\alpha$, which has been used in the above equation to obtain $MuSC(\mathcal{I}_r)$, is as follows:

$$SC(\mathcal{I}_r, g) = \quad SC(P_{r1}, g)\, \mathtt{0}^8\, SC(P_{r2}, g)\, \mathtt{0}^8 \cdots \mathtt{0}^8\, SC(P_{rn_r}, g). \tag{6}$$

It may be noted that, since images in a database $\mathcal{D}$ may have nonuniform sizes, all images in $\mathcal{D}$ are normalized to the size of $\mu \times \mu$ before finding their MuSCs. In our experiments, we have considered $\mu = 256$ pixels.

2.5 Image Retrieval using MuSC

Let $\mathcal{Q}$ be the query image. At first the multigrid shape code $MuSC(\mathcal{Q})$ of $\mathcal{Q}$ (for grid sizes g from g_1 to g_α) is obtained as shown in Eqn. 5. It may be noted that, for any image $\mathcal{I}$, $MuSC(\mathcal{I})$ represents the highest level description of all the tight isothetic polygons corresponding to the object(s) present in the image $\mathcal{I}$ for increasing grid resolution from g_1 to g_α. Hence, we resort to the next (lower) level description of each isothetic polygon in the process of retrieval as follows.

For each database image $\mathcal{I}_r$, an ordered list, namely $L(\mathcal{I}_r, g_1)$, containing (pointers to) the vertical edges of all its isothetic polygons corresponding to the grid size g_1, is prepared using $SC(P_r, g_1)$ (whose form being in accordance with Eqn. 4). It may be noted that $SC(P_r, g_1)$, in turn, is extracted from its multigrid shape code, $MuSC(\mathcal{I}_r)$ (given in Eqn. 5). The list $L(\mathcal{I}_r, g_1)$ contains $H_1 = \mu/g_1$ pointers corresponding to $h = 1$ to $h = H_1$, since μ is the height of the normalized image. The salient points about the structure of $L(\mathcal{I}_r, g_1)$ are as follows:

– the pointer $L(\mathcal{I}_r, g_1)[h]$ points to a linked list containing those edges of $SC(P_r, g_1)$, each of whose upper vertex has y-coordinate = h.

– each node in the linked list pointed from $L(\mathcal{I}_r, g_1)[h]$ represents a vertical edge $e(u, v)$ (with $h = y[u] < y[v]$ and $x[u] = x[v]$) of $SC(P_r, g_1)$, and contains (i) $x[u]$, (ii) $y[u]$, and (iii) $y[v]$, which are required in the subsequent steps of the retreival process.

– in the linked list that $L(\mathcal{I}_r, g_1)[h]$ points to, the edges (nodes) are sorted in ascending order of their x-coordinates (of upper vertices).

A similar list, $L(\mathcal{Q}, g_1)$, is also prepared for the query image $\mathcal{Q}$ to find out the similarity between the isothetic polygon(s) of $\mathcal{Q}$ and those of $\mathcal{I}_r$ for grid size g_1. In order to do this, we construct a binary matrix $M(\mathcal{I}_r, g_1)$, having size $H_1 \times H_1$, considering $g = g_1$ and $H = H_1$ in Eqn. 7. A similar matrix, namely $M(\mathcal{Q}, g_1)$, is also being constructed for $\mathcal{Q}$ using Eqn. 7, and these two matrices, $M(\mathcal{I}_r, g_1)$ and $M(\mathcal{Q}, g_1)$, are used to find the Hamming distance $L\left(M(\mathcal{I}_r, g_1), M(\mathcal{Q}, g_1)\right)$ between the isothetic polygons of database image $\mathcal{I}_r$ and those of the query image $\mathcal{Q}$ corresponding to grid size g_1, considering $g = g_1$ and $H = H_1$ in Eqn. 8.

$$M(\mathcal{I}_r, g)[w][h] = \begin{cases} 1, & \text{if } |\{e(u,v) \in L(\mathcal{I}_r, g) : y[u] \le h < y[v] \text{ AND } x[u] \le w\}| \text{ is odd ;} \\ 0, & \text{otherwise;} \end{cases} \quad \text{for } 1 \le w, h \le H. \tag{7}$$

$$L\left(M(\mathcal{I}_r, g), M(\mathcal{Q}, g)\right) = \sum_{w=1}^{H} \sum_{h=1}^{H} |M(\mathcal{I}_r, g)[w][h] - M(\mathcal{Q}, g)[w][h]| \tag{8}$$

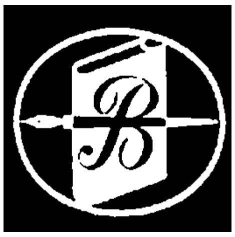
It is evident from Eqn. 7 and Eqn. 8, if the Hamming distance $L\left(M(\mathcal{I}_r, g_1), M(\mathcal{Q}, g_1)\right)$ is small, then the image $\mathcal{I}_r$ is a probable candidate for a successful retrieval corresponding to the query image $\mathcal{Q}$. Hence the images with lower values of Hamming distance are considered one by one for next grid size g_2.

It may be noted here that the construction of isothetic polygons and the derivation of the shape codes from them and their usage in image retrieval do not involve any floating point computation (only comparisons and additions), thereby making the process very fast.

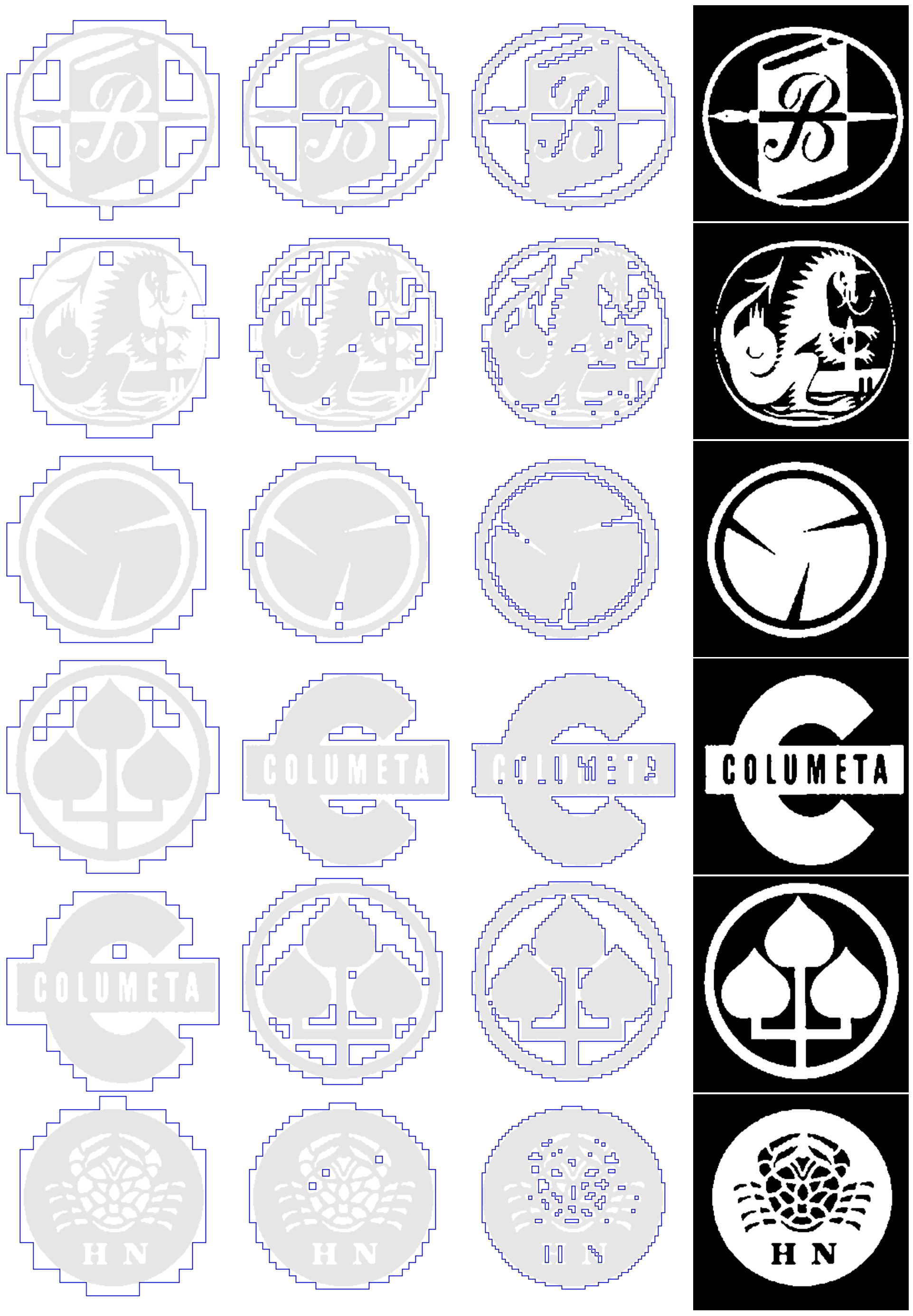
3 Results and Discussions

We have used two sets of binary images for our experiments: (i) database D1 of 1034 logo images, received on request, from Prof. Anil K. Jain and Aditya Vailya of Michigan State Univ., USA, and (ii) database D2 of 110 logo images collected from the Internet.

The proposed method is implemented in C on a Sun_Ultra 5_10, Sparc, $233\ MHz$, the OS being the SunOS Release 5.7 Generic. The results of the experiments done on the above two sets of image database are shown in Table 1. All the images are normalized to size 256×256. The table shows the amount of storage (in bytes

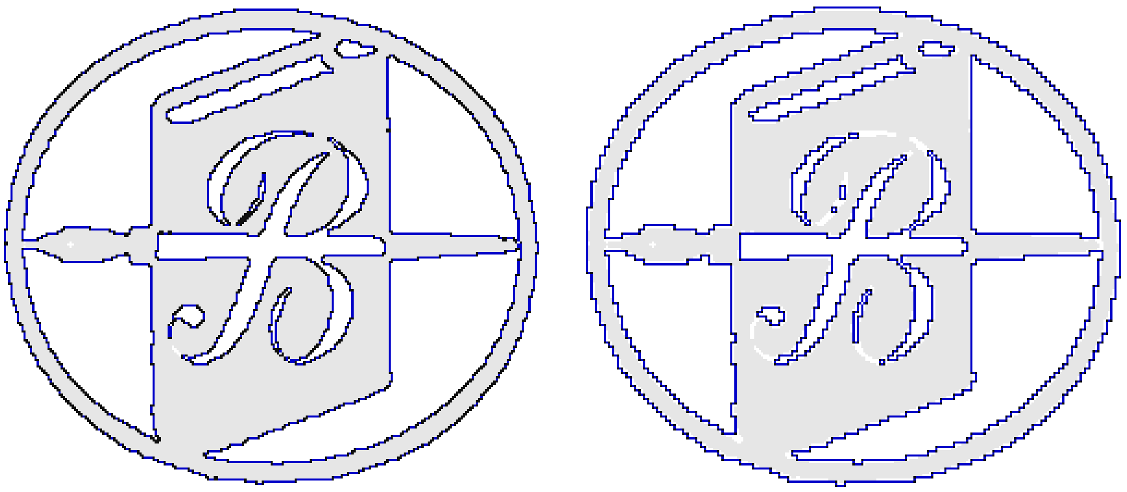


Query image



(a) $g = 16$ (b) $g = 8$ (c) $g = 4$ (d) original

Figure 9: The results of querying the database with the query image shown. The images in each column are ordered from top to bottom in increasing hamming distance while different columns correspond to different grid sizes.



(a) $g = 1$. (b) $g = 2$.

Figure 10: Shape codes extracted at low grid sizes (i.e., 1 and 2) aid in object visualization.

as well as in percentage) required for the shape codes for different grid sizes. It also shows the average CPU time (per image) required to construct the shape code. The storage requirement is remarkably lower for coarse grids compared to finer grids, as in a finer grid the shape is captured in detail. The shape codes for grid sizes $16, 8,$ and 4 are sufficient to produce good retrieval results and the total storage required is approximately 15 percent. On the other hand, for better visualization, the shape codes of grid size 2 or 1, depending upon the quality requirement, will be sufficient, consuming only 8 or 16 percent of storage. It may also be noted that the CPU time requirement decreases appreciably with the increasing grid size, which demonstrates the speed and efficiency of the proposed technique.

In Fig. 9, the retrieved images on the basis of shape codes for different grid sizes are shown. It may be noticed that the ranking of the retrieved images gets refined with the denser grid. For object visualization, shape code of grid size 4 gives a fairly good idea about the object. However, for better quality visualization, the shape code of grid size 2 or 1 may be required, as shown in Fig. 10.

4 Conclusion and Future works

This work introduces a novel technique for determining the multiresolution shape codes of a binary image using the classical properties of isothetic polygons. An efficient retrieval scheme is designed and implemented

Table 1: Results (average storage and CPU time per image) for databases D1 and D2

	D1 (1034 images)			D2 (110 images)		
	Storage required		CPU Time	Storage required		CPU Time
grid size	bytes	%	millisecs.	bytes	%	millisecs.
16	51363	0.60	22.10	456	0.05	25.17
8	126113	1.49	31.04	12338	1.37	34.79
4	298943	3.53	52.40	32038	3.55	64.85
2	677208	8.00	105.97	71532	7.93	123.32
1	1422700	16.80	227.70	151296	16.78	265.96
Total	2576327	30.41	439.21	267660	29.70	514.07

to demonstrate the power and versatility of such shape codes, irrespective of database size and diversity. The hierarchical definition of shape codes defined over grid configuration of increasing resolution has an inherent property of capturing the topological features of an image in near-optimal number of iterations, which shows the elegance and strength of the algorithm.

The proposed method produces a very effective indexing scheme for binary logo image databases, as observed in our experiments on two different databases, and in particular, for object type of images. It may not however produce good results for other databases like human face, natural scenes, etc. For gray scale images, proper adaptation of this technique, such as Euler Vector [2], may yield desired results, but this area needs further investigation. Experimentation on storing the shape codes for minimization of bits is another area which is presently under research and would be reported in some future publication. Other efficient algorithms for identifying isothetic polygonal covers are also being studied recently [5].

References

[1] A. Bemporad, C. Filippi, and F. D. Torrisi, *Inner and outer approximations of polytopes using boxes*, Computational Geometry – Theory and Applications, vol. 27, 2004, pp. 151–178.

[2] A. Bishnu, B. B. Bhattacharya, *Stacked Euler vector (SERVE): A gray-tone image feature based on bit-plane augmentation*, IEEE Trans. Pattern Anal. Mach. Intell. 29(2): 350-355 (2007)

[3] A. Biswas, P. Bhowmick, and B. B. Bhattacharya, *TIPS: On Finding a Tight Isothetic Polygonal Shape covering a 2D object*, 14th Scandinavian Conference on Image Analysis (SCIA 2005), LNCS, vol. 3540, pp. 930–939.

[4] A. Biswas, P. Bhowmick, and B. B. Bhattacharya, *MuSC: Multigrid Shape Codes and their applications to image retrieval*, International Conference on Computational Intelligence and Security: CIS 2005, Xian, China, December 15 - 19, Lecture Notes in Computer Science (LNCS), Springer, vol. 3801, pp. 1057-1063.

[5] A. Biswas, P. Bhowmick, and B. B. Bhattacharya, *Construction of isothetic covers of a digital object: a combinatorial approach*, Manuscript submitted for publication, 2007.

[6] H. Freeman, *On the encoding of arbitrary geometry configurations*, IRE Trans. Electron. Comput., vol. 10, pp. 260–268, 1961.

[7] P. Gerken, *Object-based analysis-synthesis coding of image sequences at very low bit rates*, IEEE Trans. Circuits and Systems for Video Technology, vol. 4, no. 3, pp. 228–235, 1994.

[8] M.-K. Hu, *Visual pattern recognition by moment invariants*, IRE Transactions on Information Theory, vol. 8, pp.179–187, 1962.

[9] T. Kaneko and M. Okudaira, *Encoding of arbitrary curves based on the chain code representation*, IEEE Trans. Commun., vol. 33, no. 7, pp. 697–707, 1985.

[10] A. K. Katsaggelos, L. P. Kondi, F. W. Meier, J. Ostermann, and G. M. Schuster, *MPEG-4 and rate-distortion-based shape-coding techniques*, Proc. IEEE, vol. 86, no. 6, pp. 1126–1154, 1998.

[11] A. Kaup and J. Heuer, *Polygonal shape descriptors – an efficient solution for image retrieval and object localization*, Proc. 34th Asilomar Conference on Signals, Systems, and Computers, 2000.

[12] F. Mokhtarian and A. K. Mackworth, *A theory of multiscale, curvature-based shape representation for planar curves*, IEEE Trans. PAMI, vol. 14, no. 8, pp. 789–805, 1992.

[13] F. Mokhtarian, S. Abbasi, and J. Kittler, *Efficient and robust retrieval by shape content through curvature scale space*, Image DataBases and Multi-Media Search, pp. 51–58, World Scientific Publishing, Singapore, 1997.

[14] K. J. O Conell, *Object-adaptive vertex-based shape coding method*, IEEE Trans. Circuits and Systems for Video Technology, vol. 7, no. 1, pp. 251–255, 1997.

[15] S. K. Pal and P. Mitra, *Pattern Recognition Algorithms for Data Mining*, Chapman and Hall/CRC Press, Bocan Raton, FL (2004).

[16] G. M. Shuster, A. K. Katsaggelos, *An optimal polygonal boundary encoding scheme in the rate distortion sense*, IEEE Trans. Image Processing, vol. 7, no. 1, pp. 13–26, 1998.

[17] *Facsimile coding schemes and coding functions for group 4 facsimile apparatus*, CCITT Recommendation T.6, 1994.

[18] *Coded Representation of Picture and Audio Information Progressive Bi-Level Image Compression*, ISO Draft 11544, 1992.

[19] *Information Technology Coded Representation of Picture and Audio Information Lossy/Lossless Coding of Bi-Level Images*, ISO FCD. 14492, 1999.