Geometric Characterization of Digital Objects

Algorithms and Applications to Image Analysis

Arindam Biswas

Advisor
Professor Bhargab B. Bhattacharya

A thesis submitted for the degree of
Doctor of Philosophy
in
Computer Science

Indian Statistical Institute
Kolkata 700108, India
September 2008

Dedication

To my teachers

Acknowledgements

At first I would like to express my thanks and deep sense of gratitude to my thesis supervisor, Professor Bhargab B. Bhattacharya for his valuable guidance through this journey. I have been indeed fortunate in being able to pursue my research under his guidance. The long discussions, I have had with him led to many interesting observations, results, and it had been very educative for me.

I would like to acknowledge in this thesis the following distinguished researchers, Dr. Partha Bhowmick, Dr. Arijit Bishnu, Dr. Tinku Acharya, Professor Malay K. Kundu, Professor C. A. Murthy, Professor Subhas C. Nandy, and Dr. Sandip Das, with whom I have co-authored several publications and patents reported in this thesis.

Dr. Partha Bhowmick, a collaborator and a great friend of mine, has been an integral part of the research work reported in this thesis. We have spent many hours together discussing different problems and their solution strategies. The interactions had always been very inspiring and fruitful. It has been an wonderful experience to have Partha beside me during this period. I convey very special thanks to him.

Ever since I entered into the arena of academics, Dr. Arijit Bishnu, a very good friend of mine, has been very supportive in my academic endeavors. I derived great enthusiasm from the company of Arijit, Subhas-da (Professor Subhas C. Nandy) and Sandip-da (Dr. Sandip Das). I thank all of them for their kind support for a person who was new to academics.

I would also like to thank Late Professor Prasanta Kr. Nandi, Professor Amit Das, and Professor Uma Bhattacharya for providing me uninterrupted laboratory facilities to carry out my research work after I had joined the Department of Computer Science and Technology, Bengal Engineering and Science University, Shibpur, Howrah, India. I'm also grateful to the other faculty members, technical personnel, and nontechnical staff of this department for necessary assistance and cooperation. Specially, I express my gratitude to Professor Amit Das for his cheerful words of inspiration which helped me pursuing my research with renewed spirit. He always provided the right ambience for research with his pleasant personality.

I remember the desire of my father Late P. Biswas, who wanted me to pursue a career in academics, which inspired me further in carrying out this work.

Finally, I would like to thank my wife, Enakshi, who has been a continuous source of inspiration for me in pursuing my doctoral dissertation during these years. I would also like to thank my mother and my in-laws who have been ever so supportive in this endeavor.

Arindam Biswas
`abiswas@cs.becs.ac.in`
September 2008, India

Abstract

Several problems of characterizing a digital object, and particularly, those related to boundary description, have been studied in this thesis. New algorithms and their applications to various aspects of image analysis and retrieval have been reported. A combinatorial technique for constructing the outer and inner isothetic covers of a digital object has been developed. The resolution of the background 2D grid can be changed by varying the grid spacing, and this procedure can be used to extract shape and topological information about the object. Next, an algorithm has been designed for constructing the orthogonal (convex) hull of a digital object against a background grid by employing a combinatorial technique. On the application side, it has been shown how the shape code derived from the isothetic cover with varying resolutions can be used in image analysis and retrieval. The nature of the isothetic cover also provides a measure of the shape complexity of the underlying object. Several properties of isothetic cover are shown to be useful in defining features of handwritten characters. An algorithm for polygonal approximation of a thick digital curve based on its outer and inner isothetic covers has also been developed.

To describe the contour of a digital object by an unordered set of points of optimal or sub-optimal size, a new concept called the "pointillist ensemble", is introduced in this thesis. This is derived from the end points of digital straight pieces describing the curve satisfying certain neighborhood properties. This is based on the idea of pointillism, a style of painting with dots, developed by the Neo-Impressionist painters of France. The procedure of reconstructing the object from the ensemble is also presented. Finally, other topological properties of an object, namely, those based on the Euler number have been used to devise an algorithm for efficient image indexing. This has been formulated deploying a novel concept of randomized spatial masking of the digital object.

The algorithms, the proofs of their correctness, and the relevant experimental results on diverse databases have been reported to substantiate the theoretical findings presented in this thesis.

Contents

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

The eight chapters comprising this doctoral thesis present several algorithms and their applications to image analysis using geometric representation and characterization of objects in the digital plane. Algorithms presented in this thesis are mostly combinatorial in nature and have been designed in the perspective of digital geometry. The applications involving geometric characterization have been shown to produce encouraging results, which are relevant to digital image processing, shape analysis, image indexing, etc. This chapter summarizes the overall flow of the thesis.

1.1 Geometric Characterization of Digital Objects

Characterization of digital objects has gained immense importance with the advent of digital imaging. Concomitantly, several new areas of theoretical interest have emerged, which include,

(i) digital calculus [Nakamura and Aizawa (1984), Nakamura and Rosenfeld (1997)];

(ii) digital geometry [Bertrand *et al.* (2001), Klette (1982, 2001a,b), Klette and Rosenfeld (2004a), Rosenfeld (1974)];

(iii) discrete tomography [Balazs (2008), Gesù *et al.* (2008), Gesù and Valenti (2004), Herman (1980), Herman and Kuba (1999), Lorentz (1949)];

(iv) digital topology [Brimkov and Barneva (2004), Kong (2001), Kong and Rosenfeld (1996), Rosenfeld (1979)];

Such geometric characterization of digital objects may be used in various real-world applications as listed below:

(i) VLSI layout design [Nandy and Bhattacharya (2000), Preparata and Shamos (1985)];

(ii) robot grasping [Gatrell (1989), Kamon *et al.* (1995), Lengyel *et al.* (1990), Morales *et al.* (2002)];

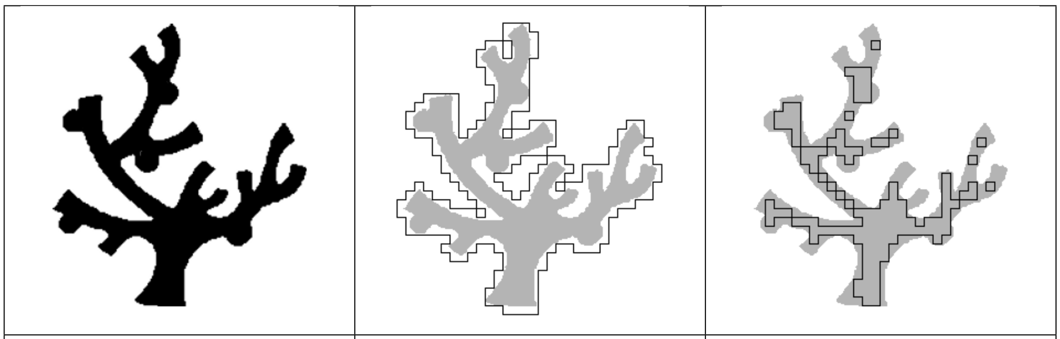
(iii) rough sets [Pal and Mitra (2004a,b), Pawlak (1990), Tadrat *et al.* (2007), Zhu and Wang (2007)];
(iv) document analysis [Antonacopoulos and Meng (2002), Gatos and Mantzaris (2000), Gatos *et al.* (1999), Jain and Yu (1998), Yacoub *et al.* (2005)];
(v) digital imaging and modeling [Gonzalez and Woods (1993), Li and Holstein (2003), Rosenfeld and Kak (1982), Tian *et al.* (2003)];
(vi) studies in digital planarity [Brimkov and Barneva (2005), Brimkov *et al.* (2007)];
(vii) analysis of discrete curvature [Anderson and Bezdek (1984), Fischler and Wolf (1994), Freeman and Davis (1977), Teh and Chin (1989), Wuescher and Boyer (1991)]
(viii) shape analysis [Antoine *et al.* (1996), Bhowmick *et al.* (2007a), Biswas *et al.* (2005b, 2007b), Buvaneswari and Naidu (1998), Pavlidis *et al.* (1997), Rosin (2000, 2003)]
(ix) image indexing and retrieval [Biswas *et al.* (2004, 2007a), Ducksbury and Varga (1997), Gu and Tjahjadi (1999), Mokhtarian and Mohanna (2002), Wolf *et al.* (2000)].

1.2 Scope of the Thesis

In this thesis, we have studied the following problems: (i) construction of tight isothetic covers and orthogonal hull of a digital object and their applications to shape analysis, (ii) polygonal approximation of a thick digital curve, (iii) describing the contour of a digital object using a pointillist ensemble, and (iv) image indexing based on connectedness properties of a digital object. For each of these problems, several combinatorial and geometric properties have been explored, on the basis of which new algorithms have been developed and relevant applications have been demonstrated.

1.2.1 Tight Isothetic Covers of a Digital Object

The shape of a digital object can be approximately captured in the form of a tight isothetic cover taken from the exterior (outer cover) or from the interior (inner cover). Several applications related to these representations in shape analysis and retrieval have been demonstrated. An isothetic cover of a digital object, which comprises a set of isothetic polygons, not only specifies a simple representation of the object but also provides approximate information about its structural content and geometric characteristics. Such covers can be used in various applications like image retrieval, deriving shape complexity, outer and inner approximation in rough sets, VLSI design layout, etc. In Chapter 2, we have proposed a combinatorial algorithm, which constructs the isothetic cover (outer or inner)



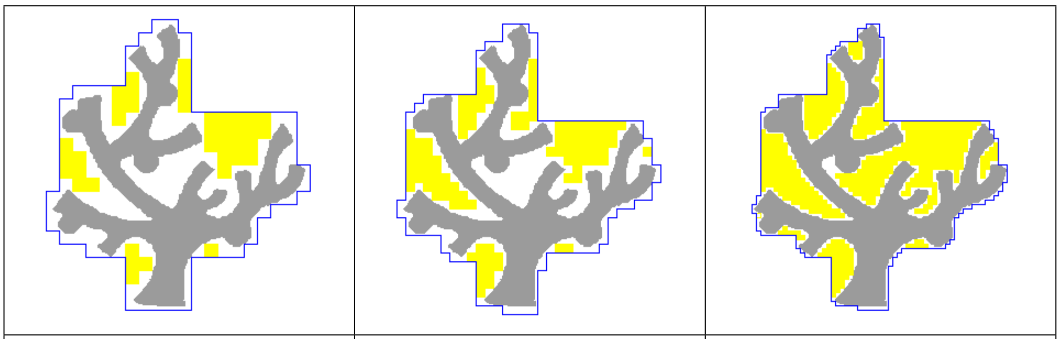
(a) A digital object. (b) Outer Isothetic Cover. (c) Inner Isothetic Cover.

Figure 1.1: The isothetic covers of a digital object.

of a given digital object [Biswas *et al.* (2005b)]. A digital object (digitized on unit-sized grid, i.e., for $g = 1$), and its outer and inner isothetic covers for $g = 8$, are shown in Fig. 1.1. The isothetic covers can be used for deriving shape-codes [Biswas *et al.* (2005a, 2008)], shape complexity measure [Biswas *et al.* (2007b)], for ranking optical character prototypes in a large database [Bhowmick *et al.* (2007b)], and for polygonal approximation of thick digital curves [Bhowmick *et al.* (2006)].

1.2.2 Orthogonal Hull of a Digital Object

Akin to the notion the isothetic cover, which provides a good approximation of the shape of a digital object, another way of characterizing the object is to derive its orthogonal (convex) hull, which further approximates its shape. Orthogonal hull may be employed in various applications, such as, analysis of land-mark data, shape analysis and classification, computer vision and pattern recognition [Bookstein (1991), Costa and R. M. Cesar (2001), Hyde *et al.* (1997), Pitty (1984)], in discrete tomography [Balazs (2008)], etc. Further, a digital object may be characterized by its convex deficiency tree, which can be derived from its orthogonal hull [Gonzalez and Woods (1993)], and can be used to facilitate image indexing and retrieval. In Chapter 3, an algorithm has been proposed which finds the orthogonal hull of a given digital object in $\mathbb{Z}^2$, such that the hull edges lie on a set of equally spaced horizontal and vertical grid lines. Fig. 1.2 shows the orthogonal hulls of the same object shown in Fig. 1.1, for different grid sizes. The orthogonal hulls, with decreasing grid size, give a more accurate characterization of the object, thereby enabling a tunable procedure. The regions shown in yellow touching the boundary of the outer



(a) $g = 12$ (b) $g = 8$ (c) $g = 4$

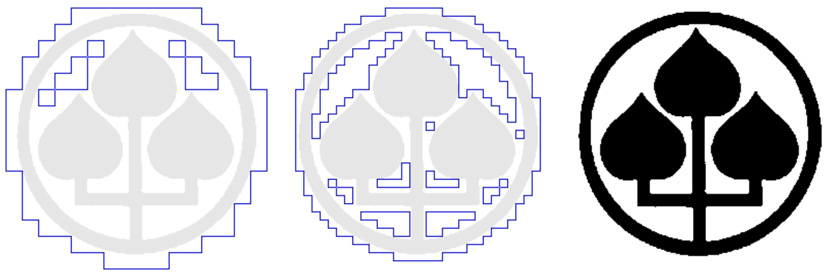
Figure 1.2: The orthogonal hulls of a digital object for different grid sizes.

isothetic cover of the object have been removed by the algorithm, in order to construct the orthogonal hull.

1.2.3 Shape Analysis using Isothetic Covers

In Chapter 4 of this thesis, we demonstrate three different applications [Bhowmick *et al.* (2007b), Biswas *et al.* (2007b, 2008)] of outer isothetic cover for shape analysis. The first one presents an elegant shape coding technique, which captures 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. Such a Multigrid Shape Code (MuSC) of an object is shown in Fig 1.3, where, the code is written below the image. We have also presented an image retrieval scheme based on such shape codes (Sec. 4.2.4). The second application derives a shape complexity measure by merging the consecutive concavity and convexity of the outer isothetic cover in multiple tiers to capture the spatial complexity. The method uses only integer operations (comparison and addition), does not need to extract the boundary, and is therefore, very fast, efficient, and robust in the presence of noise.

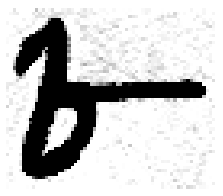
As a third application we introduce a new concept of conferring ranks on the prototypes of a large database of handwritten characters based on isothetic chord lengths. The proposed scheme has a striking potential in deciding the relative merit and usefulness of a particular prototype with respect to the other prototypes of the corresponding character in the relevant database. Several quantitative measures along with supporting experimental results have been discussed, which justify the strength and efficiency of the proposed



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

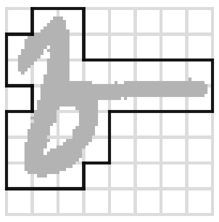
$$\begin{array}{ll}
\text{MuSC:}\,(\underline{5})\ \ (\underline{0}) & (11000001)(01000010)\cdots(01000110)(0^8) \ \triangleright \text{ end of 1st polygon}\\
\qquad\ \ (\underline{7})\ \ (\underline{2}) & (01000010)(01000001)\cdots(01000010)(0^8) \ \triangleright \text{ end of 2nd polygon}\\
\qquad\ \ (\underline{12})(\underline{7}) & (11000001)(01000001)\cdots(01000001)(0^8)(0^8) \ \triangleright \text{ end of } g_1 = 16\\
 & \vdots\\
\qquad\ \ (\underline{12})(\underline{22}) & (11000001)(01000001)\cdots(01000001)(0^8)(0^8) \ \triangleright \text{ end of } g_2 = 8
\end{array}$$

Figure 1.3: A sample logo image and its Multigrid Shape Codes (MuSC) corresponding to its OIC.



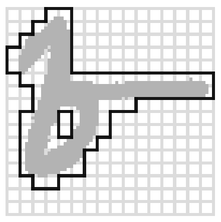
(a)

a prototype of the Bangla numeric character ‘**8**’.



(b)

$g = 8$.
$\mathfrak{Y} = \langle 2, 3, 8, 8, 8, 4, 4, 3 \rangle$.



(c)

$g = 4$.
$\mathfrak{Y} = \langle 2, 3, 4, 5, 5, 16, 15, 14, 9, 6, 7, 6, 5, 5, 2 \rangle$.

Figure 1.4: OICs and their total horizontal chord lengths (elements of $\mathfrak{Y}$) for a typical prototype of the Bangla numeral ‘**8**’.

method. In Fig. 1.4, we have shown the values of horizontal chord lengths of a typical prototype of Bangla numeral ‘**8**’ for two different grid sizes.

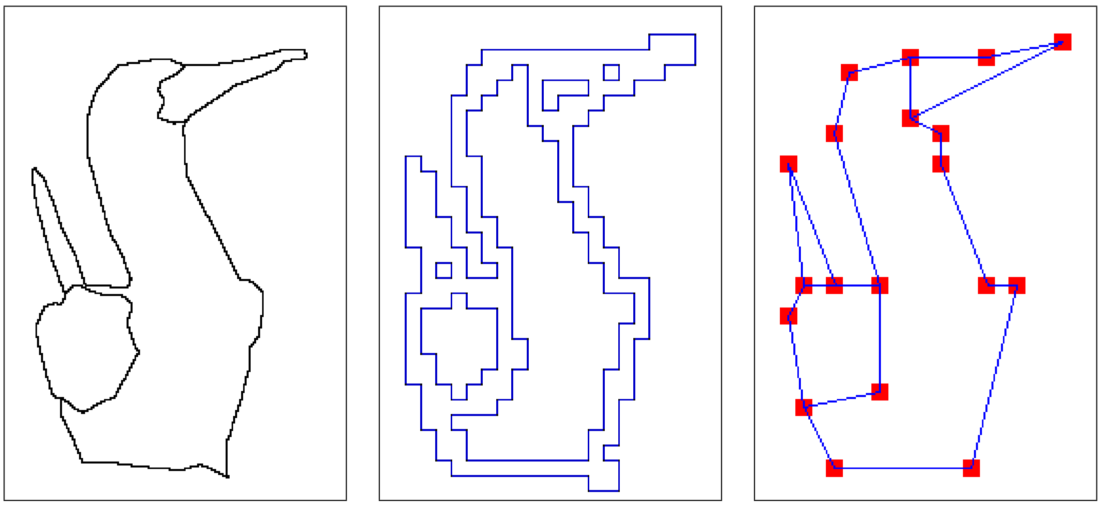


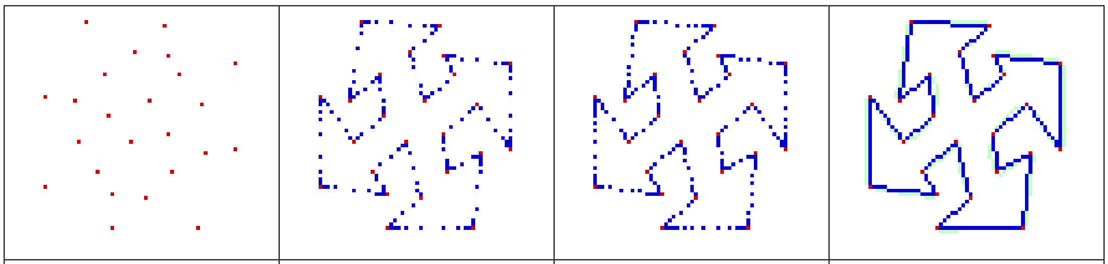
Figure 1.5: Polygonal approximation of an image "duck" for $g = 8$.

1.2.4 Polygonal Approximation of Thick Digital Curves

Polygonal approximation of digital curves have received enormous attention since the inception of successful rasterization of curves and objects in the digital space. Several algorithms have been proposed for approximating a given digital curve. However, these algorithms use thinning as a preprocessing step before approximating a digital curve. In Chapter 5, we have introduced a novel thinning-free algorithm for polygonal approximation of an arbitrarily thick digital curve. The proposed method is demonstrated in Fig. 1.5 with an example, where, the algorithm is run on a digital object (left), showing the contour of a "duck" image. The corresponding cellular envelope (middle), and the final polygonal approximation (right) are also shown. The encapsulation of a digital curve by the cellular envelope enables the proposed method to approximate arbitrarily thick real-world curves/curve-shaped objects, even in the presence of noise and unexpected disconnectedness.

1.2.5 Pointillist Ensemble of a Digital Object

In Chapter 6 of this thesis, we address the problem of representing the contour of a digital object by an unordered set of points. Such representation of an object finds applications in many diverse areas, namely, video encoding, biometrics, deformation analysis, shape-based segmentation, etc. Pointillism, a movement of painting with dots that would blend



(a) Vertex set, $\mathcal{P}$, of a digital polygon.

(b) Ensemble for minimum pointillist factor ($\phi = 1$).

(c) Ensemble for a larger pointillist factor ($\phi = 2$).

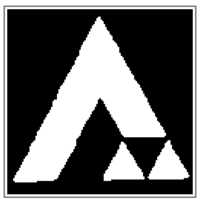
(d) Actual object.

Figure 1.6: The digital object (d), the vertex set of the digital polygon corresponding to (d), and the point ensembles, (b) and (c), for different pointillist factors.

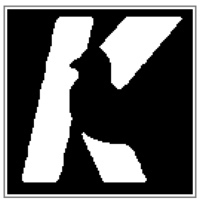
in the viewers eye, was developed by certain Neo-Impressionists of France late in the 19th century. In Chapter 6, we have introduced the pointillistic approach to construct a minimal ensemble of unordered points to describe the contour of a digital object. Fig. 1.6 (d) shows an object for which the vertex set of the corresponding polygon is shown in Fig. 1.6 (a), which hardly gives an impression of the underlying object. The point ensemble Fig. 1.6 (b) which is a superset of the vertex set, represents the object more prominently. Fig. 1.6 (c) shows the pointillist ensemble with an increased pointllist factor.

1.2.6 Image Indexing with Connectivity Features

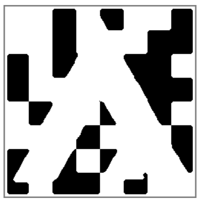
In Chapter 7, we have studied the connectivity features of digital object and their applications to image indexing. Topological properties of a digital image [Klette and Rosenfeld (2004a)] typically represent some information about the underlying geometric shape of the image. One such important topological feature of a binary image is its Euler Number, which is defined as the number of connected components minus the number of holes present in the image [Gonzalez and Woods (1993), Pratt (1978)]. In Chapter 7, we have presented a mechanism for image indexing using a novel concept of randomized spatial masking to modify the connectivity features, related to the Euler number. This procedure, combined with a fuzzy approach, yields a unique feature vector for each image in the database. The algorithm is based on *xor*-ing the image bit-plane with a few pseudo-random synthetic masks, and its novelty lies in computing the connectivity-based feature vectors iteratively, depending on the size and diversity of the image database. For example, in Fig. 1.7, two images are shown with same Euler number, and with the same number of connected



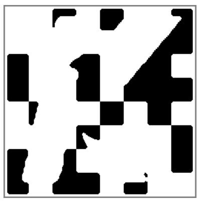
$\mathcal{I}_{109} : \langle -2 \rangle \langle 1, 3 \rangle$



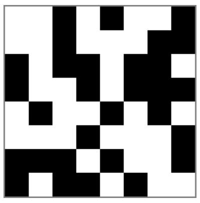
$\mathcal{I}_{274} : \langle -2 \rangle \langle 1, 3 \rangle$



$\mathcal{J}^{(1)}_{109}$:
$\langle -2 \rangle \langle 1, 3 \rangle \langle 7, 1 \rangle$



$\mathcal{J}^{(1)}_{274}$:
$\langle -2 \rangle \langle 1, 3 \rangle \langle 7, 0 \rangle$



Mask: $\mathcal{M}^{(1)}_{8}$

Figure 1.7: Two images, having the same Euler number and number of connected components and holes (first row), are *xor*-ed with a random mask (second row rightmost) to produce images (second row left and middle) with distinct pairs of number of connected components and holes.

components and holes, each. Thus, they cannot be discriminated using these features. However, when these two images are *xor*-ed with a random mask, they produce different pairs of connected components and holes, namely $\langle 7, 1 \rangle$ and $\langle 7, 0 \rangle$, and therefore, yield two distinct feature vectors.

1.3 Organization of the Thesis

We have studied various aspects of geometric characterization of a digital object, starting with the isothetic cover, the orthogonal hull, followed by several applications of these representations, which are discussed in Chapter 2, Chapter 3 and Chapter 4 respectively. In Chapter 5, we have studied digital curves and presented a polygonal approximation algorithm. The problem of representing a digital curve by an unordered set of points that preserves visual perception of the curve and aids its subsequent reconstruction, has been studied in Chapter 6. Next, we have deployed the connectivity features for devising a mechanism for image indexing in Chapter 7. Finally, in Chapter 8, we conclude the thesis with critical remarks and discuss some open problems and other future issues.

Chapter 2

Construction of Isothetic Covers of a Digital Object

2.1 Introduction

Determination of the (minimum-)maximum-area outer(inner) isothetic cover corresponding to a 2D digital object is of relevance to various fields. Given a set of isothetic grid lines under the object plane, an isothetic cover corresponds to a collection of isothetic polygons, which bears a structural and geometric relation with the concerned object, and hence can be useful to many interesting applications such as VLSI layout design, robot grasping and navigation, rough sets, document image analysis. In VLSI layout design, computation of a minimum-area safety region, referred as the classical *safety zone problem* [Nandy and Bhattacharya (2000)], may be necessary to ensure the correctness of design rules before fabricating an integrated circuit. The trade-off lies between the minimization of total area of the fabricated parts (an obvious economic constraint) and the necessary electrical relationship (insulation or contact) in the presence of possible production fluctuations [Preparata and Shamos (1985)]. In robotics, identification of free configuration space (path-planner) is a pertinent problem in robot navigation. For example, a real-time robot motion planner often uses standard graphics hardware to rasterize the configuration space into a series of bitmap slices, and then applies a dynamic programming technique to calculate paths in this rasterized space [Lengyel *et al.* (1990)]. The motion paths produced by the planner should be minimal with respect to the Manhattan distance (L_1) metric. Similarly, for grasping a 3D object, its outer isothetic cover may be helpful, as the mechanical fingers of a robot may be constrained by only axis-parallel movements [Gatrell (1989), Kamon *et al.* (1995), Morales *et al.* (2002)]. In many applications of rough sets, computation of the lower and upper approximations of a rough set is required [Pal and Mitra (2004a,b), Tadrat *et al.* (2007), Zhu and Wang (2007)]. For example, in image mining [Liu *et al.* (2004)], a challenging problem is to discover valid, novel, potentially useful, and ultimately understandable knowledge from a large image database, in order to overcome the curse of dimensionality. Solutions, using rough-set concepts, mainly include several partitioning and dimension-reduction algorithms, where the (possibly not equi-spaced) demarcating lines (analogous to the background grid lines used while finding isothetic covers) are specified by the corresponding feature values (low, medium, high, etc.). Subsequently, a tight isothetic cover of the region of interest can be obtained following these demarcating lines. Deriving the electronic version of a paper document for the purpose of storage, retrieval, and interpretation, requires an efficient representation scheme. A document representation involves the steps of skew detection, page segmentation, geometric layout analysis, and logical layout analysis, for which isothetic polygons can be used [Gatos and Mantzaris (2000), Gatos *et al.* (1999), Jain and Yu (1998)]. For example, in the page segmentation method [Akindele and Belaid (1993)], a document page image is cut into polygonal blocks using the inter-column and the inter-paragraph gaps as horizontal and vertical lines, followed by the construction of simple isothetic polygonal blocks using an intersection table. Isothetic polygons can also be used for ground truthing of complex documents [Antonacopoulos and Meng (2002), Yacoub *et al.* (2005)].

The problem addressed in this chapter is stated as follows. Given a 2D digital object S (Def. 2.2.4) registered on a set of horizontal and vertical grid lines $\mathcal{G}$ (Def. 2.2.5), the problem is to find a tight (outer and inner) isothetic cover (Def. 2.2.9 and Def. 2.2.10) of S. Clearly, the isothetic covers depend on the registration of the object with the underlying grid. The structure and complexity of the covers are likely to change if the object is made to translate on the fixed grid or if the grid size is altered. Fig. 2.1 shows the original "bear" image (left), its outer isothetic cover (OIC) and the inner isothetic cover (IIC) in the middle, and the superimposed outer and inner isothetic covers (right). It is seen that the boundary of the object lies in the region OIC minus the interior of IIC.

2.2 Definitions and Preliminaries

Definition 2.2.1 (Digital Plane). *The digital plane, $\mathbb{Z}^2$, is the set of all points having integer coordinates in the real plane $\mathbb{R}^2$. A point in the digital plane is called a* digital point*, or called a* pixel *in the case of a digital image.*

Henceforth, the terms "point" and "digital point" will be used interchangeably.

Definition 2.2.2 (k-connectedness). *The set of four horizontally and vertically adjacent*

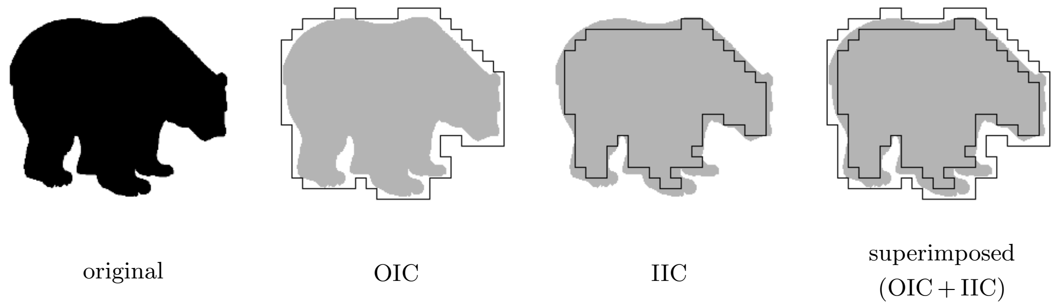


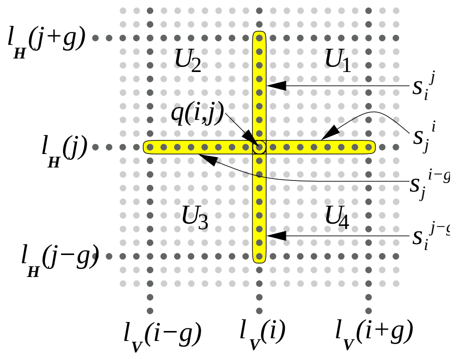
Figure 2.1: The outer isothetic cover (OIC) and the inner isothetic cover (IIC), corresponding to the image "bear".

points of a point $p(x,y) \in \mathbb{Z}^2$ *is called the* 4-neighborhood *of* p*, which is given by* $N_4(p) := \{(x',y') : (x',y') \in \mathbb{Z}^2 \wedge |x-x'| + |y-y'| = 1\}$*. The set of all eight neighbors, i.e., the four horizontal and vertical neighbors, and the four diagonal neighbors, defines the* 8-neighborhood *of* p*, given by* $N_8(p) := \{(x',y') : (x',y') \in \mathbb{Z}^2 \wedge \max(|x-x'|, |y-y'|) = 1\}$*. Each point in* $N_k(p)$ *is said to be a* k-neighbor *(*$k = 4$ *or* 8*) of* p*. Two points* p *and* q *are* k-connected *in a digital set* $S \subset \mathbb{Z}^2$ *(Def. 2.2.4) if and only if there exists a sequence* $\langle p := p_0, p_1, \ldots, p_n := q\rangle \subseteq S$ *such that* $p_i \in N_k(p_{i-1})$ *for* $1 \leq i \leq n$*. For any point* $p \in S$*, the set of points that are* k*-connected to* $p \in S$ *is called a* k-connected component *of* S*. In other words, a* k*-connected component of a nonempty set* $S \subseteq \mathbb{Z}^2$ *is a maximal* k*-connected set of* S*. If* S *has only one connected component, it is called a* k-connected set*.*

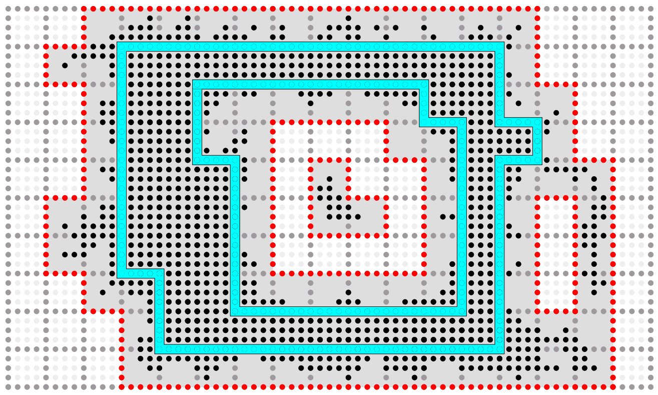
Definition 2.2.3 (Isothetic distance)**.** *The* (isothetic) distance between two points, $p(i_p, j_p)$ *and* $q(i_q, j_q)$*, is the Minkowski norm* L_∞ *[Klette and Rosenfeld (2004a)], which is given by* $d_\top(p,q) = \max\{|i_p - i_q|, |j_p - j_q|\}$*. The* distance of a point p from an object S *is* $d_\top(p,S) = \min\{d_\top(p,q) : q \in S\}$*, and the* distance between two connected components S_1 *and* S_2 *is* $d_\top(S_1,S_2) = \min\{d_\top(p,q) : p \in S_1, q \in S_2\}$*.*

Definition 2.2.4 (Digital object)**.** *A* digital object *(henceforth referred as an* object*) is a finite subset of* $\mathbb{Z}^2$*, which consists of one or more* k*-connected components.*

In this work, each component (8-connected, i.e., $k = 8$) of an object S may contain one or more holes. A hole of S is a $\bar{k}$-connected component of $\mathbb{Z}^2 \smallsetminus S$ which is completely surrounded by one of the components of S (for $k = 8$, we have $\bar{k} = 4$, and vice versa) [Rosenfeld and Kak (1982)]. Let $\mathcal{I}$ be a finite rectangular subset of $\mathbb{Z}^2$, which contains the entire object S, then the complement of S is given by $S' := \mathcal{I} \smallsetminus S$.



(a) UGBs incident at $q(i, j)$.



(b) OIC and IIC.

Figure 2.2: (a) Four neighboring UGBs, namely $U_1 = \text{UGB}(i, j)$, $U_2 = \text{UGB}(i - g, j)$, $U_3 = \text{UGB}(i - g, j - g)$, and $U_4 = \text{UGB}(i, j - g)$, incident at a grid point $q(i, j)$. (b) OIC (in red) and IIC (in cyan) corresponding to an object S (in black) having two connected components.

Definition 2.2.5 (Digital grid)**.** *A* digital grid *(henceforth referred simply as a* grid*)* $\mathcal{G} := (\mathcal{H}, \mathcal{V})$ *consists of a set* $\mathcal{H}$ *of* horizontal (digital) grid lines *and a set* $\mathcal{V}$ *of* vertical (digital) grid lines*, where,* $\mathcal{H} = \{\ldots, l_H(j-2g), l_H(j-g), l_H(j), l_H(j+g), l_H(j+2g), \ldots\} \subset \mathbb{Z}^2$ *and* $\mathcal{V} = \{\ldots, l_V(i-2g), l_V(i-g), l_V(i), l_V(i+g), l_V(i+2g), \ldots\} \subset \mathbb{Z}^2$*, for a* grid size*,* $g \in \mathbb{Z}^+$*. Here,* $l_H(j) := \{(i', j) : i' \in \mathbb{Z}\}$ *denotes the horizontal grid line and* $l_V(i) := \{(i, j') : j' \in \mathbb{Z}\}$ *denotes the vertical grid line intersecting at the point* $(i, j) \in \mathbb{Z}^2$*, called the* grid point*, where* i *and* j *are multiples of* g*.*

Definition 2.2.6 (Grid segment)**.** *A (digital)* grid segment *(horizontal/vertical) is the set of points on a (horizontal/vertical) grid line between and inclusive of two consecutive grid points. Thus, the* horizontal grid segment *belonging to* $l_H(j)$ *and lying between two consecutive vertical grid lines,* $l_V(i)$ *and* $l_V(i+g)$*, is* $s_j^i = \{(i', j) \in l_H(j) : i \le i' \le i + g\}$*. Similarly, the* vertical grid segment *belonging to* $l_V(i)$ *between* $l_H(j)$ *and* $l_H(j + g)$ *is* $s_i^j = \{(i, j') \in l_V(i) : j \le j' \le j + g\}$*.*

Definition 2.2.7 (Unit Grid Block(UGB))**.** *The horizontal digital line* $l_H(j)$ *divides* $\mathbb{Z}^2$ *into two half-planes (each closed at one side by* $l_H(j)$*), which are given by* $h_H^+(j) := \{(i', j') \in \mathbb{Z}^2 : j' \ge j\}$ *and* $h_H^-(j) := \{(i', j') \in \mathbb{Z}^2 : j' \le j\}$*. Similarly,* $l_V(i)$ *divides* $\mathbb{Z}^2$ *into two half-planes, which are* $h_V^+(i) := \{(i', j') \in \mathbb{Z}^2 : i' \ge i\}$ *and* $h_V^-(i) := \{(i', j') \in \mathbb{Z}^2 : i' \le i\}$*. Then the set, given by* $\big(h_V^+(i) \cap h_V^-(i+g)\big) \cap \big(h_H^+(j) \cap h_H^-(j+g)\big)$*, is defined* *as the* unit grid block, UGB(i,j). *The* interior of $U_1 :=$ UGB(i,j) *is given by* $U_1 \smallsetminus \left(s_i^j \cup s_{i+g}^j \cup s_j^i \cup s_{j+g}^i\right)$.

Hence, for a given grid point, $q(i,j)$, there are four neighboring UGBs, namely, $U_1 :=$ UGB(i,j), $U_2 :=$ UGB$(i-g,j)$, $U_3 :=$ UGB$(i-g,j-g)$, and $U_4 :=$ UGB$(i,j-g)$, as shown in Fig. 2.2. Two adjacent UGBs share a common (horizontal or vertical) grid segment. For example, the vertical grid segment shared between U_1 and U_2, is given by $s_i^j := U_1 \cap U_2$. Thus, U_1, U_2, U_3, and U_4 share a common point, namely the grid point $q(i,j)$.

Definition 2.2.8 (Isothetic polygon)**.** *A* (isothetic) polygon P *is a simple polygon (i.e., with non-intersecting sides) of finite size in* $\mathbb{Z}^2$ *whose alternate sides are subsets of the members of* $\mathcal{H}$ *and* $\mathcal{V}$. *The polygon* P*, hence given by a finite set of* UGB*s, is represented by the (ordered) sequence of its vertices, which are grid points. The* border B_P *of* P *is the set of points belonging to its sides. The* interior *of* P *is the set of points in the union of its constituting* UGB*s excluding the border of* P.

An isothetic polygon P can be classified into:

- *Outer polygon*: $P \cap S \neq \emptyset$; for each $p \in B_P$, $0 < d_\top(p, P \cap S) \leqslant g$.
- *Inner polygon*: $P \cap S \neq \emptyset$; for each $p \in B_P$, $0 < d_\top(p, S' \smallsetminus P) \leqslant g$.
- *Outer hole polygon*: for each $p \in B_P$, $0 < d_\top(p, S \smallsetminus P) \leqslant g$.
- *Inner hole polygon*: for each $p \in B_P$, $0 < d_\top(p, P \cap S') \leqslant g$.

Roughly speaking, an outer polygon contains one or more components of S such that its border is a subset of S' and the number of its constituting UGBs is minimum. Similarly, an inner polygon is contained in exactly one component of S such that its border is a subset of S and the number of its constituting UGBs is maximum. An outer hole polygon lies in a hole (or a sufficiently large background region with a narrow opening in S) and its border is a subset of S', whereas, an inner hole polygon contains a hole and its border is a subset of S.

Definition 2.2.9 (Outer Isothetic Cover)**.** *The* outer (isothetic) cover *(OIC), denoted by* $\overline{P}(S)$*, is a set of outer polygons and (outer) hole polygons, such that the region, given by the union of the outer polygons minus the union of the interiors of the hole polygons, contains a* UGB *if and only if it has object occupancy (i.e., has a nonempty intersection with* S*).*

Definition 2.2.10 (Inner Isothetic Cover)**.** *The* inner (isothetic) cover *(IIC), denoted by* $\underline{P}(S)$*, is a set of inner polygons and (inner) hole polygons, such that the region, given by the union of the inner polygons minus the union of the interiors of the hole polygons, contains a* UGB *if and only if it is a subset of* S*.*

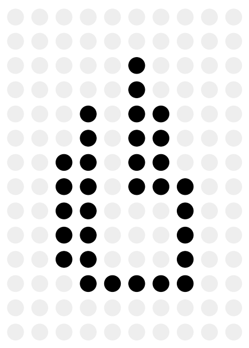
Hence, the OIC of an object consists of minimum number of UGBs and its IIC consists of maximum number of UGBs. As a consequence, given an object S and a grid imposed on S, the OIC and the IIC are respectively of minimum and of maximum areas measured in the grid unit.

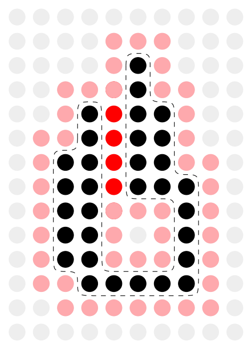
The *interior* of an OIC is given by the union of the interiors of its outer polygons minus the union of its hole polygons. A point, therefore, is said to *lie inside the OIC* if and only if it is an interior point; it is said to *lie on the OIC* if and only if it lies on the border of one of its outer polygons or hole polygons. The OIC and the IIC of an object S having two components are shown in Fig. 2.2. The OIC (shown in red) consists of two outer polygons and two hole polygons, where one hole polygon corresponds to an actual hole of the object and the other to a sufficiently large background region with a narrow opening in S. The IIC polygons (in cyan) are two in number, the larger one being the inner polygon and the smaller one being the hole polygon.

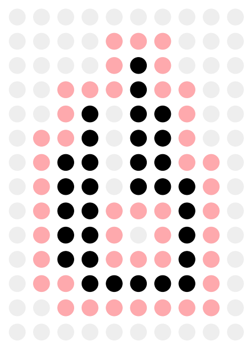
2.3 Related Works

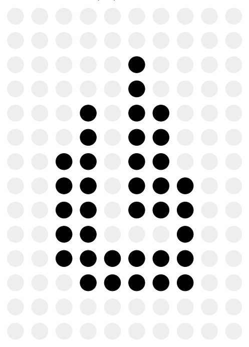
Rosenfeld and Kak (1982) had presented algorithms for finding individual borders between the components of the digital object S and S', where S' is the complement of S. The border is defined as the set of points of S that are 4-adjacent to S'. The border is followed by inspecting the 8-neighbors in counter-clockwise order. The border can also be extracted by the crack following algorithm. Let C and D be the components of S and S' respectively, and let P and Q be 4-adjacent points of C and D, so that (P, Q) defines one of the cracks on the (C, D)-border. Then, depending on how (U, V) (the pair of points facing (P, Q)) belongs to C or D, the crack takes a right, left, or no turn and the next crack, (P', Q'), is determined. The algorithm terminates when it comes back to the initial pair. This algorithm can find the C-border of D (set of points of D that are 4-adjacent to C) and D-border of C (set of points of C that are 4-adjacent to D).

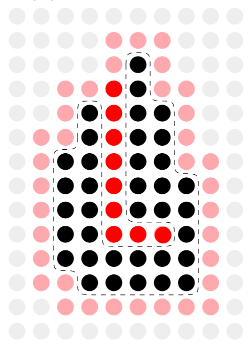
Fig. 2.3(a) shows a digital object S_1. The D-border of C, which consists of the background pixels, constructed by applying the crack following algorithm of Rosenfeld and Kak is shown in Fig. 2.3(b). The red pixels denote the border. It may be noted here that the dark red pixels have been traversed twice while following the crack (shown in dotted


(a) S_1.


(b) D-border of C.


(c) OIC of S_1.


(d) S_2.


(e) D-border of C.

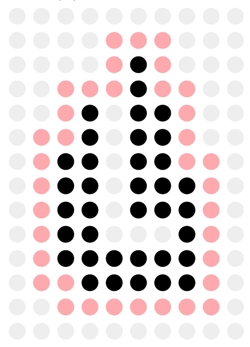

(f) OIC of S_2.

Figure 2.3: Two digital objects S_1 and S_2 (a) and (d); their D-borders (b) and (e), which are traced by Rosenfeld's crack following algorithm; the OICs (c) and (f) as constructed by the proposed algorithm.

line). In fact, we get two polygons, which are connected by a single pixel thick curve. In our proposed algorithm, the crack following algorithm has been modified for finding the outer and inner isothetic covers. The OIC constructed by using this modified algorithm consists of one *outer polygon* and one *outer hole polygon*, as shown in Fig. 2.3(c), where the *grid size* g is taken as 1. Retracing of the same path has been avoided by using a combinatorial technique. Also, in Fig. 2.3 (e) and (f), the D-border of C and the OIC of another object S_2 is shown. The crack following algorithm produces a cover of the object in which the dark red pixels are traced twice when the crack is followed. The OIC avoids such retracing or backtracking, and outputs one *outer polygon*.

For a given digital object S, the outer Jordan digitization $J^+(S)$ is the union of all such 2-cells (grid squares) that have nonempty intersections with S, and the inner Jordan digitization $J^-(S)$ is the union of all 2-cells that are completely contained within S. Jordan digitization (inner/outer) gives the union of cells but does not specify the (inner/outer) cover as a sequence of vertices.

Klette and Rosenfeld demonstrated a similar concept called isothetic *frontier* [Klette and Rosenfeld (2004a)]. However, there exist some differences between an isothetic cover and an isothetic frontier. To find the frontier ϑS of an object S in any metric space $[A, d]$, the (open) *ε-neighborhood* of a point p in A is defined as $U_\varepsilon(p) = \{q : 0 \leq d(p,q) < \varepsilon\}$. Then, $p \in S$ is a *frontier point* of $S \subseteq A$ if and only if, for any $\varepsilon > 0$, $U_\varepsilon(p)$ contains points of S as well as points of $A \setminus S$. In the digital plane, ε is always a positive integer and $A \subseteq \mathbb{Z}^2$. Hence, if $\varepsilon = 1$, then there exists no point $p \in A$ for which $U_\varepsilon(p)$ contains a point of $A \setminus S$. In the case of an isothetic cover, the metric used is isothetic distance $d_\top$ (Def. 2.2.3), and if a point p lies on the cover, then there exist point(s) $q \in S$ as well as point(s) $q' \in S' (:= A \setminus S)$ such that $0 < d_\top(p,q), d_\top(p,q') \leq g$, g being the grid size (as in Def. 2.2.5). Adopting such a policy ensures that both the outer (or the inner) isothetic covers have minimum (maximum) area. Also, the isothetic frontier of an object is defined in a metric space where the background grid is not required. On the contrary, the isothetic cover of an object is identified assuming that the object is registered on the underlying grid with the goal of controlling the complexity of the cover by changing the grid size. For a higher grid size, we get a coarser description of the object, whereas for a lower one, we get a finer description.

In the literature, several other works dealing with boundary approximation, minimum perimeter polygons, and concavity trees can be found. Sklansky (1972b) described an algorithm for computing minimum-perimeter-polygon (MPP) of digitized silhouettes (often referred as cellular complexes). A technique has been described for describing and measuring the concavities of such cellular complexes [Sklansky (1972a)]. It combines the concept of half-cell expansion and the method of finding the cellular convex hulls in order to determine the concavity tree. The algorithm uses the concept of a stretched string constrained to lie in the cellular boundary of the digitized silhouette. Sloboda and Zat'ko (1995) presented a method of boundary approximation for finding the minimum-perimeter polygon in a given polygonally-defined annular region.

2.4 Contribution of Our Work

The difficulty, in finding the outer (inner) isothetic cover as a sequence of vertices (grid points), lies in the fact that during the traversal of an isothetic polygon, if the traversed path enters a complex region (background region in the case of an outer cover or object region in the case of an inner cover) for which a path of retreat from that region at a

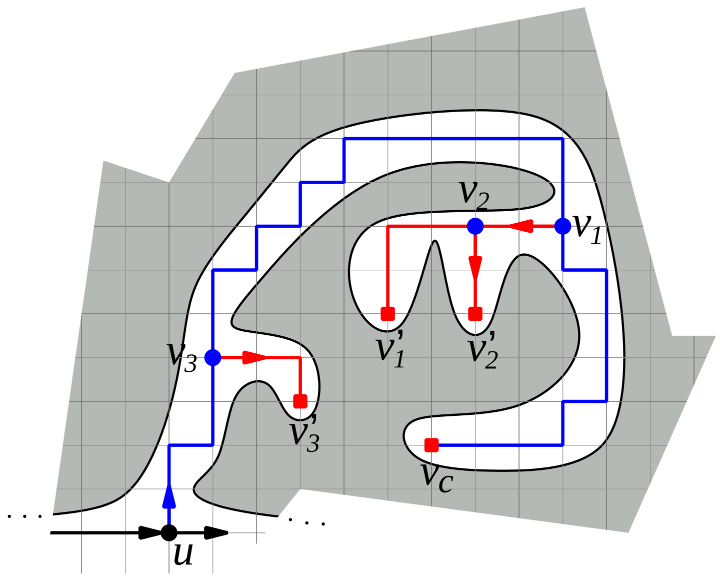


Figure 2.4: An example of backtracking in a complex region from the current vertex v_c lying in a *dead end* (object S shown in gray and background S' in white). In the proposed algorithm, no such backtracking is required. See text for details.

later stage is not possible, which marks a *dead end*, then a backtracking is required from an appropriate vertex lying on the traversed path. If the current vertex v_c lies in a *dead end*, then a vertex v has to be found on the path $v_s \rightsquigarrow v_c$, where v_s denotes the start vertex, by tracing back from v_c to v_s, along the path $v_c \rightsquigarrow v_s$. Such a vertex v should have an alternative path that would possibly come out of the complex region. However, the alternative path from v may again lead to a vertex v' lying in a dead end, which again results in backtracking the path $v' \rightsquigarrow v$, and still failing to produce a feasible path coming out of the corresponding region. This, in turn, calls for (recursively or iteratively) backtracking and searching another alternative path from some other vertex in $v \rightsquigarrow v_s$.

An instance where backtracking is required, has been illustrated in Fig. 2.4. The path (shown in blue) from u leads to the vertex v_c (current vertex) that lies in a dead end (shown as a red square). While backtracking from v_c, none of the nine grid points lying on the path $v_c \rightsquigarrow u$ provides any alternative path, which has to be verified for each of these points. The vertex v_1 is the first vertex found during backtracking, from which an alternative path is possible. While traversing along this alternative path from v_1, however, it is found that the path terminates at the vertex v'_1 that also marks a dead end. Next, backtracking is made along $v_1' \rightsquigarrow v_1$, and a vertex v_2 is found from which the alternative path again ends at the vertex v_2' lying in a dead end. No other vertex in the remaining part of $v_1' \rightsquigarrow v_1$ (i.e., $v_2 \rightsquigarrow v_1$) admits an alternative path, and therefore, again a new vertex is looked for by resuming the backtracking along $v_1 \rightsquigarrow u$. One such vertex is v_3, which again does not produce a successful solution. Finally, the backtracking ends at the vertex u, which has an alternative path lying outside the concerned region.

The challenge, therefore, lies in recognizing such a complex region against a background grid, which, if entered, would need backtracking. If the entry point of such a region can be recognized during the traversal of a polygon belonging to an isothetic cover, then instead of following the path entering the complex region, the alternative path along the grid lines can be chosen. Such a strategy would be entirely free of the backtracking process, and hence can construct the isothetic cover in a simpler and faster way.

We have proposed an algorithm for finding the outer and the inner isothetic covers of an arbitrarily-shaped digital object. This is a modification of the earlier crack following algorithm [Rosenfeld and Kak (1982)]. The proposed algorithm does not require any backtracking of the already traversed path. Also, the concept of background grid has been introduced whose resolution can be varied to enable fine or coarse analysis of the underlying shape. Two different approaches on finding the isothetic covers have been discussed in this chapter; one is for the uniform grid [Biswas *et al.* (2005b)] and the other for nonuniform grid [Bhowmick *et al.* (2005b)].

The strength of the proposed algorithm lies in the fact that it takes into account the combinatorial arrangement of the grid lines with respect to the object while simultaneously traversing and determining the boundary of the polygon defining an isothetic cover. The algorithm is completely devoid of any backtracking, and therefore, is fast and efficient. Avoidance of backtracking also makes the algorithm output-sensitive, in the sense that its time-complexity becomes linear in the length of the sum of the perimeters of the polygons constituting the isothetic cover (in grid units).

The chapter is organized as follows. The preliminaries, including the basic notions of digital geometry, necessary definitions, and review of earlier works have already been given in Sec. 2.2 and Sec. 2.3. The algorithm for constructing the outer isothetic cover (OIC) for a digital object consisting of a single connected component is described in Sec. 2.5. The generalized algorithm to construct the isothetic covers (both outer isothetic cover (OIC) and inner isothetic cover (IIC)) of an object with multiple connected components, is given in Sec. 2.6. The construction of isothetic covers of an object, when the underlying grid is non-uniform (not equi-spaced), is presented in Sec. 2.7. Section 2.8 contains the relevant experimental results on several databases and elaborates the scope of further applications using isothetic covers. Finally, in Sec. 2.9, we conclude the chapter with few open problems and with pointers to some possible future directions.

2.5 Outer Isothetic Cover of a Single Connected Component

The algorithm for constructing the outer isothetic cover (OIC), $\overline{P}(S)$, corresponding to an object S, which consists of only one connected component without holes, is stated here. The OIC of such an object consists of only one (outer) polygon. The generalized algorithm for an object having multiple connected components with or without holes is stated in Sec. 2.6. To construct $\overline{P}(S)$, we consider $\mathcal{I}$ to be a finite rectangular subset of $\mathbb{Z}^2$, which contains the entire object S. Let the height h and the width w of $\mathcal{I}$ be such that the grid size, g, divides both $h-1$ and $w-1$, and the boundary UGBs of $\mathcal{I}$ do not contain any part of S.

2.5.1 Combinatorial Classification of a Grid Point

An isothetic polygon has two types of vertices with internal angles 90^0 and 270^0($i.e.$, -90^0). The procedure for finding $\overline{P}(S)$ is based on classifying the grid points of $\mathcal{I}$ while constructing $\overline{P}(S)$. The type of any grid point, $q(i,j)$, is determined using the object occupancies of its neighboring UGBs and their combinatorial arrangements, which are obtained as follows.

Object Occupancy: UGB(i,j) is described by two horizontal (s_j^i and s_{j+g}^i) and two vertical (s_i^j and s_{i+g}^j) grid segments (Fig. 2.2). Since S is 8-connected and a grid segment is 4-connected, the intersection between them is well-defined [Rosenfeld and Kak (1982)]. Hence, S has an intersection with UGB(i,j), or UGB(i,j) is said to have an *object occupancy*, if and only if there exists a point p in $S \cap \left(s_j^i \cup s_{j+g}^i \cup s_i^j \cup s_{i+g}^j\right)$.

Hence, for each UGB, U_a $(a = 1,2,3,4)$, incident at $q(i,j)$, there are two possibilities: either $U_a \cap S \neq \emptyset$ or $U_a \cap S = \emptyset$, thus giving rise to $(2^4 =)$ sixteen possible arrangements of the four UGBs. These sixteen possible arrangements are further grouped into five combinatorial classes in our algorithm, where a particular class C_m includes all arrangements for which exactly $m(= 0,1,2,3,4)$ out of the four neighboring UGBs of q has/have object occupancy and the remaining $(4-q)$ ones do not have any object occupancy.

Combinatorial Arrangements of UGB*s*: Let $f(U_a)$ denote the object occupancy of U_a $(a = 1,2,3,4)$, defined as follows.

$$f(U_a) = \begin{cases} 0 & \text{if } U_a \cap S = \emptyset \\ 1 & \text{if } U_a \cap S \neq \emptyset \end{cases} \tag{2.1}$$

The number of UGBs having object occupancy is, therefore, given by $m = f(U_1) + \ldots + f(U_4)$. As a result, the grid point q will belong to one of the following five classes corresponding to five different values of m, which are also illustrated in Fig. 2.5.

1. C_0 $(m = 0)$: None of the four UGBs has object occupancy.

2. C_1 $(m = 1)$: Exactly one out of the four UGBs has object occupancy. Four such arrangements are possible depending on the UGB having occupancy.

3. C_2 $(m = 2)$: There are two arrangements for $m = 2$: UGBs having object occupancy (i) have a grid segment in common (i.e., adjacent); (ii) do not have a common grid segment (i.e., diagonally opposite).

   i) C_{2A} (Adjacent): U_a and U_b $(a, b \in \{1, 2, 3, 4\}, a \neq b)$ have object occupancy, such that $(a + b) \mod 2 = 1$. Four such arrangements are possible depending on the common grid segment.

   ii) C_{2B} (Diagonal): Either U_1 and U_3, or U_2 and U_4, have object occupancy.

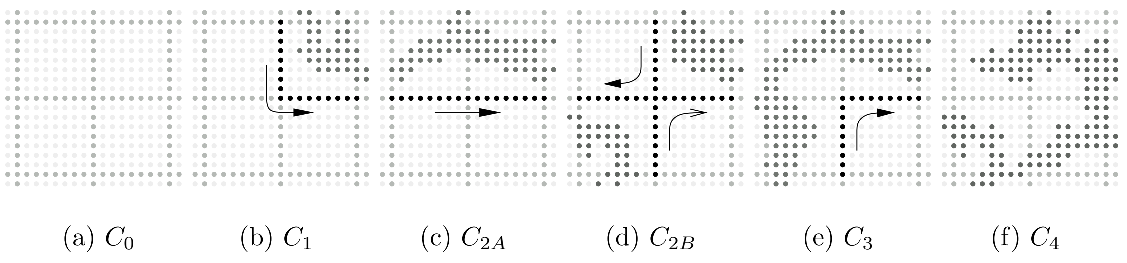
4. C_3 $(m = 3)$: Three neighboring UGBs have object occupancy. Four arrangements are possible depending on the UGB that has no object occupancy.

5. C_4 $(m = 4)$: All four neighboring UGBs have object occupancy.

The positions of a UGB, of a grid segment, and of a grid point, w.r.t. the OIC are determined using the following lemmas and theorems.

Lemma 1. *The interior of a* UGB *lies outside $\overline{P}(S)$ if and only if the* UGB *has no object occupancy.*

Proof. Let, for contradiction, there be a UGB, $U' \in \overline{P}(S)$, with no object occupancy. Then, either U' has at least one grid segment on (the boundary of) $\overline{P}(S)$, or U' has none of its four grid segments on $\overline{P}(S)$. If U' has one or more grid segments on the boundary of $\overline{P}(S)$, then the interior and the segments of U' lying on $\overline{P}(S)$ can be excluded from $\overline{P}(S)$ so that the segments of U' lying inside $\overline{P}(S)$ redefine the corresponding boundary of $\overline{P}(S)$. If U' has no segment on $\overline{P}(S)$, then only the interior of U' is excluded from $\overline{P}(S)$ and the four segments of U' add to the boundary of $\overline{P}(S)$. In either case, $\overline{P}(S)$ with a UGB



(a) C_0 (b) C_1 (c) C_{2A} (d) C_{2B} (e) C_3 (f) C_4

Figure 2.5: Vertex classification in an OIC: (a) exterior point, (b) 90^0 vertex, (c) edge point, (d) cross vertex, (e) 270^0 vertex, and (f) interior point.

having no object occupancy is not of minimum area, which contradicts the definition of the OIC (Definition 2.2.9).

Conversely, if U' lies outside $\overline{P}(S)$, then U' cannot have an object occupancy as no part of S lies outside $\overline{P}(S)$. □

Lemma 2. *A grid segment s belongs to the boundary of $\overline{P}(S)$ if and only if, out of the two* UGB*s having s in common, the interior of one lies outside and that of the other lies inside $\overline{P}(S)$.*

Proof. Let the two UGBs with s as the common grid segment, be U and U'. Hence, by Lemma 1, if the interior of U lies inside and that of U' outside $\overline{P}(S)$, then the segment s lies on $\overline{P}(S)$.

The converse is obvious; if s lies on the boundary, then the interior of one UGB is in $\overline{P}(S)$ and that of the other is not. □

Theorem 1 (90^o vertex)**.** *A grid point q is a 90^o vertex of $\overline{P}(S)$ if and only if q belongs to class C_1.*

Proof. W.l.o.g., let U_1 be occupied by the object S and the other three neighbor UGBs of $q(i,j)$ are not. As, of the two horizontally adjacent UGBs, U_1 and U_2, only U_1 has object occupancy, the grid segment s_i^j, common to U_1 and U_2, are on $\overline{P}(S)$ as stated in Lemma 2. Similarly, s_j^i is on the boundary of $\overline{P}(S)$, since between the two vertically adjacent segments, U_1 and U_4, only U_1 is occupied by S. The grid segment s_j^{i-g} is not on the boundary of $\overline{P}(S)$ as none of its adjacent UGBs, U_2 and U_3, has object occupancy. Same is true for s_i^{j-g}. Hence, s_i^j and s_j^i are two grid segments on $\overline{P}(S)$ incident at q, thereby making it a 90^0 vertex.

Conversely, if q is a 90^0 vertex, then exactly two mutually perpendicular grid segments, say s_i^j and s_j^i, of the four incident grid segments at q, are on $\overline{P}(S)$. As s_j^{i-g} and s_i^{j-g} are not on $\overline{P}(S)$, it can be shown that only U_1 has object occupancy. This is true for other three combinations for q being a 90^0 vertex. Thus, if q is a 90^0 vertex, then it belongs to class C_1. □

Theorem 2 (270^o vertex)**.** *A grid point q is a 270^o vertex of $\overline{P}(S)$ if and only if q belongs to class C_3 or class C_{2B}.*

Proof. If q belongs to C_3, then exactly three of the UGBs have object occupancy. Let, w.l.o.g., U_1 has no object occupancy. Then, s_i^j $(= U_1 \cap U_2)$ and s_j^i $(= U_1 \cap U_4)$ are on the boundary of $\overline{P}(S)$ by Lemma 2. The included angle at q being 270^0, q is a 270^0 vertex.

If q belongs to C_{2B}, then two diagonal UGBs are occupied by S. By Lemma 2, all the four grid segments incident at q lie on $\overline{P}(S)$. Hence, q occurs twice in $\overline{P}(S)$. However, as each polygon in $\overline{P}(S)$ is simple, q cannot occur twice in a particular polygon of $\overline{P}(S)$. Thus, q occurs in two polygons, once for each. Further, since there is exactly one outer polygon in $\overline{P}(S)$ (S being a single component, with or without holes), q lies on at least one hole polygon. As shown in Fig. 2.5(d), if the segments s_i^{j-g} and s_j^{i-g} lie on the same polygon, and S lies left of the polygon during its traversal, then, while taking a left turn at q (traversing s_i^{j-g} first and s_j^{i-g} next), the part of S in U_3 lies left but that in U_1 lies right, which is a contradiction. If there is a right turn at q by traversing s_i^{j-g} first and s_j^i next, then S lies left, both in U_1 and U_3. This implies that q is a 270^0 vertex for one polygon. Clearly, for the other polygon, q will be again a 270^0 vertex for similar reasons.

Conversely, if q is a 270^0 vertex, then one UGB, say U_1, has no object occupancy and both of its adjacent UGBs (U_2 and U_4) have object occupancy. If U_3 has object occupancy, then q belongs to C_3, and if U_3 has no object occupancy, then q belongs to C_{2B} by Lemma 2. □

Theorem 3 (edge point)**.** *A grid point q is a non-vertex edge point of $\overline{P}(S)$ if and only if q belongs to class C_{2A}.*

Proof. If q belongs to class C_{2A}, then two adjacent UGBs, say U_1 and U_2, are occupied by the object and the other two are not. Thus, by Lemma 2, s_j^{i-g} and s_j^i are on the boundary of $\overline{P}(S)$. As both the grid segments are horizontal and $s_j^{i-g} \cap s_j^i = q$, q is a non-vertex edge point on the boundary of $\overline{P}(S)$.

If q is an edge point, then both the segments incident at q, which are part of $\overline{P}(S)$, are either horizontal or vertical. Thus, only two adjacent UGBs have object occupancy.

Hence, by Lemma 2, when q is an edge point, it belongs to class C_{2A}. □

If q belongs to class C_{2B}, then q occurs twice as a vertex in the OIC, which is referred as a *cross vertex* in Fig. 2.5(d). If a grid point q is neither a vertex nor an edge point of $\overline{P}(S)$, then it lies either inside or outside $\overline{P}(S)$. The classification of such a grid point w.r.t. $\overline{P}(S)$ is done using the following theorems.

Theorem 4 (exterior point)**.** *A grid point q lies outside $\overline{P}(S)$ if and only if q belongs to class C_0.*

Proof. If q belongs to class C_0, then no U_a $(a = 1, 2, 3, 4)$, incident at q, has object occupancy, and hence, by Lemma 2, none of the four grid segments incident at q lies on $\overline{P}(S)$. As a result, the intersection of these four grid segments, which is q, does not belong to $\overline{P}(S)$.

If q lies outside $\overline{P}(S)$, then none of the four grid segments incident at q is on the boundary of $\overline{P}(S)$. Hence, none of the neighboring UGBs has object occupancy, which means q belongs to class C_0. □

Theorem 5 (interior point)**.** *A grid point q is in the interior of $\overline{P}(S)$ if and only if q belongs to class C_4.*

Proof. When q belongs to class C_4, all four UGBs incident at q have object occupancy. None of the four grid segments incident at q qualifies for being on the boundary of $\overline{P}(S)$ by Lemma 1 and Lemma 2. Thus, q is an interior point of $\overline{P}(S)$.

Conversely, when the grid point q is in the interior of $\overline{P}(S)$, none of the grid segments incident at q is on the boundary of $\overline{P}(S)$. Hence, all the corresponding UGBs have object occupancy, which means, q belongs to class C_4. □

It may be mentioned here that, if S is a subset of the interior of a UGB, which may arise if S is sufficiently small relative to the grid size, then no grid segment intersects S. Such a degenerate case, where all grid points are classified to C_0, can be handled by determining the object occupancy based on the intersection of S with the interior of a UGB.

2.5.2 Construction of $\overline{P}(S)$

Here we assume that the object S consists of one connected component without any hole, and its top-left point, $p_0(i_0, j_0)$, is given. The point p_0 is the top-left point of S if and only if, for each other point $(i, j) \in S$, either $j < j_0$, or $j = j_0$ and $i > i_0$. The *start point* of traversal, q_s, is first determined from p_0. Then the construction starts from q_s and ends when the traversal reaches q_s (Theorem 7). The traversal is guided by the type of each grid point q, which is obtained from the corresponding class of q. The type t of q corresponding to its internal angle 90^0, 180^0, and 270^0, w.r.t. an isothetic polygon is considered to be $\mathbf{1}$, $\mathbf{0}$, and $\mathbf{-1}$ respectively. The algorithm outputs $\overline{P}(S)$ (here, one outer polygon) as an ordered set of vertices. The information stored corresponding to each vertex are its coordinates and type. In the degenerate case in which S lies in the interior of a single UGB, the $\overline{P}(S)$ is empty.

Determination of Start Point: If p_0 lies in the interior of a UGB (w.l.o.g, say U_1), then the top-left grid point of U_1 is considered as the start point, q_s. If p_0 coincides with a grid point, then the top-left grid point of U_2 (incident at p_0, Fig. 2.2 (a)) is considered as q_s. Otherwise, p_0 lies on a horizontal (vertical) grid segment common to two adjacent UGBs, say U_1 and U_4; then the top-left grid point of U_1 is considered as q_s. The coordinates of q_s, therefore, are given by

$$i_s = (\lceil i_0/g \rceil - 1) \times g, \; j_s = (\lfloor j_0/g \rfloor + 1) \times g. \tag{2.2}$$

The intersections of the neighboring UGBs of q_s with S are computed and the vertex type of q_s is determined (Theorems 1, 2, 3). Let d denote the direction to be followed from a vertex or an edge point q while constructing an isothetic polygon of $\overline{P}(S)$. The different values of d, namely $0, 1, 2,$ and 3, denote the directions towards right, top, left, and down, respectively. The starting direction from q_s is given by $(2 + t_s) \mod 4$. By Eqn. 2.2, neither U_1 nor U_2 (of q_s), and no UGB above U_1, can have object occupancy. However, U_3 may or may not have an object occupancy, thereby giving rise to two distinct cases: q_s is a 90^0 vertex when only U_4 is occupied; q_s is an edge point when both U_3 and U_4 are occupied. We get the following lemma.

Lemma 3. *The start point is either a 90^0 vertex or an edge point.*

Tracing the Polygon: During the traversal of (the boundary of the polygon in) $\overline{P}(S)$, the direction d_c from the current grid point q_c, and the coordinates (i_c, j_c) of q_c, being known, we compute the coordinates of the next grid point, $q_n := (i_n, j_n)$, lying on $\overline{P}(S)$, as follows.

$$\begin{array}{rl|rl|rl|rl} d_c = 0: & i_n = i_c + g & d_c = 1: & i_n = i_c & d_c = 2: & i_n = i_c - g & d_c = 3: & i_n = i_c \\ & j_n = j_c & & j_n = j_c + g & & j_n = j_c & & j_n = j_c - g \end{array}$$

which is expressed in a simpler way to

$$(i_n, j_n) = d_c \circledast (i_c, j_c). \tag{2.3}$$

The vertex type t_n of q_n is computed by determining the object occupancy of its neighboring UGBs. The direction of traversal, d_n, from q_n is given by

$$d_n = (d_c + t_n) \mod 4 \tag{2.4}$$

Iteratively, the details of each grid point lying on $\overline{P}(S)$ are computed until q_n coincides with q_s. If the grid point is a vertex, then it is added to the sequence of vertices comprising $\overline{P}(S)$. The following theorems justify the correctness of $\overline{P}(S)$ (Definition 2.2.9) and how the traversal converges to obtain the sequence of its vertices.

Theorem 6. *If p is a point lying on $\overline{P}(S)$, then $0 < d_\top(p, S) \leq g$.*

Proof. Since no point on $\overline{P}(S)$ is in S, $d_\top(p, S) > 0$. To show that $d_\top(p, S) > g$, let, w.l.o.g., p be any point on the horizontal grid segment s_j^i that lies on $\overline{P}(S)$. Let $d_\top(p, S) = h > g$, if possible. For any point q, belonging to U_1 or U_4, $d_\top(p, q)$ is less than or equal to g. Since $h > g$, neither U_1 nor U_4 has any object occupancy, wherefore their interiors are outside $\overline{P}(S)$ (Lemma 1). Hence, s_j^i does not lie on $\overline{P}(S)$ (Lemma 2), which means p cannot be on $\overline{P}(S)$ — a contradiction. □

Theorem 7. *The construction of $\overline{P}(S)$ concludes at the start vertex.*

Proof. Excepting the degenerate case where S is a subset of the interior of a UGB, S intersects a subset of UGBs comprising $\mathcal{I}$. Note that in the degenerate case where S is a subset of the interior of a UGB, S still intersects a UGB but it does not intersect the border of that UGB and the $\overline{P}(S)$ is empty. The construction of $\overline{P}(S)$ starts from a 90^0 vertex or an edge point, q_s (Lemma 3), whose distance from S is at most g (Theorem 6). The construction process continues along those grid segments such that all points belonging to such a grid segment have the maximum distance of g from S (Theorem 6). If q_s is a 90^0 vertex (edge point), then the downward (leftward) grid segment from q_s is traversed at the first step, and its other grid segment on $\overline{P}(S)$ remains untraversed. Each other vertex or edge point of $\overline{P}(S)$ also has exactly two grid segments lying on $\overline{P}(S)$, which would be traversed in two consecutive steps during the construction. Hence, the untraversed grid segment incident at q_s is finally traversed, which concludes the construction of $\overline{P}(S)$. □

2.5.3 Algorithm Make-OIP

The algorithm Make-OIP that constructs the outer polygon P, corresponding to an object S consisting of a single connected component, is shown in Fig. 2.6. The start

```
Algorithm Make-OIP $(S, p_0)$

Steps:
1. $L \leftarrow \emptyset$
2. $q_s \leftarrow$ Start $(S, p_0)$
3. $q \leftarrow q_s$
4. $d \leftarrow (2 + t_s) \mod 4$
5. do
6.     if $t \in \{1, -1\}$ then $L \leftarrow L \cup \{q\}$
7.     $q \leftarrow d \circledast q$ ▷ Eqn. 2.3
8.     $t \leftarrow$ VType$(S, q)$
9.     if $t = -2$ then  $t \leftarrow -1$
10.    $d \leftarrow (d + t) \mod 4$
11. while $(q \neq q_s)$
12. return $L$
```

```
Procedure Start $(S, p_0)$

Steps:
1. $i_s \leftarrow (\lceil i_0/g \rceil + 1) \times g$,
   $j_s \leftarrow (\lfloor j_0/g \rfloor - 1) \times g$
2. $t_s \leftarrow$ VType$(S, q_s)$
3. return $q_s$ ▷ $(i_s, j_s, t_s)$
```

```
Procedure VType $(S, q)$

Steps:
1. $m \leftarrow 0, r \leftarrow 0$
2. for $k \leftarrow 1$ to 4
3.     if $U_i^q \cap S \neq \emptyset$
4.         $m \leftarrow m + 1$, $r \leftarrow r + k$
5. if $r \in \{4, 6\}$ and $m = 2$ then $t \leftarrow -2$
6. else if $m \in \{0, 4\}$ then $t \leftarrow 0$
7. else $t \leftarrow 2 - m$
8. return $t$
```

Figure 2.6: The algorithm Make-OIP and the related procedures to construct $\overline{P}(S)$ for an object S having a single connected component.

point, q_s, is determined by the procedure Start using the top-left point, p_0, as input. The subsequent vertices and edge points of P are visited one by one in the **do-while** loop (Steps 5–11) of the algorithm Make-OIP. If $|P|$ denotes the length of the perimeter of P, then the number of grid points (i.e., vertices and edge points) lying on the perimeter of P is $|P|/g$, when the grid size is g. In each iteration (**do-while** loop), the type of a grid point q lying on P is checked and added to the list of vertices, L, if q is a vertex of P. The coordinates and the type of q lying on P are computed in each iteration. The procedure VType finds the type of q using the four UGBs incident at q. For each UGB, each point on each of its four defining grid segments is considered one by one to check whether it belongs to S, which needs $O(1)$ time in the best case and $O(g)$ time in the worst case. Thus, in the worst case, each iteration takes $O(g)$ time, finding the start point q_s takes $O(g)$ time, and all other steps of the algorithm Make-OIP take $\Theta(1)$ time in total. Hence, the worst-case time-complexity of the algorithm Make-OIP for an object S with a single connected component without any hole, is given by $(|P|/g) \times O(g) + O(g) + \Theta(1)$, which simplifies to $(|P|/g) \times O(g) = O(|P|)$. The best-case time-complexity, found in a similar way, is $(|P|/g) \times O(1) + O(1) + \Theta(1) = O(|P|/g)$. The algorithm is, therefore, linear on the perimeter of the outer polygon, and hence output-sensitive. It may be noted here that the crack following algorithm of Rosenfeld and Kak (1982) is also linear on the perimeter, however the outputs are different (as explained in details in Sec. 2.3).

2.6 Generalized Algorithms for Constructing Isothetic Covers

2.6.1 Outer Isothetic Cover

The algorithm MAKE-OIC to construct the OIC of an object S, having multiple components with or without holes, is given in Fig. 2.7. Each grid point, $q \in \mathcal{I}$, initialized as unvisited (Steps 2–4), is considered in row-major order (Steps 5–11). If q is not already visited (Step 6) and qualifies as a vertex (Step 7 and Step 9), then the construction of a new polygon of $\overline{P}(S)$ starts from q as the start vertex (Step 8 and Step 10). The procedure MAKE-IP traces an outer polygon if q is a 90^0 vertex, and a hole polygon if it is a 270^0 vertex. The procedure MAKE-IP is similar in nature as the algorithm presented in Fig. 2.6, except that the grid points traversed in course of constructing the polygon are marked as "visited" in Step 4 of MAKE-IP. In Step 9 of MAKE-IP, the start vertex is added to the ordered list, L, to mark the end of one polygon. The vertices of the subsequent polygons, if any, are appended to L starting at (and ending with) the corresponding start vertices, so that the final form of OIC is given by $L = \{q_{11}, q_{21}, \ldots, q_{11}, q_{21}, q_{22}, \ldots, q_{21}, \ldots\}$, where, q_{i1} is the start vertex and q_{ij} is the jth vertex of the ith polygon, P_i.

Time Complexity: The basic operation used in the algorithm is the checking of each grid point of $\mathcal{I}$, in row-major order, for its possible candidature of being a vertex or an edge point or none, which needs $O(g)$ time in the worst case. This is done exactly once for each grid point, as already-traversed grid points are marked as "visited". The number of grid points in $\mathcal{I}$, excepting the boundary grid points (since they will not lie on the isothetic cover as per our consideration of $\mathcal{I}$), is $((h-1)/g-1) \times ((w-1)/g-1) = \Theta(hw/g^2)$. Hence, the worst-case time-complexity of the algorithm MAKE-OIC is $\Theta(hw/g^2) \cdot O(g) = O(hw/g)$.

2.6.2 Inner Isothetic Cover

The inner cover, $\underline{P}(S)$, can be constructed in a manner exactly converse to that of $\overline{P}(S)$. A grid point $q \in S$ is classified as a vertex of $\underline{P}(S)$, depending on the intersections of the

```
Algorithm MAKE-OIC ($S$)

Steps:
1.  $L \leftarrow \emptyset$
2.  for each grid point $q \in \mathfrak{I}$
        $\triangleright$ in row-major order
3.      $visited[q] \leftarrow$ FALSE
4.      $t \leftarrow$ VTYPE $(S, q)$
5.  for each grid point $q \in \mathfrak{I}$
6.      if $visited[q] =$ FALSE
7.          if $t = 1$
8.              MAKE-IP $(S, q, 3)$
9.          else if $t = -1$
10.             MAKE-IP $(S, q, 0)$
11. return $L$
```

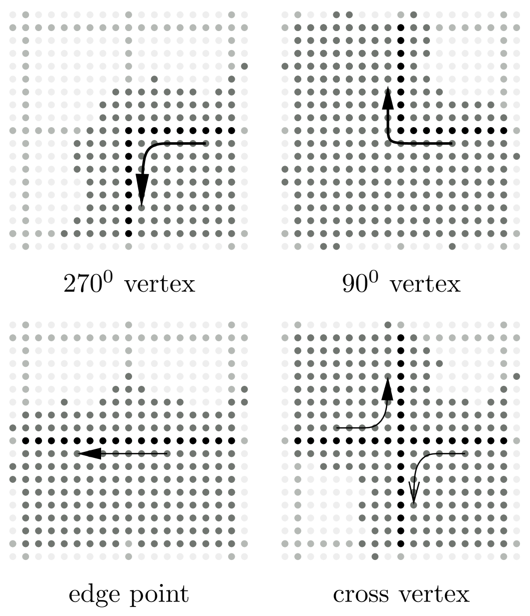
```
Procedure MAKE-IP ($S, q, d$)

Steps:
1.  $q_s \leftarrow q$
2.  do
3.      if $t = -2$ then $t = -1$
4.      else $visited[q] \leftarrow$ TRUE
5.      if $t \in \{1, -1\}$ then $L \leftarrow L \cup \{q\}$
6.      $q \leftarrow d \circledast q$ $\triangleright$ Eqn. 2.3
7.      $d \leftarrow (d + t) \mod 4$
8.  while $(q \neq q_s)$
9.  $L \leftarrow L \cup \{q_s\}$
10. return
```

Figure 2.7: The generalized algorithm MAKE-OIC and the procedures to construct $\overline{P}(S)$ for an object S with multiple connected components (possibly having holes).

UGBs (incident at q) with $S' := \mathfrak{I} \smallsetminus S$. If m $(0 \leqslant m \leqslant 4)$ denotes the total number of UGBs intersected by S', then q belongs to class C'_m. Class C'_1 and class C'_3 correspond to 90^0 and 270^0 vertices, respectively. For class C'_2, if the UGBs intersected by S' has a common grid segment, then q is an edge point; otherwise, q is a 270^0 vertex. Points in C'_0 and in C'_4 lie respectively inside and outside $\underline{P}(S)$.

Figure 2.8 states the algorithm MAKE-IIC for constructing the inner cover of a general object S, having one or more components with or without holes. The algorithm is similar to MAKE-OIC, but it takes the background, S', as the input. The construction of a polygon of $\underline{P}(S)$ starts from a 90^0 vertex, q_s, keeping S' left during the traversal. The polygon is traced to the next grid point, q_n. The type of q_n is decided and the direction of traversal from q_n is computed. The traversal is continued until q_s is reached. As the basic steps of the algorithm MAKE-IIC are similar to those of MAKE-OIC, the time complexity of the algorithm is same as that of MAKE-OIC.



```
Algorithm MAKE-IIC ($S', g$)

Steps:
1. $L \leftarrow \emptyset$
2. for each grid point $q \in \mathcal{I}$
       $\triangleright$ in row-major order
3.     $visited[q] \leftarrow$ FALSE
4.     $t \leftarrow$ VTYPE ($S', q$)
5. for each grid point $q \in \mathcal{I}$
6.         if $visited[q] =$ FALSE
7.             if $t = 1$
8.                 MAKE-IP ($S', q, 0$)
9.             else if $t = -1$
10.                MAKE-IP ($S', q, 3$)
11. return $L$
```

Figure 2.8: Vertex classification and the algorithm to construct the IIC.

2.7 Isothetic Covers on a Non-uniform 2D Grid

The construction of isothetic covers on a non-uniform grid is based on the same basic principle as discussed earlier for uniform grid. In non-uniform grid, the grid size g, spacing between two consecutive horizontal/vertical grid lines may not be equal as shown in Fig. 2.9. The classification of a grid point q is done in the same manner by determining the object occupancy of the four incident UGBs which are probably of different dimensions (height and width). The algorithms used to construct the OICs and IICs of a digital object on uniform grid can be used on non-uniform grid. However, it should be noted, while applying these algorithms, that g is variable and each UGB can have two different values of g (one in horizontal direction and another in vertical direction).

2.8 Experimental Results

We have implemented the proposed algorithm in C in SunOS Release 5.7 Generic of Sun Ultra 5_10, Sparc, 233 MHz. The algorithm is run on different sets of binary images, such as i) geometric figures, ii) logo images, iii) object-type images, iv) optical and handwritten characters, and v) scanned document images. The results related to non-uniform grid are

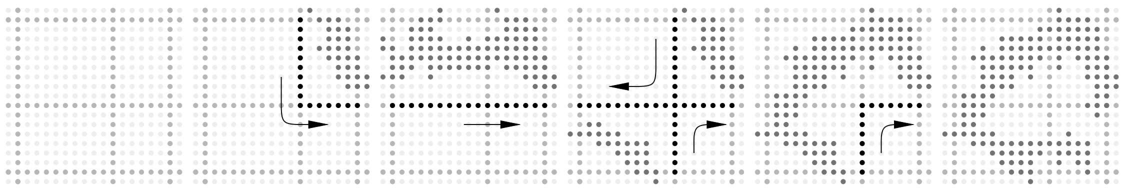

(a) C_0 (b) C_1 (c) C_{2A} (d) C_{2B} (e) C_3 (f) C_4

Figure 2.9: Vertex classification in an OIC on non-uniform grid: (a) exterior point, (b) 90^0 vertex, (c) edge point, (d) cross vertex, (e) 270^0 vertex, and (f) interior point.

also provided in this section. The results and related findings are discussed in the following sections.

2.8.1 Geometric Figures

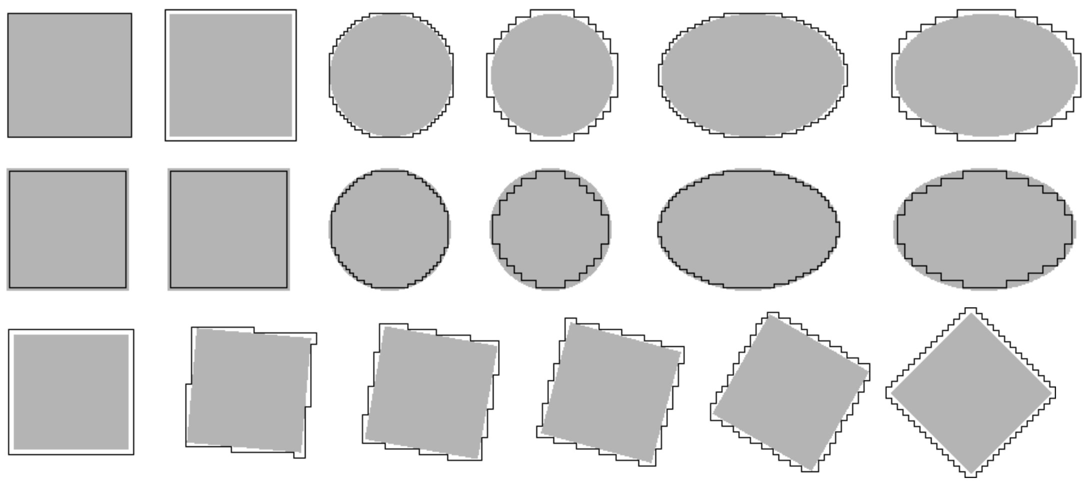


Figure 2.10: The first row shows the OICs and the second row shows the IICs of different geometric shapes for $g = 3$ (odd columns) and $g = 6$ (even columns). The third row shows the OICs of a square, rotated at the angles 5^0, 10^0, 15^0, 30^0, and 45^0.

The isothetic covers for $g = 3$ and $g = 6$ corresponding to a few simple geometric shapes are shown in Fig. 2.10. For a square-shaped object, as shown in this figure, the number of vertices is four for $g = 3$ and $g = 6$. This is in conformity with the fact that the bounding shapes (or complexities) of the OIC and the IIC of an axis-parallel square should not alter with a change in the grid size. However, if the square is tilted, then the

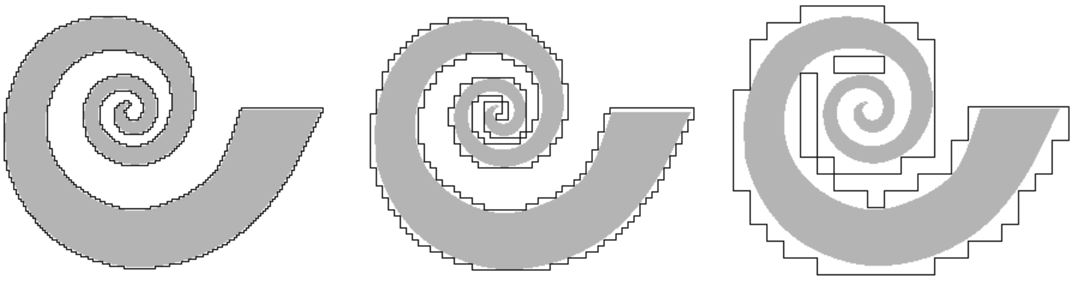


Figure 2.11: The OICs of a spiral at different grid sizes: g = 2, 5, and 14.

complexity (i.e., the number of vertices) of the OIC and that of the IIC increases with a decrease of the grid size. Further, in the vertex sequence of the OIC (IIC) of a tilted square, as shown in Fig. 2.10, each vertex of type **1** is followed by a vertex of either type **1** or type $-\mathbf{1}$, and each vertex of type $-\mathbf{1}$ by a vertex of type **1**. Each pattern $\mathbf{1}(-\mathbf{1})$ contains a horizontal/vertical segment of length kg or $(k+1)g$, where $k > 1$, and each pattern $(-\mathbf{1})\mathbf{1}$ contains a vertical/horizontal segment of length g. Such a sequence satisfies the properties of digital straightness [Freeman (1961b), Klette and Rosenfeld (2004a)], indicating that the longest subsequence of the OIC having the form $\mathbf{1}((-\mathbf{1})\mathbf{1})^*\mathbf{1}$ corresponds to a straight part of the underlying shape. Two consecutive vertices with type **1** imply the start of a new straight edge, thereby corresponding to a 90^0 corner of a square.

For a disc, the complexities of both the OIC and the IIC decrease with an increase of the grid size and vice versa, as shown in Fig. 2.10. The symmetry owing to the circular shape of a disc is also captured in the outer and the inner covers. The symmetry, however, is only present in each of the two covers only about the vertical, horizontal, and two diagonal lines (45^0 and 135^0) passing through the center of the corresponding cover due to the anisotropic nature of an orthogonal grid. Similarly, for an ellipse, the corresponding OIC and IIC also possess the desired horizontal and vertical symmetries, but not the diagonal symmetries.

As a more complicated geometric shape, we have considered a spiral and have shown its OICs in Fig. 2.11. A series of the pattern $(\mathbf{1}(-\mathbf{1}))^*(-\mathbf{1})(-\mathbf{1})$ occurs while the OIC traverses inwards or "spirals in". The spiral unwinds with a consecutive occurrence of the pattern $(\mathbf{11})(\mathbf{1}(-\mathbf{1}))^*$. The pattern $(-\mathbf{1})(-\mathbf{1})$ indicates a concave region and the pattern **11** indicates a convex region. The asymmetry of a spiral is also captured in its OIC and IIC. The complexity of the cover decreases with an increase of the grid size. When the grid size is sufficiently high, the OIC may not capture the concavity of the spiral as the

Table 2.1: Numbers of vertices (n), perimeters (s), and areas (a) of the isothetic covers, and the respective CPU times (T, in milliseconds) corresponding to different grid sizes.

image	image size (object size)	g	OIC				IIC			
			n	s	a	T	n	s	a	T
logo	200 × 200	4	100	886	8971	6	104	808	5205	2
	(6588)	8	40	704	10589	6	46	684	3481	2
		16	14	512	14337	4	12	192	867	2
leaf	296 × 412	4	346	2400	49055	14	322	2120	38844	17
	(42835)	8	168	2301	54140	14	148	1936	35084	8
		16	63	1855	61855	8	70	1696	26708	8
spiral	442 × 442	4	300	2096	28777	48	286	2021	20437	20
	(23776)	8	158	2032	33009	38	116	1584	16219	12
		16	60	1472	41437	25	42	1184	10322	9
myth	420 × 710	4	468	4304	55409	69	464	3792	34194	66
	(42773)	8	208	3504	63634	49	206	3005	27178	45
		16	82	2592	77580	38	66	1600	16168	35

width of the concavity is less than that of the grid size.

Table 2.1 shows the number of vertices and the perimeter of OIC, and the CPU time required for the computation of an OIC on different grid sizes for several images. As g increases, the number of vertices of an OIC decreases. The CPU time also decreases with an increase of the grid size. Figure 2.12 shows how the number of vertices and the perimeter of an OIC vary with increasing grid size. It may be noted here that the "myth" image referred here has been shown later in Fig. 2.26. Figure 2.13 shows that the CPU time decreases as g increases. Figure 2.14 shows how the number of vertices, n, of the OIC of a given object decreases with the increase of g. The change in number of vertices n of the OIC, corresponding to $g = 4$, on rotating a given object by an angle θ, from 1^0 to 90^0, is plotted in Fig. 2.14. The plot shows that the pattern of variation of n is repeated every 45^0, the pattern is regular for a symmetric object (e.g., square), and it varies for more complex, irregular objects like logo and spiral. In Fig. 2.15, we have furnished two 3D plots of the error frequency versus g and $d_\top$ corresponding to the images "spiral" and "myth". For a grid size g, the error frequency, $f(g, \delta)$, is given by the number of points on $\overline{P}(S)$ for which the nearest object points are at a distance δ (measured using $d_\top$).

2.8.2 Logo and Object-type Images

The OICs of four logo images for $g = 8$ and $g = 16$ are shown in Fig. 2.16. The OIC can be coded using the vertex types, $\mathbf{1}$ and $\mathbf{-1}$, and the edge lengths, to generate a

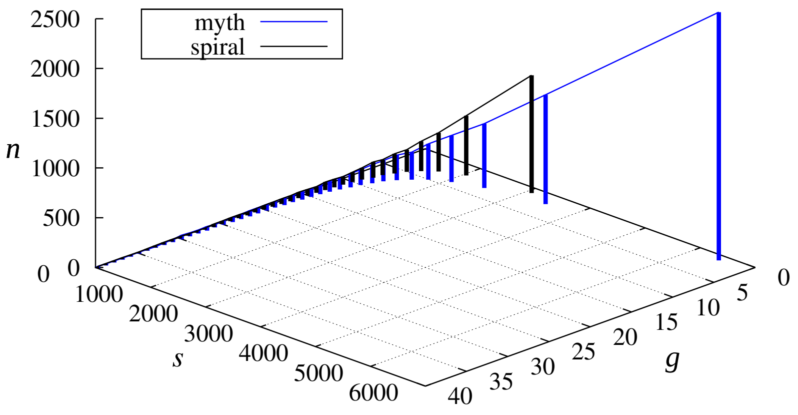


Figure 2.12: Plot on the number of vertices (n) and the perimeter (s) of the OICs versus the grid size (g) corresponding to the images “spiral” and “myth”.

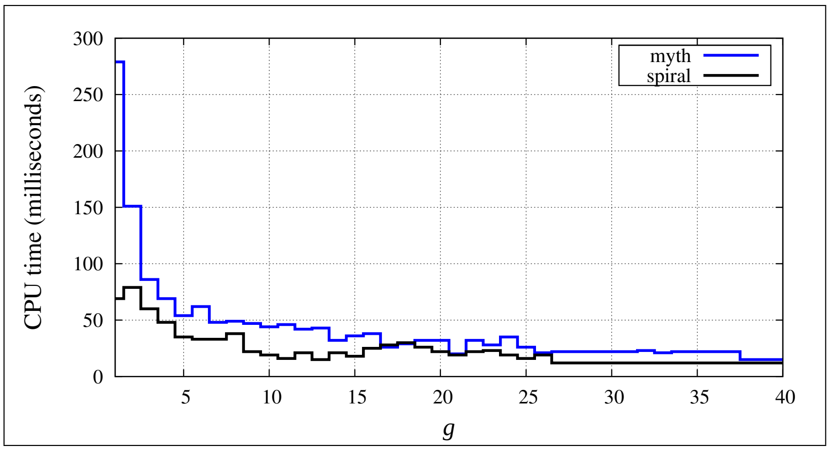


Figure 2.13: Plot on CPU time (in milliseconds) for construction of the OICs versus the grid size (g) corresponding to the images “spiral” and “myth”.

shape code that describes the OIC, and hence to capture the structural information about the underlying object. Such shape codes can be generated for appropriate grid sizes to derive a multi-resolution feature-vector. These feature-vectors can be used to compute

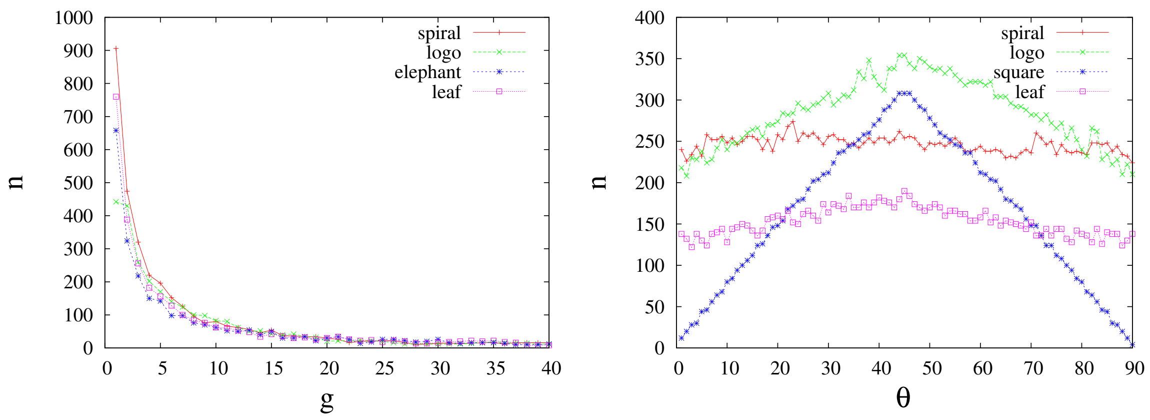


Figure 2.14: Plot on the number of vertices, n, versus g, and plot on n versus θ, where θ is the angle of rotation for $g = 4$.

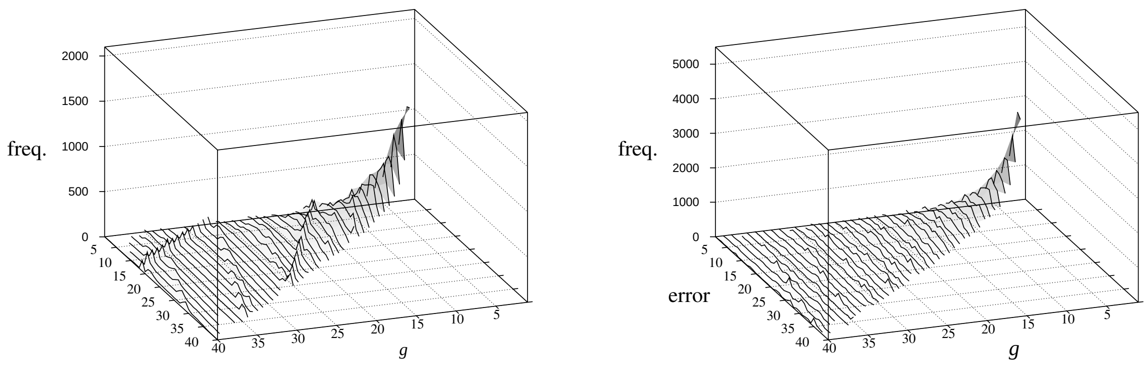


Figure 2.15: Frequency of errors ($f(g, d_{\top})$) plotted against the isothetic error ($d_{\top}$) and the grid size (g in $[1, 40]$), corresponding to the images "spiral" (left) and "myth" (right).

the Hamming distance between two or more objects as a measure of similarity for object indexing and retrieval [Biswas *et al.* (2005a)].

In Fig. 2.17, the OICs of four animal images, and in Fig 2.18, the OICs of two different types of leaves, are shown for $g = 8$ and $g = 16$. Like logo images, the OICs capture the shape of the animal images in a similar way. Usually, the OIC consists of a single polygon. In some cases, however, the OIC also contains a few hole polygons or pseudo-hole polygons. For example, the OIC of the image "elephant" has one hole polygon and

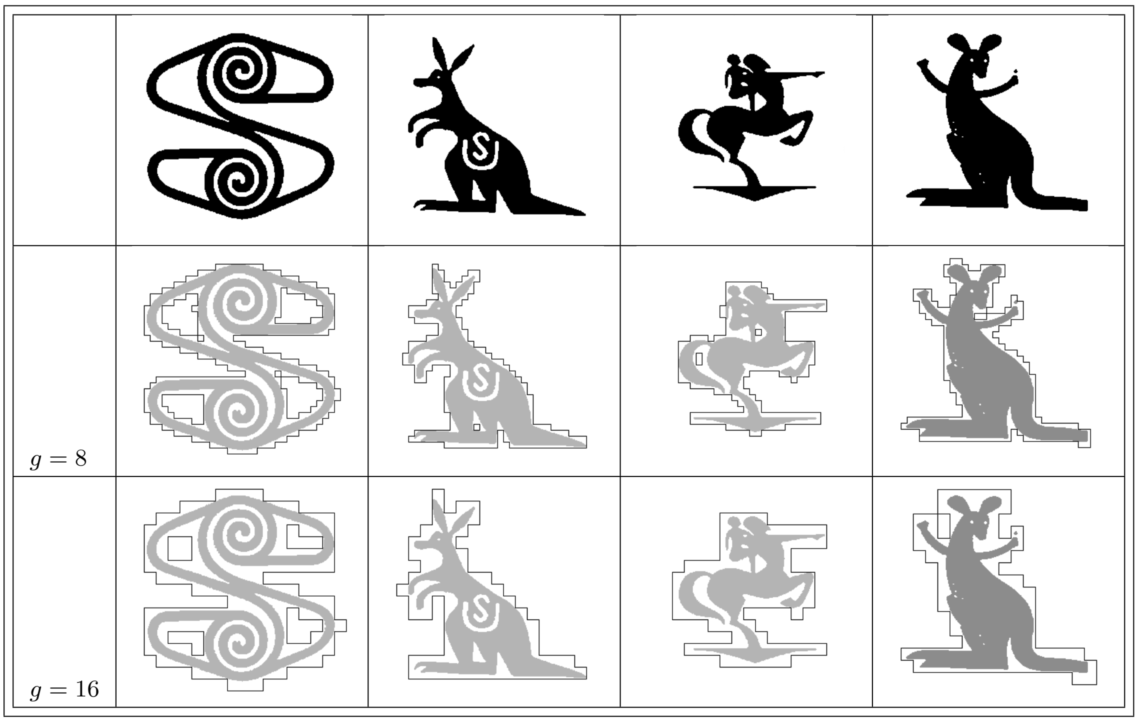


Figure 2.16: The outer isothetic covers (OIC) for a set of logo images.

one pseudo-hole polygon for $g = 8$, and has two pseudo-hole polygons for $g = 16$.

The isothetic covers of a digital object can be used for many practical applications. For example, the shape complexity of a digital object can be computed by chain-code encoding [Freeman (1961a)] of the OIC and reducing the chain code using certain reduction rules (chapter 4 of this thesis [Biswas *et al.* (2007b)]). Another example is the polygonal approximation of thick and rough digital curves, which can be derived using their isothetic covers (chapter 5 of this thesis [Bhowmick *et al.* (2006)]). An example is shown in Fig. 2.19, where a polygonal approximation of a digital curve representing the map of India is shown. The outer polygon and the hole polygon corresponding to this curve approximates the map of India. Similarly, a real-world object, when digitized, may possess disconnectedness, noisy information, etc. The existing algorithms on component labeling are liable to produce components that are larger in number than sought for. Whereas, using the isothetic covers, two or more components that are spatially not far off, may be reported as a single loosely connected component [Bhowmick *et al.* (2007a)]. Figure 2.20 shows how we can get a single loosely connected components corresponding to "a flock of birds" for $g = 16$.

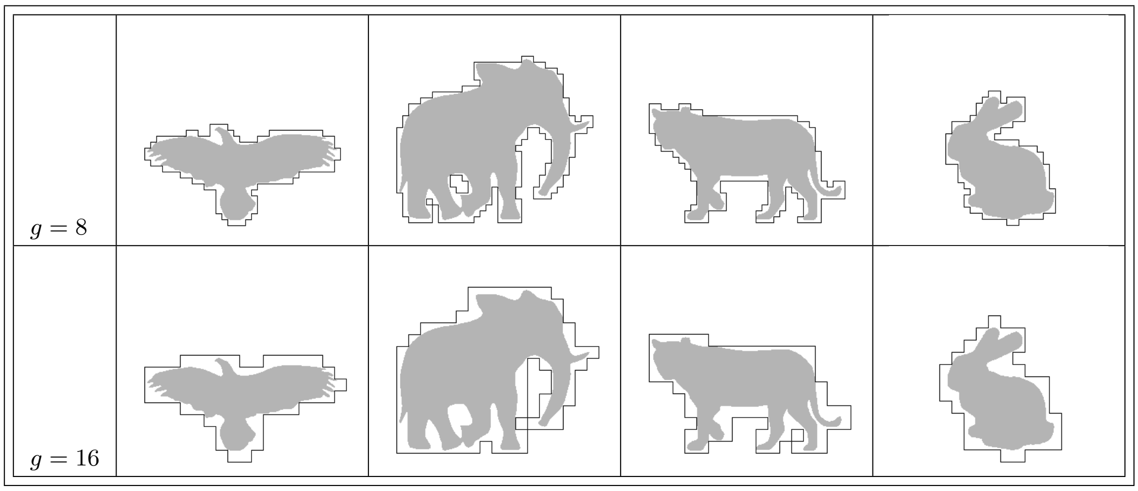


Figure 2.17: The outer isothetic covers (OIC) for a set of animal images.

2.8.3 Optical and Handwritten Characters

Figure 2.21 shows the OICs corresponding to a set of optical and handwritten Bengali characters and numerals. It is evident from Fig. 2.21 that the OIC corresponding to an optical character has a regular and definite shape. However, the OIC corresponding to the same character in the handwritten set is not as definite and regular as the optical one. For example, a straight segment (horizontal, vertical, or inclined) of the Bengali character "kaw" is being conformed by the corresponding vertex pattern of its OIC in a more prominent way in the optical set than in the handwritten set. It is also interesting to note that the OIC of the handwritten "kaw" consists of a hole polygon inside the main polygon — a fact supporting the results corresponding to the optical "kaw". The optical "kaw" has a pseudo-hole polygon in excess, which is, however, much smaller in size, and hence might play a negligible role in the similarity measure.

Similar results on few English characters and numerals are presented in Fig. 2.22, and they demonstrate how the OICs capture the structural information of English characters. From these results, it is apparent that the covers of optical and handwritten characters can be used to design an OIC-based system for optical or handwritten character recognition (OCR/HCR).

Another area in which the OICs of handwritten characters can be used, is ranking the character-prototypes in a large database, which has been presented in Chapter 4 [Bhowmick *et al.* (2007b)]. Recognizing a handwritten character involves matching the character with the prototypes or their features — a complex procedure with a large

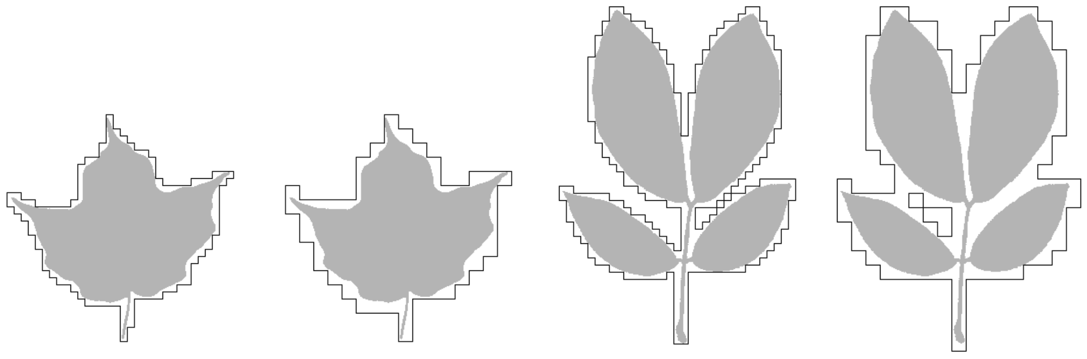


Figure 2.18: The outer isothetic covers (OIC) for two leaf images for $g = 8$ (first one) and $g = 16$ (second one).

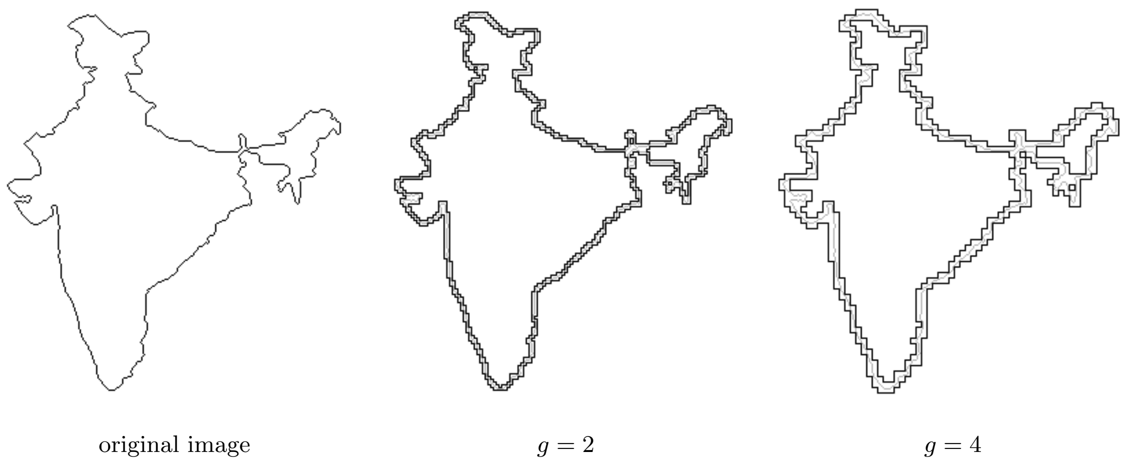
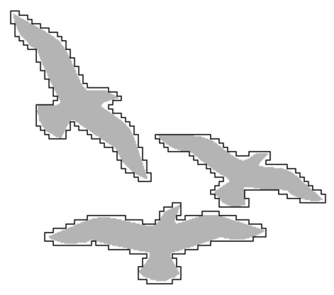
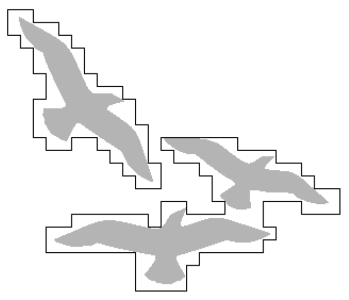


Figure 2.19: OICs corresponding to the image "India" for $g = 2$ and $g = 4$.

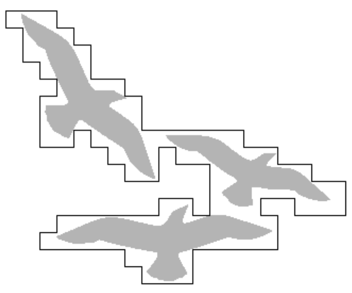
overhead. The process can be made faster if we can arrange the prototypes in some order, using their OICs, so that the recognition time may be reduced. For a given character, a subset of the prototypes, which are almost similar in their OIC-related information, may be replaced by a smaller subset depending on the tradeoff between speed and precision. Similarly, to include a new prototype in the database, we can consider its degree of dissimilarity with the existing prototypes corresponding to the concerned character in the database, and take the decision on including that prototype, accordingly.



$g = 4$



$g = 12$



$g = 16$

Figure 2.20: Numbers of loosely connected components are 3, 2, and 1 for $g = 4$, 12, and 16 respectively.

2.8.4 Scanned Document Images

The proposed algorithm can be used for document image segmentation also. To find the OIC of a document image (Fig. 2.23), the object occupancy of an UGB is decided by checking all the pixels constituting the corresponding UGB. The text lines, which are sufficiently close along the vertical direction, would lie within a single OIC. If two consecutive paragraphs lie in the same OIC for a particular grid size, say $g = g_1$, then these paragraphs can be separated into two different OICs by reducing the grid size appropriately, say to $g = g_2 < g_1$. If the inter-paragraph spacing of a document image is same as its inter-line spacing, then they can be separated by observing that a paragraph starts with a fixed indentation from the left. A document page can thus be segmented into its paragraphs. Similarly, from a segmented paragraph, we can segment its lines by decreasing the grid size further; from a line, we can extract its words; and so forth. Other document features like mathematical equations, figures and graphical objects, tabular structures, etc., can also be extracted from the image using an appropriate analysis of the OICs.

In Fig. 2.24, the OICs (in yellow) are shown corresponding to each letter for grid size $g = 1$. Some of the letters are touching each other; in such a situation, one single OIC is derived (shown in blue). For $g = 5$, all the words are segregated. Thus, with the help of a multi-resolution treatment, different parts of a document page can be segmented using its different sets of OICs.

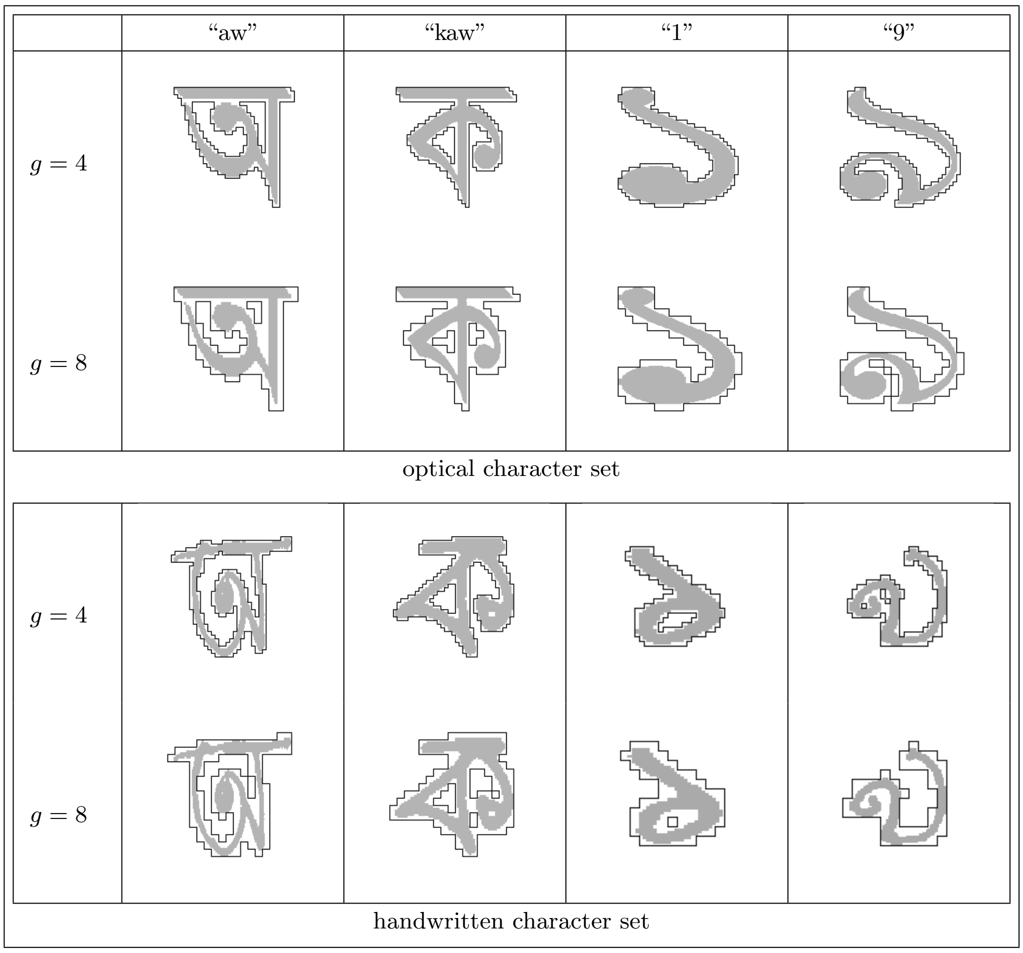


Figure 2.21: The OICs of a few Bengali characters (optical and handwritten) for two different grid sizes.

2.8.5 Inner Isothetic Cover

Figure 2.25 shows the IIC of a logo image for different grid sizes. For grid size $g = 1$ or $g = 2$, the IIC is a single polygon; for $g = 4$, there are three polygons tightly inscribing the object; whereas, for $g = 6$, it is again only one polygon. For $g = 8$, the IIC comprises a total of four polygons. It may be noted that the fractional part of the object lying outside the IIC increases with the increase of the grid size.

As per the definition, an OIC should consist of minimum number of UGBs whose union contains the object without intersecting it, and an IIC should consist of maximum number of UGBs of the object without intersecting the background. Figure 2.25 shows how an

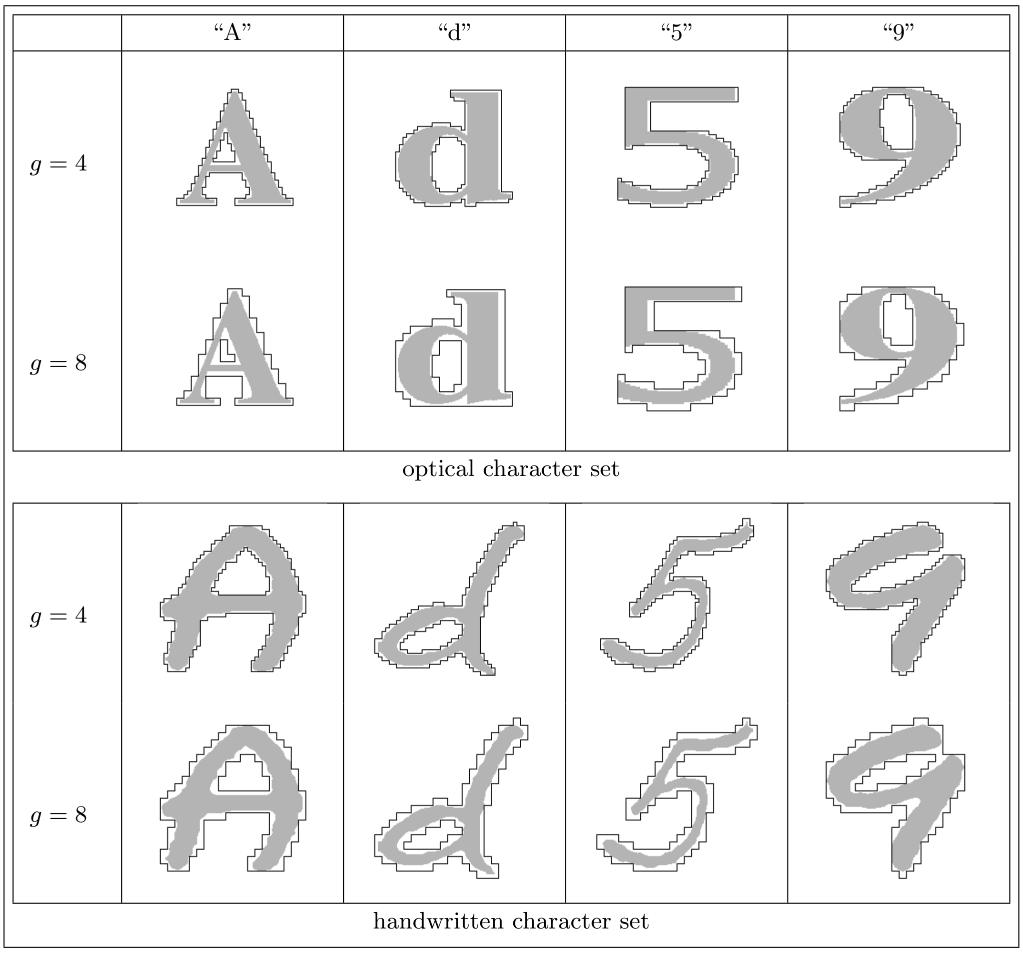
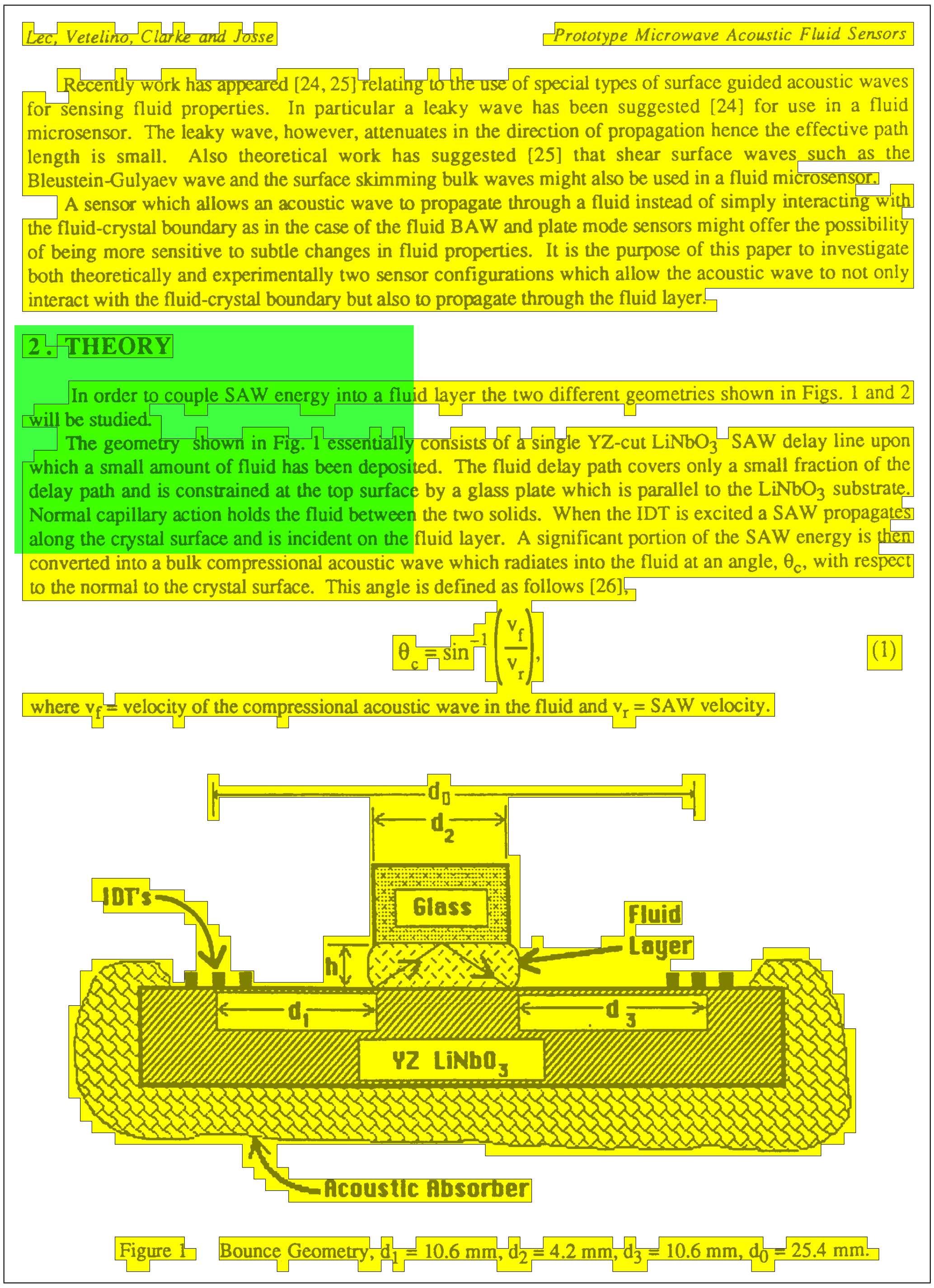


Figure 2.22: The OICs of a few English characters (optical and handwritten) for two different grid sizes.

IIC consists of maximum number of UGBs belonging to the object. The difference of the interior of the IIC from the OIC gives an isothetic region in which the object boundary lies, which has been already illustrated in Fig. 2.1. Figure 2.26 shows the two IICs (bottom row) of the "myth" image for $g = 4$ and $g = 8$ along with two OICs (top row) for the same grid size.

2.8.6 Non-uniform Grid

The isothetic covers on non-uniform grid are shown in Fig. 2.27. The grid lines are generated randomly where the grid spacing g varies between 4 to 10. The result shows



Lec, Vetelino, Clarke and Josse *Prototype Microwave Acoustic Fluid Sensors*

Recently work has appeared [24, 25] relating to the use of special types of surface guided acoustic waves for sensing fluid properties. In particular a leaky wave has been suggested [24] for use in a fluid microsensor. The leaky wave, however, attenuates in the direction of propagation hence the effective path length is small. Also theoretical work has suggested [25] that shear surface waves such as the Bleustein-Gulyaev wave and the surface skimming bulk waves might also be used in a fluid microsensor.

A sensor which allows an acoustic wave to propagate through a fluid instead of simply interacting with the fluid-crystal boundary as in the case of the fluid BAW and plate mode sensors might offer the possibility of being more sensitive to subtle changes in fluid properties. It is the purpose of this paper to investigate both theoretically and experimentally two sensor configurations which allow the acoustic wave to not only interact with the fluid-crystal boundary but also to propagate through the fluid layer.

2. THEORY

In order to couple SAW energy into a fluid layer the two different geometries shown in Figs. 1 and 2 will be studied.

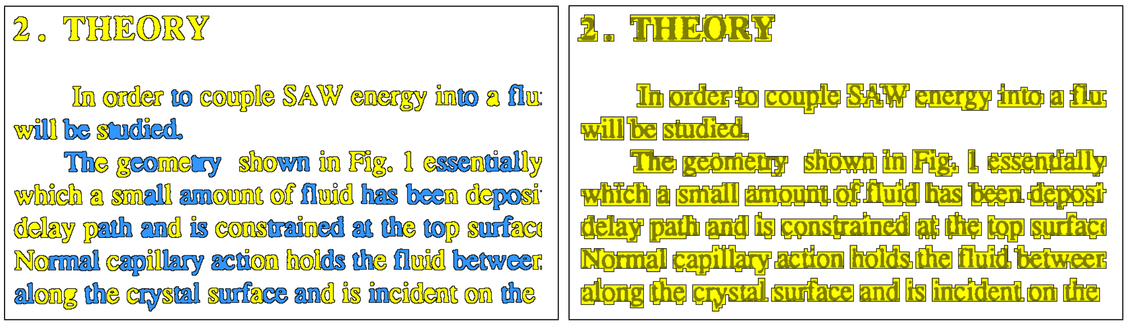
The geometry shown in Fig. 1 essentially consists of a single YZ-cut $LiNbO_3$ SAW delay line upon which a small amount of fluid has been deposited. The fluid delay path covers only a small fraction of the delay path and is constrained at the top surface by a glass plate which is parallel to the $LiNbO_3$ substrate. Normal capillary action holds the fluid between the two solids. When the IDT is excited a SAW propagates along the crystal surface and is incident on the fluid layer. A significant portion of the SAW energy is then converted into a bulk compressional acoustic wave which radiates into the fluid at an angle, θ_c, with respect to the normal to the crystal surface. This angle is defined as follows [26],

$$\theta_c = \sin^{-1}\left(\frac{v_f}{v_r}\right), \qquad (1)$$

where v_f = velocity of the compressional acoustic wave in the fluid and v_r = SAW velocity.

Figure 1 Bounce Geometry, d_1 = 10.6 mm, d_2 = 4.2 mm, d_3 = 10.6 mm, d_0 = 25.4 mm.

Figure 2.23: The OIC of a document image for grid size $g = 24$ corresponding to different regions of interest in a typical document page.


2. THEORY

In order to couple SAW energy into a flu
will be studied.
The geometry shown in Fig. 1 essentially
which a small amount of fluid has been deposi
delay path and is constrained at the top surfac
Normal capillary action holds the fluid betweer
along the crystal surface and is incident on the

2. THEORY

In order to couple SAW energy into a flu
will be studied.
The geometry shown in Fig. 1 essentially
which a small amount of fluid has been deposi
delay path and is constrained at the top surfac
Normal capillary action holds the fluid betweer
along the crystal surface and is incident on the

Figure 2.24: The OICs of a part of a document image for grid sizes $g = 1$ (left) and $g = 5$ (right) show their abilities in extracting the letters and the words from a document page.

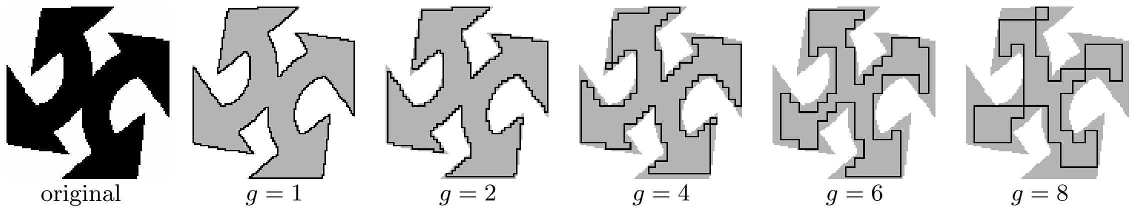


Figure 2.25: The IICs of a logo image corresponding to different grid sizes.

that the isothetic covers (outer and inner) can be extracted correctly using the same principle which has been employed for uniform grid lines. The isothetic covers on non-uniform grid is particularly useful for finding the inner and outer approximations in rough sets.

In Fig. 2.28, we have shown the OICs of the binarized images corresponding to different real-world gray-scale images for grid size $g = 8$.

2.9 Conclusion

We have shown how the minimum-(maximum-)area outer (inner) isothetic cover of a digital object can be constructed corresponding to a given grid. The algorithm proposed here does not require any backtracking, and hence is an output-sensitive algorithm. The time complexity is linear on the length of the perimeter of the cover measured in grid units. Experimental results on various databases justify the efficiency of the algorithm and demonstrate potential applications.

Several open problems may arise in context to an isothetic cover of a digital object. It is evident that such a cover of a digital object depends on how the object is positioned

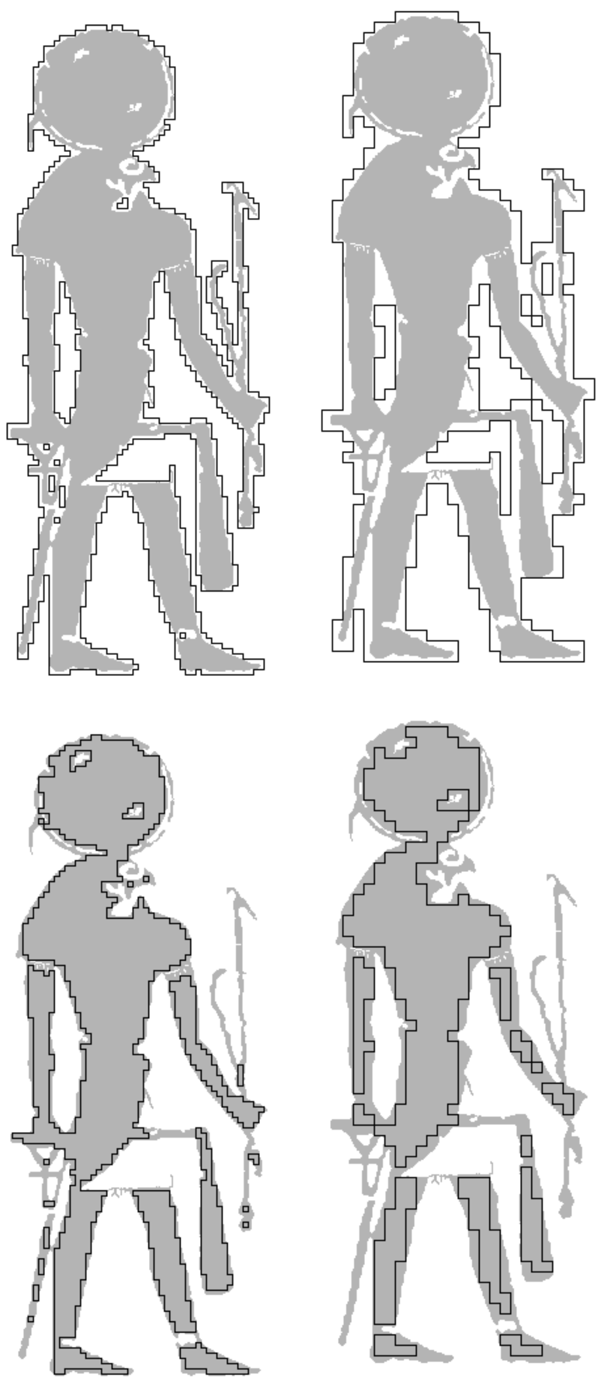


Figure 2.26: OICs (top row) and IICs (bottom row) corresponding to the image “myth” for $g = 4$ and $g = 8$.

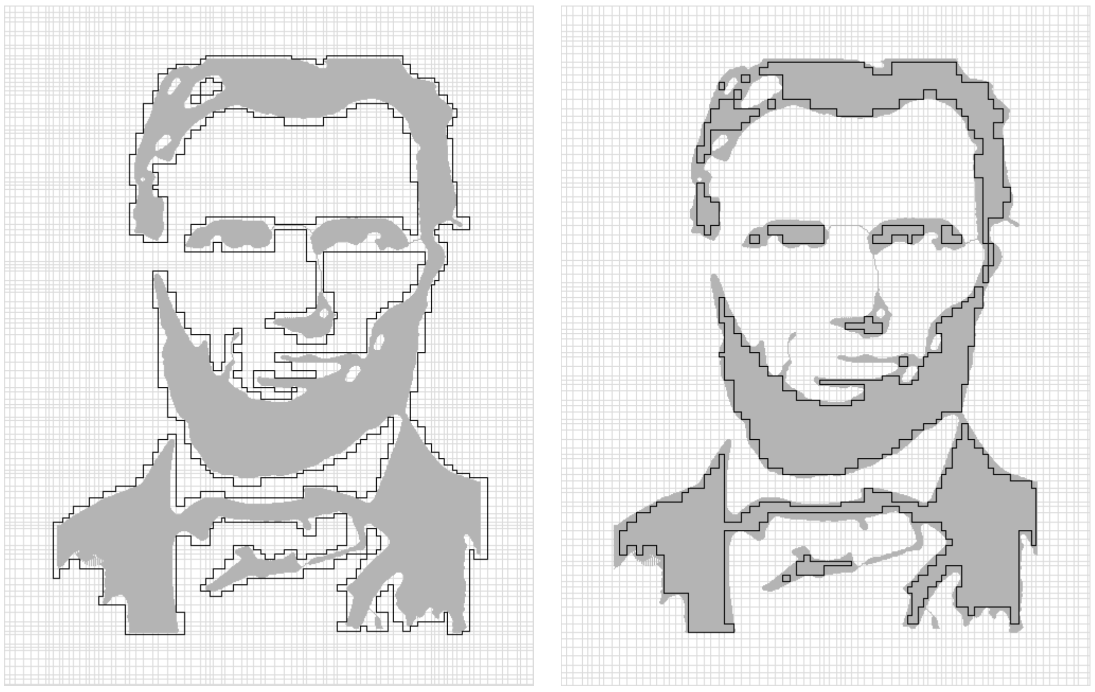

(a) Outer: $g = 4 - 10$. (b) Inner: $g = 4 - 10$.

Figure 2.27: Inner and outer polygons of "Lincoln" image on non-uniform grids.

or oriented w.r.t. the underlying grid. A challenging problem is, therefore, finding the object-grid registration for which the complexity of an isothetic cover is minimum. The complexity measure may be in terms of the number of vertices, or the perimeter, or the area of the isothetic cover. Another interesting problem is, given a collection of isothetic covers corresponding to different positions or orientations of the object for a grid size, to design an algorithm to (approximately) reconstruct the original object. The algorithm can be extended to higher dimensions for modeling 3-D objects. Also, the algorithm when extended to the background of non-uniform grid for higher dimensions can be useful in determining the upper and lower approximations in rough sets.

There are several applications of isothetic polygons, which have been mentioned in Sec. 2.1 with some results shown in Sec. 2.8. One such application is document image analysis, where there still remains ample scope of exploiting isothetic polygons for solving related problems. Extension of the proposed algorithm to 3- and to higher dimensional digital space is possible following the same principle, however the neighborhood relationship for object occupancy has to be redefined. This requires further investigation.

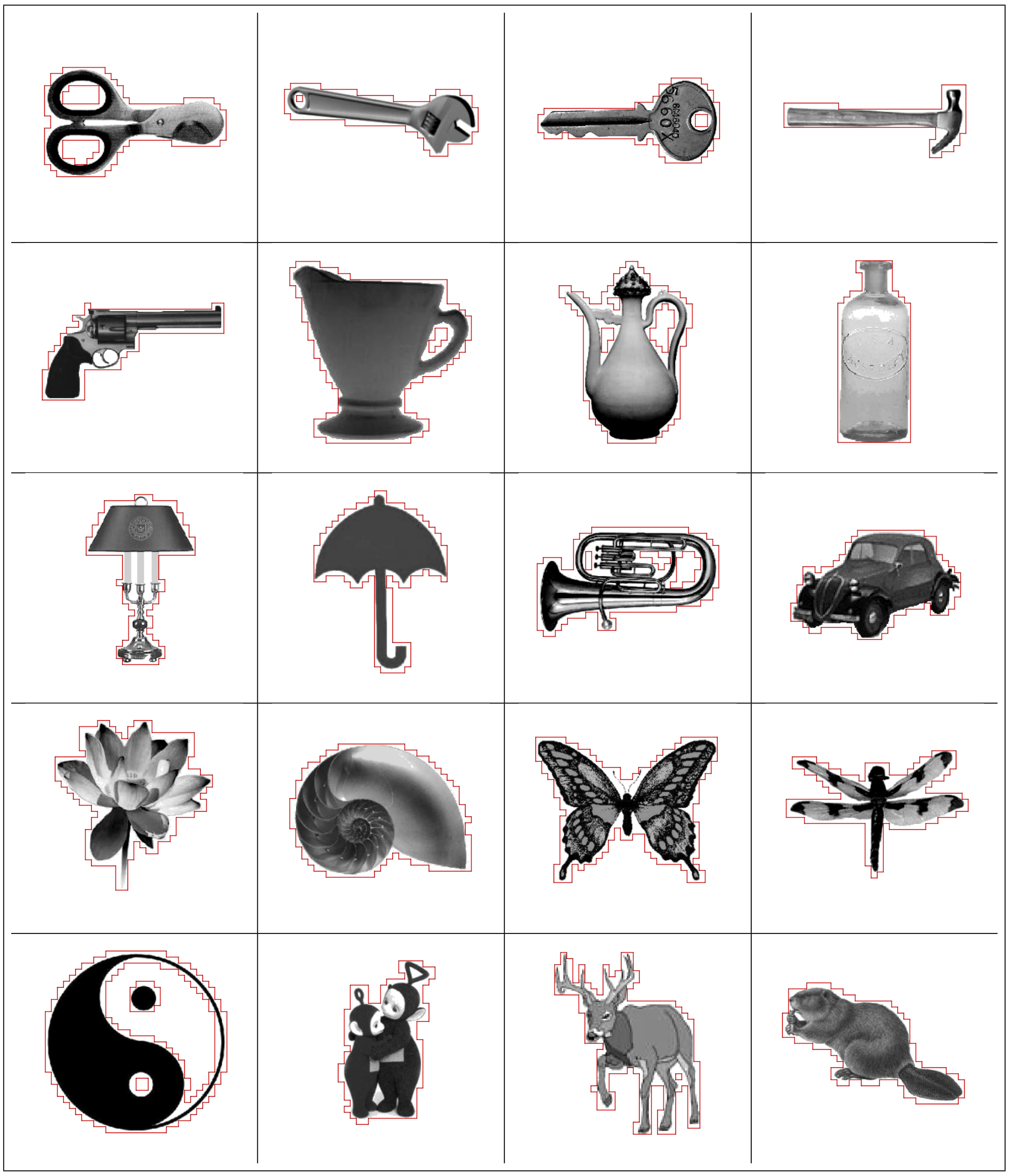


Figure 2.28: The OICs ($g = 8$) of the binarized images corresponding to different real-world gray-scale images.

Chapter 3

Construction of Orthogonal Hull of a Digital Object

3.1 Introduction

The convex hull of an object A, denoted by $CH(A)$, is the smallest convex set that contains A. There exist a number of computational geometric algorithms [Berg *et al.* (2000), Cormen *et al.* (2000), Preparata and Shamos (1985)] to find the convex hull of a point set or a polygonal object A having arbitrary shape on the real/digital plane. The time complexities of some of the well-cited ones are of order $O(n^3)$ (brute force), $O(n \log n)$ (Graham scan [Graham (1972)]), $O(nh)$ (Jarvis march [Jarvis (1973)]), and $O(n \log h)$ (Kirkpatrick-Siedel's algorithm [Kirkpatrick and Seidel (1986)]), where, n is the number of points/vertices constituting A, and h is the number of vertices of $CH(A)$. Also, there are other algorithms for finding the convex hull, e.g., [Barber *et al.* (1993), Chazelle (1993), Swart (1985)]. A detailed analytical study of the convex hull algorithms is also available in the literature [Avis and Bremner (1995)].

Apart from the concept of convex hull, other types of hulls, such as pseudo-hull, near-hull [Klette and Rosenfeld (2004a)], digital convex hull [Chaudhuri and Rosenfeld (1998)], relative convex hull [Sklansky and Kibler (1976)], and α-hull [Edelsbrunner (1992)], can be also found in the literature, which are designed for specific applications. However, the execution of these algorithms for a sufficiently large digital object is not as fast as required in a practical application. This is caused by the inherent procedural complexities in these algorithms. For example, in Graham scan, to decide whether there is a left-turn or a right-turn or no turn at a point p_i, considering its previous point p_{i-1} and its next point p_{i+1}, the sign of a 3×3 determinant (which includes the coordinates of the three consecutive points) is used. Computation of this determinant needs multiplication apart from comparison and addition/subtraction. Similarly, in Jarvis march, polar coordinates of the points are computed from their Cartesian coordinates, which involves trigonometric operations. On the contrary, in our algorithm, only comparison and addition/subtraction are required in the integer domain, while computing the orthogonal hull of a digital object.

3.1.1 Motivation and Related Works

Given a 2D digital object S imposed on the background digital grid $\mathcal{G}$ (Defn. 2.2.5) consisting of a set of equi-spaced horizontal and vertical lines, the problem is to compute the orthogonal hull (Defn. 3.2.1) of S. It is evident that the resulting orthogonal hull is dependent upon the registration of the object with the background grid. Also, the precision and the complexity of the hull can be made to change by varying the distance between two consecutive (horizontal and vertical) grid lines, thereby making it amenable to multi-grid treatment. Depending on the requirement of an application, the specification of an orthogonal hull covering a digital object may be tuned, therefore, in order to suit the desired criterion.

Orthogonal hulls find real-world applications in today's diversifying technologies on modern computing and digital imaging. For example, in designing a fault-tolerant algorithm in mesh-connected computers, it is important to define a faulty region that is convex, and at the same time, to include a minimum number of non-faulty nodes. Recently, an algorithm has been presented by Wu and Jiang (2005) to compute the minimum orthogonal convex polygon, which includes the faulty blocks and minimum number of non-faulty nodes. The algorithm is based on a labeling scheme. It grows the region with faulty nodes and finally shrinks to form the orthogonal convex polygon. The algorithm is particularly suitable for a set of nodes representing the processors in an orthogonal grid. However, it is not suitable for image analysis, since it is designed to operate on a small number of nodes.

Some other typical applications involving orthogonal hulls are analysis of land-mark data, shape analysis and classification, measuring the polygonal entropy, and many such areas of computer vision and pattern recognition [Bookstein (1991), Costa and R. M. Cesar (2001), Hyde *et al.* (1997), Pitty (1984)]. Orthogonal convex polygons also find use in models of polymers, cell growth, and percolation [Bousquet-Mélou (1996)]. In discrete tomography, the reconstruction of discrete sets is done using the concept of hv-convex discrete sets [Balazs (2008)]. A properly defined convex polygon describing a real or a digital object is often considered to be the domain of interest of the underlying object. As a result, the subject has received a considerable attention amongst researchers [Boxer

(1993), Sonka *et al.* (1993), Stern (1989), Zunic and Rosin (2004)].

Karlsson and Overmars (1988) have presented an algorithm using scanline technique to compute the orthogonal (convex) hull for a set of points in $\mathbb{Z}^2$ with a time complexity $O(n \log\log_n u)$, where n is the set size and u is the grid size. The survey paper by Audet *et al.* (2007) elaborates several optimization issues related to the problem of finding an empty convex polygon of maximum area or perimeter amidst a point set in $\mathbb{R}^2$. Recently, an algorithm has been proposed by Nandy *et al.* (2008) to find the largest empty ortho-convex polygon in a (possibly sparse and scattered) point set in $\mathbb{R}^2$. These algorithms use the scan-line strategy after doing a lexicographical sorting of the points in $\mathbb{R}^2$. The proposed algorithm, on the contrary, finds the orthogonal hull for a given object in $\mathbb{Z}^2$, which is defined by one or more connected components. The algorithm avoids any sorting and finds the hull while traversing tightly around the object contour. The runtime of the proposed algorithm has been shown to be, therefore, proportional to the perimeter of the object.

The concept of row-convex/column-convex polygons are found in the literature [Klette and Rosenfeld (2004a)], and there exist some works related with such polygons [Bousquet-Mélou (1996), Bousquet-Mélou and Fédou (1995)]. The latter work deals with the enumeration of different classes of column-convex polygons using complex generating functions, according to their perimeter, width, and area. The formula for generating convex polyominoes has also been suggested [Bousquet-Mélou (1996)]. A column-convex polygon is characterized by its nature of intersection with a horizontal or a vertical line segment. In contrast with an orthogonal hull that has always a single line segment when intersected by a horizontal or a vertical line, a column-convex polygon may have multiple line segments as a result of intersection with a horizontal line segment. Further, an implementation of the generating functions in the digital plane is not readily realizable for their complex nature.

3.1.2 Main Results

We have designed and tested a novel algorithm for finding the orthogonal hull (Defn. 3.2.1) of a given digital object such that the hull edges lie on a set of equally spaced horizontal and vertical grid lines. The ordered list of hull vertices is obtained by an analysis of the object occupation of the four neighboring quadrants corresponding to a grid point (Defn. 2.2.5) lying near the boundary of the object. The orthogonal hull consists of fewer vertices with an increase of the grid size (spacing between two consecutive horizontal/vertical grid

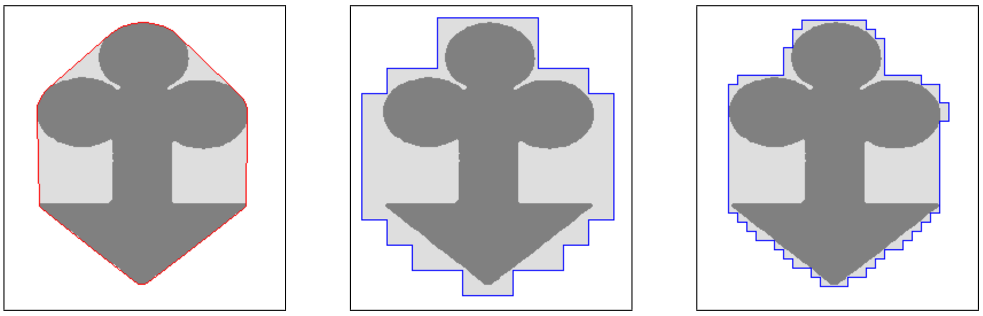


Figure 3.1: A sample 2D object, its convex hull (left), and its orthogonal hulls for grid size $g = 22$ (middle) and $g = 8$ (right).

lines), enabling an analysis of the object, from fine to coarse. The algorithm is based on the fact that a polygon is orthogonally convex if and only if a counterclockwise traversal of its boundary never makes two consecutive right turns (alternatively, a clockwise traversal of its boundary never makes two consecutive left turns). The algorithm involves only comparison and addition/subtraction in the integer domain, and hence runs very fast, as demonstrated by the CPU time in our experiments (Sec. 3.5).

The convex hull and the orthogonal hulls for $g = 22$ and $g = 8$, corresponding to a digital object (22404 pixels), are shown in Fig. 3.1. The convex hull algorithm (Graham scan) on a digital object shown in this figure takes 2573 milliseconds, whereas the proposed algorithm on finding the orthogonal hull takes only a few milliseconds ($g = 22 : 0.94$ milliseconds, $g = 8 : 3.16$ milliseconds). The time required by the Graham scan algorithm may be reduced by considering the object contour instead of the entire object as input; but finding the object contour needs an edge extraction algorithm. On the contrary, given a digital object, the proposed algorithm runs on the object contour without resorting to any edge extraction.

Apart from speed, the proposed algorithm has the ability to capture the shape information of an arbitrary object. For example, for $g = 22$, the orthogonal hull of the object shown in Fig. 3.1 is vertically symmetrical, which conforms to the vertical symmetry of the object; for $g = 8$, the orthogonal hull is also almost symmetrical. The vertices of the orthogonal hull are reported in counterclockwise order in terms of their types (90^0 and 270^0), from which the symmetry can be ascertained. The regions of OIC that give rise of non-convexity are removed based on a combinatorial analysis These regions also capture the shape complexity of the concerned object, which can be used in subsequent applications.

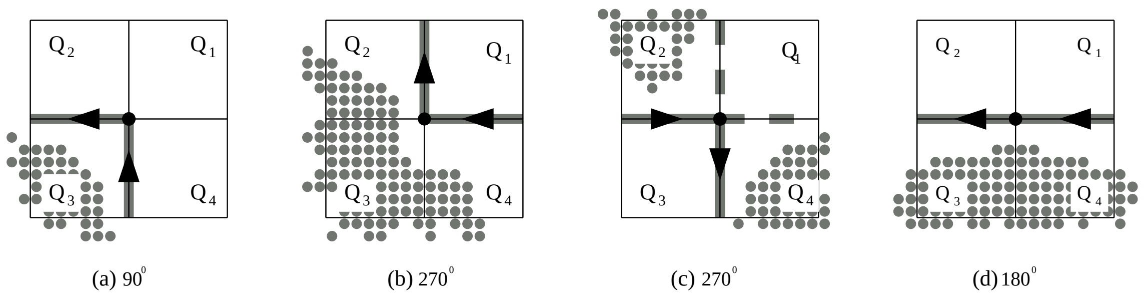


Figure 3.2: Different vertex types of an orthogonal polygon are decided using the object (gray dots) occupation in the four cells/quadrants incident at a grid point.

Rest of the chapter is organized as follows. Section 3.2 contains the definitions and preliminaries, and summarizes how a tight orthogonal traversal around the contour of a digital object can be made. Section 3.3 describes the rules required to find the orthogonal hull from the orthogonal traversal explained in Sec. 3.2. Section 3.4 describes the algorithm to construct the orthogonal hull, presents a detailed demonstration, and justifies the time complexity of the algorithm. The experimental results and their physical analyses are reported in Sec. 3.5. Finally, the concluding notes along with some directions for future work, are presented in Sec. 3.6.

3.2 Definitions and Preliminaries

Definition 3.2.1. *The* orthogonal convex hull*, or simply* orthogonal hull*, of a digital object S, denoted by $OH(S)$, is the smallest-area isothetic polygon such that (i) each point $p \in S$ lies inside $OH(S)$ and (ii) intersection of $OH(S)$ with any horizontal or vertical line is either empty or exactly one line segment.*[1]

3.2.1 Orthogonal Traversal of the Object Contour

In order to detect and remove the concavities, we traverse around the object contour, orthogonally along the grid lines. The nature of traversal is such that we visit the vertices (in order) of the smallest-area orthogonal cover of the digital object using an efficient com-

[1]It may be noted that, a discrete set is referred as horizontally and vertically convex (shortly, *hv*-convex) in digital tomography [Balazs (2008)] if all the rows and columns of the set are 4-connected. We provide the above definition in the perspective of the real plane, since it helps proving the correctness of our algorithm in a simpler way.

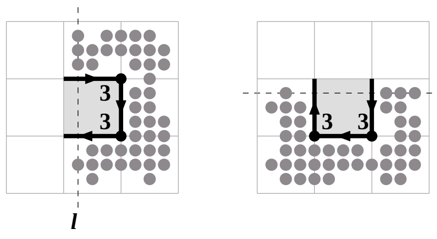


Figure 3.3: A concave region possesses two or more consecutive vertices of Type **3**, which give rise to multiple intersections with a vertical line (left) or a horizontal line (right).

binatorial technique based on object containments of the four cells incident at a particular grid point [Biswas *et al.* (2005b)]. The classification of a grid point, q, is based on the object containment of the four cells ($Q_1 - Q_4$) incident at q (Fig. 3.2), as discussed in Sec. 2.5.1.

During the traversal, a grid point is determined either as a vertex or as a non-vertex point. Since we traverse orthogonally, a grid point, q, if detected as a vertex, can be a 90^0 vertex or a 270^0 vertex. Otherwise, q is simply a point on the edge of the orthogonal cover, or a grid point lying inside the object and also inside the orthogonal cover, or lying outside the object and also outside the orthogonal cover. Henceforth in this work, a 90^0 vertex is referred to as a Type **'1'** vertex ($\mathbf{1} \times 90^0$), and a 270^0 vertex as a Type **'3'** vertex ($\mathbf{3} \times 90^0$) for ease of notation.

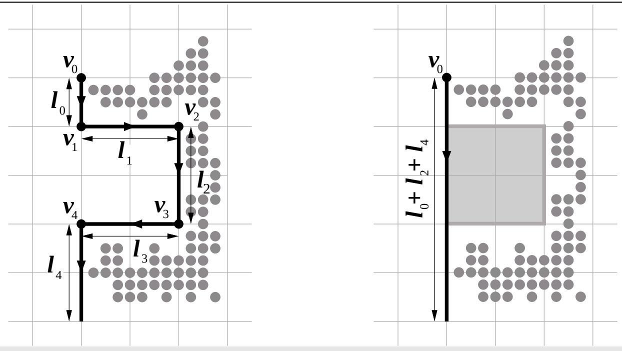
An intermediate vertex, v_i, is represented by a three-tuple $\langle t_i, d_i, l_i \rangle$, where, t_i (= **1** or **3**) is the type of the vertex, d_i is the direction of traversal from v_i to v_{i+1}, and l_i is the length of the (horizontal/vertical grid-) line segment from v_i to v_{i+1}. The direction of traversal, d_i, from v_i, can assume the value $0, 1, 2,$ or 3, indicating the direction towards right, top, left, or bottom respectively. The direction of traversal d_i is derived from the previous direction, d_{i-1} (the direction from v_{i-1} along which v_i has been traversed to), and the type of the vertex v_i, and is given by $d_i = (d_{i-1} + t_i) \bmod 4$. Once d_i is computed, the next grid point is determined and its class is evaluated. Thus the traversal proceeds from v_i to v_{i+1} and finally concludes when it returns back to start vertex. It may be noted that the start vertex, v_s, of the traversal is determined by a row-wise scan of the grid points (see Sec. 2.5.2). The start vertex v_s is always a 90^0 vertex, since v_s is the top-left grid point that is classified as a vertex during the row-wise scan. The direction of traversal d_s, from the start vertex is towards bottom for a counterclockwise traversal. So, t_s and d_s for v_s have the values 1 and 3 respectively.

3.3 Rules for Finding the Orthogonal Hull

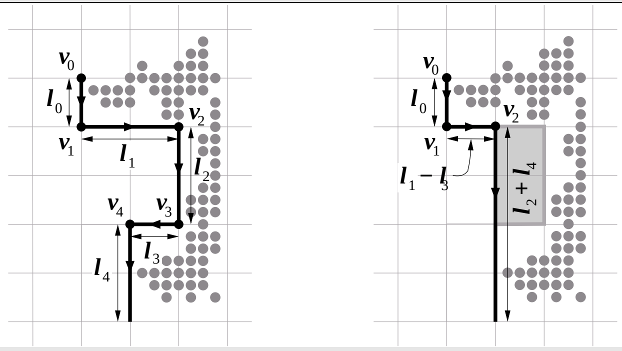
The concavities present in the orthogonal cover are detected and removed when the object boundary is traversed orthogonally along the grid lines as mentioned in Sec. 3.2.1. More importantly, the proposed algorithm finds the orthogonal hull of a digital object without any prior knowledge about its orthogonal cover. Deriving the orthogonal hull, therefore, proceeds with the orthogonal traversal around the object boundary.

During the traversal, if two vertices of Type **3** appear consecutively, then it implies a concave region, which defies the property of orthogonal convexity (Defn. 3.2.1). Illustrated in Fig. 3.3 are two such patterns for which the intersection of a vertical or a horizontal line, l, with the orthogonal polygon has more than one segment. Our goal is to identify such regions and derive the edges of the orthogonal hull such that the properties of orthogonal convexity are maintained. In this incremental algorithm, the part of the orthogonal hull obtained upto a point does not contain two consecutive vertices of Type **3**, which acts as the invariant of the algorithm. Whenever such an occurrence appears, we apply necessary reduction rules to maintain the invariant and to ensure the orthogonal convexity, thereof. However, rest of the patterns, **13**, **31**, and **11**, are in conformance with the algorithm's invariant and hence do not violate the properties of orthogonal convexity. Hence, in effect, our strategy is to remove all patterns of **33** during the orthogonal traversal such that the resulting orthogonal polygon is free of two consecutive vertices of Type **3** and is also area-minimized.

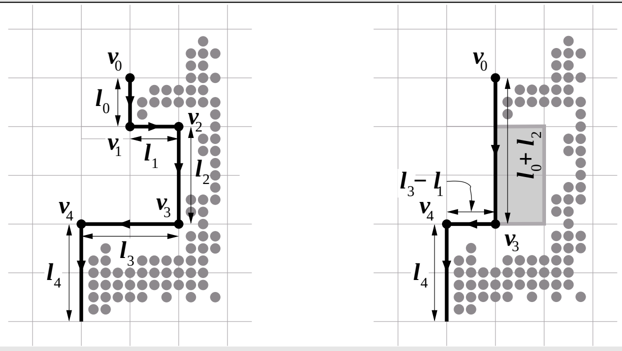
Let v_0, v_1, v_2, v_3, and v_4 be five consecutive vertices for which the rule has to be applied in order to remove the concavity, if any, where v_4 is the most recently traversed vertex, and $v_0 \ldots v_3$ are the previous four vertices already visited. Since a reduction rule is applied only when two consecutive vertices are of Type **3**, we have designed two sets of rules depending on whether the type of the vertex following the pattern **33** is **1** or **3**. We consider that the two consecutive vertices of Type **3** are designated by v_2 and v_3, and the vertex v_1 preceding v_2 is always of Type **1**. The type of the vertex v_0 can be either **1** or **3**. For, in our algorithm, the traversal always starts from a vertex of Type **1**, which is verified from the combinatorial arrangement of its four neighboring cells, as explained in Sec. 3.2.1. Adopting this policy of starting the traversal always ensures that there will be at least one vertex of Type **1** preceding two consecutive vertices of Type **3**. A dummy vertex is introduced before the start vertex to handle the situation when two consecutive Type **3** vertices appear immediately after the start vertex. It may be observed that only the Rule **R12** may have to be applied in such a case.



Rule R11 $(l_1 = l_3)$:
$\langle v_0(\mathbf{t_0}, l_0), v_1(\mathbf{1}, l_1), v_2(\mathbf{3}, l_2), v_3(\mathbf{3}, l_3), v_4(\mathbf{1}, l_4)\rangle \rightarrow$
$\langle v_0(\mathbf{t_0}, l_0 + l_2 + l_4)\rangle$



Rule R12 $(l_1 > l_3)$:
$\langle v_0(\mathbf{t_0}, l_0), v_1(\mathbf{1}, l_1), v_2(\mathbf{3}, l_2), v_3(\mathbf{3}, l_3), v_4(\mathbf{1}, l_4)\rangle \rightarrow$
$\langle v_0(\mathbf{t_0}, l_0), v_1(\mathbf{1}, l_1 - l_3), v_2(\mathbf{3}, l_2 + l_4)\rangle$



Rule R13 $(l_1 < l_3)$:
$\langle v_0(\mathbf{t_0}, l_0), v_1(\mathbf{1}, l_1), v_2(\mathbf{3}, l_2), v_3(\mathbf{3}, l_3), v_4(\mathbf{1}, l_4)\rangle \rightarrow$
$\langle v_0(\mathbf{t_0}, l_0 + l_2), v_3(\mathbf{3}, l_3 - l_1), v_4(\mathbf{1}, l_4)\rangle$

Figure 3.4: Concavity (shaded regions) detection and removal rules for pattern **1331**.

3.3.1 Pattern 1331:

This pattern signifies a Type **1** vertex followed by two consecutive Type **3** vertices and another Type **1** vertex. Occurrence of two consecutive **3**s essentially signifies a concavity in the object, as explained earlier. The rules for removal of the associated concavities are stated in Fig. 3.4. The concave regions are detected and coalesced to their corresponding convex products using the related edge lengths in the reduction mechanism. There can arise three cases depending on the relation between l_1 and l_3, which are as follows:

Rule **R11***:* Applied when $l_1 = l_3$.
Vertices $v_1, v_2, v_3,$ and v_4 are removed, and the length of v_0 is modified to $l_0 + l_2 + l_4$.

Rule **R12***:* Applied when $l_1 > l_3$.
The lengths l_1 and l_2 are modified as $l_1 - l_3$ and $l_2 + l_4$ respectively. The vertices v_3 and v_4 are removed.

Rule **R13***:* Applied when $l_1 < l_3$.
Here, v_1 and v_2 are removed, and l_0 and l_3 are updated to $l_0 + l_2$ and $l_3 - l_1$ respectively.

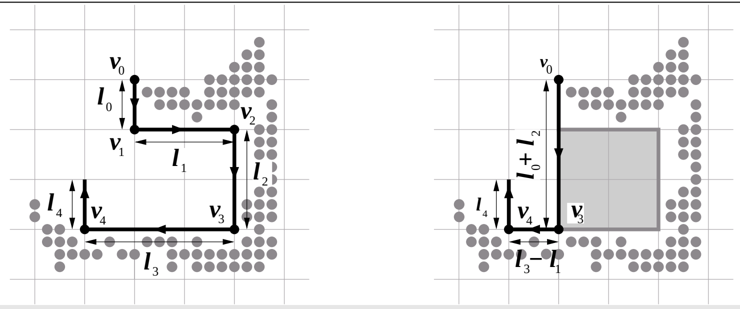
Note that, in each of the above three cases, the type t_0 of the vertex v_0 remains unaltered, which holds true and causes no problem even if v_0 is the dummy vertex.

3.3.2 Pattern 1333:

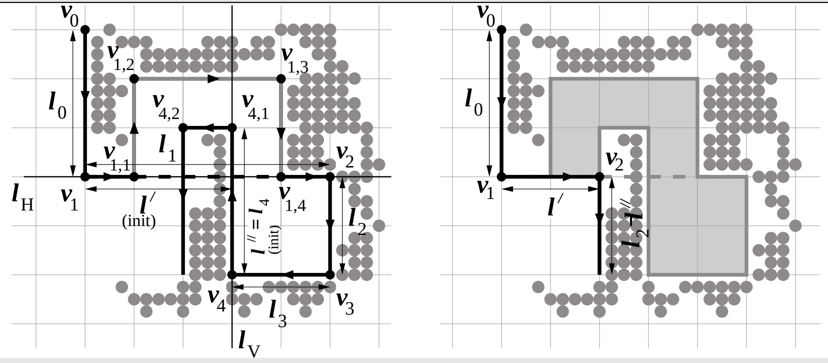
Such a pattern signifies a convoluted object boundary. Hence, the traversal is continued until the orthogonal chain comes out of the convoluted (and non-convex, thereof) region. The rules for removal of concavity corresponding to this pattern, shown in Fig. 3.5, are as follows.

Rule **R21***:* Applied when $l_1 < l_3$.
Here, v_1 and v_2 are removed, and l_0 and l_3 are updated to $l_0 + l_2$ and $l_3 - l_1$ respectively.

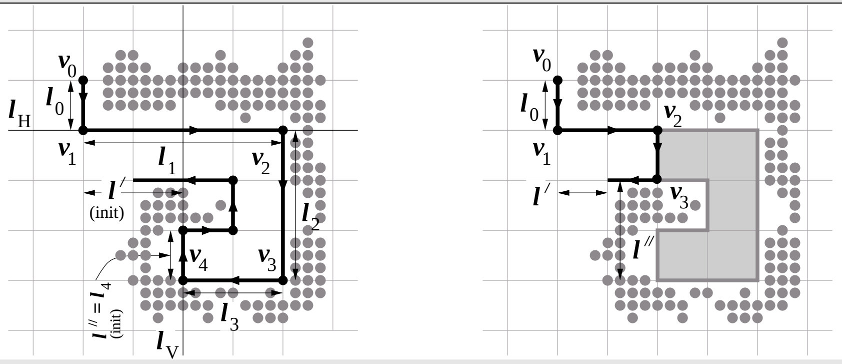
Rule **R22***:* Applied when $l_1 \geqslant l_3$ and $d = d_2$.
Three Type **3** vertices in succession indicate the beginning of a convoluted region (Fig. 3.5). The traversal is continued from the vertex v_4. Let v denote the current vertex up to which the traversal has progressed so far. Let l_H be the horizontal line passing through v_2 and l_V be the vertical line passing through v_4. The reduction rule is applied only when v lies below the half-plane defined by l_H and to the left of the half-plane defined by l_V simultaneously; otherwise the traversal is continued. As any point above (and inclusive of) l_H or right (and inclusive) of l_V falls within the concave region, no action is taken as long as the current vertex v lies within this region. Two variables, l' and l'', which are



Rule R21 $(l_1 < l_3)$:
$\langle v_0(\mathbf{t_0}, l_0),\ v_1(\mathbf{1}, l_1),\ v_2(\mathbf{3}, l_2),\ v_3(\mathbf{3}, l_3),\ v_4(\mathbf{3}, l_4)\rangle \rightarrow$
$\langle v_0(\mathbf{t_0}, l_0 + l_2), v_3(\mathbf{3}, l_3 - l_1), v_4(\mathbf{3}, l_4)\rangle$



Rule R22 $(l_1 \geqslant l_3$ and $d = d_2)$:
$\langle v_0(\mathbf{t_0}, l_0),\ v_1(\mathbf{1}, l_1),\ v_2(\mathbf{3}, l_2),\ v_3(\mathbf{3}, l_3),\ v_4(\mathbf{3}, l_4)\rangle \rightarrow$
$\langle v_0(\mathbf{t_0}, l_0),\ v_1(\mathbf{1}, l'),\ v_2(\mathbf{3}, l_2 - l'')\rangle$



Rule R23 $(l_1 \geqslant l_3$ and $d = d_3)$:
$\langle v_0(\mathbf{t_0}, l_0),\ v_1(\mathbf{1}, l_1),\ v_2(\mathbf{3}, l_2),\ v_3(\mathbf{3}, l_3),\ v_4(\mathbf{3}, l_4)\rangle \rightarrow$
$\langle v_0(\mathbf{t_0}, l_0), v_1(\mathbf{1}, l_1 - l_3), v_2(\mathbf{3}, (l_2 - l''), v_3(\mathbf{3}, (l_1 - l_3 - l')\rangle$

Figure 3.5: Concavity detection and removal rules for pattern **1333**.

initialized as $l_1 - l_3$ and l_4 respectively, are used to determine whether v has come out of the convoluted-cum-concave region. The lengths l' and l'' are updated depending upon the direction of traversal from v. If the direction of traversal from v, denoted by d, is same as the direction of traversal from v_1, i.e., d_1, then l' is updated as $l' = l' + l$, where l is the traversed length from v. On the contrary, if the direction of traversal is same as that of v_3, then l' is updated as $l' = l' - l$. Similarly, l'' is updated as $l'' = l'' + l$ or $l'' = l'' - l$, depending on whether $d = d_4$ or $d = d_2$. The traversal is continued until the conditions i) $l' < l_1 - l_3$, and ii) $l'' < l_2$ are fulfilled. Once these conditions are reached, the reduction rule is applied.

As stated above, the quarter-plane below l_H and left of l_V is the region where, when the traversal reaches, a reduction rule is applied. Entry to this quarter-plane can occur either from the half-plane above l_H or from the half-plane right of l_V. Rule **R22** formulates the reduction for the former case, i.e., when the direction of exit from convoluted region is same as d_2. The length l_1 is modified as l' and l_2 is modified as $l_2 - l''$. The vertices v_3 and v_4 are removed.

For example, in Fig. 3.5 (middle), the sequence of vertices for reduction is $\langle v_0, v_1, v_2, v_3, v_4 \rangle$. Notice that this chain is obtained after the reduction of $v_1, v_{1,1}, v_{1,2}, v_{1,3}, v_{1,4}$ (Rule **R21**). When $v_{4,1}$ is visited, no reduction is done as $l' < l_1 - l_3$ and $l'' > l_2$, and the traversal is continued to $v_{4,2}$. At $v_{4,2}$, it is found that $l'' < l_2$ and $l' < l_1 - l_3$, and hence the reduction rule **R22** is applied.

Rule **R23***:* Applied when $l_1 \geqslant l_3$ and $d = d_3$, i.e., when an entry to the quarter-plane below l_H and left of l_V occurs from the right.
The direction of traversal is d_3 when the conditions i) $l' < l_1 - l_3$ and ii) $l'' < l_2$ are fulfilled. The vertex v_4 is removed and l_1, l_2, and l_3 are modified as follows: $l_1 = l_1 - l_3$, $l_2 = l_2 - l''$, and $l_3 = l_1 - l_3 - l'$.

To summarize, each rule for the pattern **1331** removes a non-convoluted concave region depending on l_1 and l_3, whereas each rule for the pattern **1333** finds the start of a convoluted region and reduces when the traversal finally comes out of the convoluted region, using the lengths $l_0, \ldots, l_4$. It may be noted that the outcome of the Rule **R21** and the Rule **R23** has a pattern **33** at the end. This pattern is removed when the next vertex is visited and the appropriate rule is applied depending on its type.

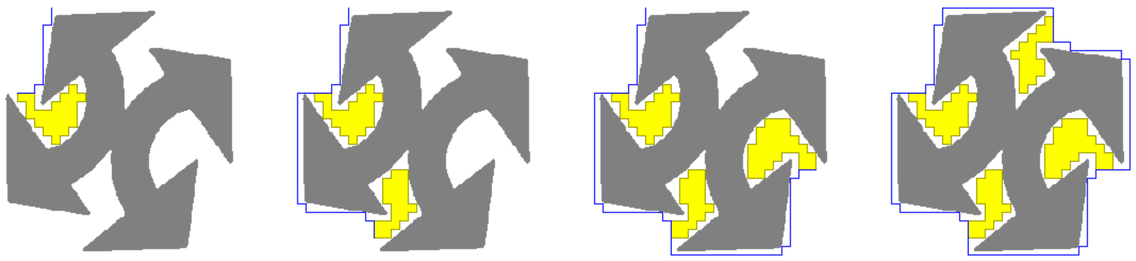


Figure 3.6: Demonstration of the algorithm on a sample 2D object. Each image shows the result after removal of the concave parts in successive steps.

3.3.3 Applying the Rules

The rules for detection and removal of concavities are applied while traversing around the object contour (without touching the contour) in an orthogonal path. A list L is initialized with a dummy vertex, whose necessity is explained earlier in Sec. 3.3. The traversal starts from the start vertex, which is determined by a row-wise scan of the grid points. As the traversal is continued, each new vertex visited is appended to the list L. Then, the last five vertices of L are checked for reducibility. If a reduction is done, then again the last five vertices of L are checked for reducibility. When no reduction is done, then the traversal proceeds to the next vertex. The process is continued until the traversal reaches the start vertex, which is the terminating condition of the algorithm. At the termination of the algorithm, the list L contains the list of the vertices of the orthogonal hull in order.

As explained in Sec. 3.3.1 and Sec. 3.3.2, each reduction rule except **R21** and **R23** produces an output that contains no consecutive **3**s. The output of the Rules **R21** or **R23** has a trailing pattern of **33**. This is removed when the next vertex is visited and the appropriate reduction rule is applied again. Figure 3.6 demonstrates the algorithm and shows how the concave regions are removed in successive iterations.

3.4 Algorithm for Constructing the Orthogonal Hull

The algorithm Ortho-Hull that outputs the ordered list of vertices of the orthogonal hull corresponding to a digital object, is shown in Fig. 3.7. It takes the digital object S, the grid size g, and the start vertex v_s as input parameters. The type (t) and the direction (d) from the start vertex is determined by a row-wise scan of the grid points, as explained in Sec. 3.2.1. The list L is initialized with a dummy vertex, v_d. The length of the edge from v_d is trivially set to zero, and the type and the direction of traversal from v_d have no significance in our algorithm. The dummy vertex is required because a reduction rule is always applied on a sequence of five vertices. A situation may arise when v_s, which is always of Type **1**, is followed by two consecutive Type **3** vertices, and one Type **1** vertex, leading to a pattern **1331** and $l_l > l_3$. Then the reduction Rule **R12** has to be applied (on five consecutive vertices), which requires a dummy vertex. In each iteration of the **do-while** loop (Steps 3–14), the next vertex is evaluated and appended to L. If the list L contains more than five vertices (Step 6), then the pattern formed by the types of the last four vertices in L is checked. The vertices v_4, v_3, v_2, v_1, and v_0, as mentioned in Sec. 3.3.1, correspond to the vertices represented by $L[k]$, $L[k-1]$, $L[k-2]$, $L[k-3]$, and $L[k-4]$ respectively, where $L[k]$ is the most recently visited vertex. If the pattern due to types of $L[k] \dots L[k-3]$ matches **1331**, then the reduction rule is applied by calling the procedure Apply-R1 (Steps 7–8). The procedure Apply-R1 returns the updated value of k due to reduction. Similarly, if the pattern matches **1333**, then the corresponding reduction rules (Apply-R2) are used (Steps 9–10). Apply-R2 returns the modified k and $\langle i, j\rangle$. If a reduction is done, the algorithm continues to evaluate the pattern of last four vertices after reduction, in the **while** loop (Steps 5–11), to check if further reductions are possible. Otherwise, when no rules are applicable (Step 11), it comes out of the **while** loop and continues to evaluate the next vertex. The variable k always points to the last vertex appended to L and is incremented with the visit of each vertex. The procedure Next-Vertex computes the length corresponding to $L[k]$. It also computes the direction d and the type t of the next vertex. The value of k is incremented in Step 12, and d and t are assigned to the incremented $L[k]$. The algorithm terminates (Step 14) when the coordinates $\langle i, j\rangle$ coincides with $\langle i_s, j_s\rangle$.

In the procedure Next-Vertex, in the **while** loop (Steps 2–11), the next vertex from the current vertex (i, j) is evaluated. The coordinates of the next grid point is determined in Step 3. Steps 4–10 determine whether the grid point is a vertex (Type **1** or **3**) or an edge point. The type of a grid point is determined by the object-intersecting quadrants incident at that grid point (Steps 4–6), as explained in Sec. 3.2.1. The algorithm continues to evaluate the subsequent grid points until a vertex ($L[k+1]$) is reached. As it proceeds to the next grid point, the length is incremented by g (Step 10). The procedure returns the type and the direction of the vertex $L[k+1]$, which it has traversed to, and the length l from the current vertex ($L[k]$) to the next vertex ($L[k+1]$). The coordinates of the next grid point are computed (Step 3) based on the direction d from (i, j). If the direction of traversal is towards right ($d = 0$) or left ($d = 2$), then the y-coordinate remains unchanged

```
**Algorithm** ORTHO-HULL $(S, g, v_s)$
**Steps:**
1.  $L[1] \leftarrow v_d, k \leftarrow 2$
2.  $L[k].d = d_s, L[k].t = t_s, \langle i, j\rangle \leftarrow \langle i_s, j_s\rangle$
3.  do
4.      $\langle (d, t, L[k].l), i, j\rangle$
            $\leftarrow$ NEXT-VERTEX $(S, i, j, L[k].d, g)$
5.      while (TRUE)
6.          if $(k \geqslant 5)$
7.              if $(L[(k-3)..k].t = \mathbf{1331})$
8.                  then $k \leftarrow$ APPLY-R1$(L, k)$
9.              else if $(L[(k-3)..k].t = \mathbf{1333})$
10.                 then $\langle k, i, j\rangle \leftarrow$ APPLY-R2$(L, k, i, j, g)$
11.             else break
12.     $k \leftarrow k + 1$
13.     $L[k].d \leftarrow d, L[k].t \leftarrow t$
14. while $(\langle i, j\rangle \neq \langle i_s, j_s\rangle)$
```

```
**Procedure** NEXT-VERTEX $(S, i, j, d, g)$
**Steps:**
1.  $m \leftarrow 0, r \leftarrow 0, l \leftarrow 0$
2.  while (TRUE)
3.      $(i, j) \leftarrow d \circledast (i, j)$
4.      for $h \leftarrow 1$ to 4
5.          if $Q_h(i, j) \cap S \neq \emptyset$
6.              $m \leftarrow m + 1,\ r \leftarrow r + h$
7.      if $r \in \{4, 6\}$ and $m = 2$ then $t \leftarrow 3$
8.      else if $m \in \{0, 2, 4\}$ then $t \leftarrow 0$
9.      else $t \leftarrow m$
10.     $d \leftarrow (t + d) \mod 4, l \leftarrow l + g$
11.     if $t \neq 0$ then break
12. return $\langle (d, t, l), i, j\rangle$
```

Figure 3.7: The algorithm ORTHO-HULL that uses the procedure NEXT-VERTEX and the reduction Rules $R1$ and $R2$ (respective procedures in Fig. 3.8 and Fig. 3.9).

```
Procedure APPLY-R1 $(L, k)$
Steps:
1.  if $(L[k-3].l = L[k-1].l)$
2.      then $L[k-4].l \leftarrow L[k-4].l + L[k-2].l + L[k].l$
3.           $k \leftarrow k-4$
4.  else if $(L[k-3].l > L[k-1].l)$
5.      then $L[k-3].l \leftarrow L[k-3].l - L[k-1].l$
6.           $L[k-2].l \leftarrow L[k-2].l + L[k].l$
7.           $k \leftarrow k-2$
8.  else
9.           $L[k-1].l \leftarrow L[k-1].l - L[k-3].l$
10.          $L[k-4].l \leftarrow L[k-4].l + L[k-2].l$
11.          $L[k-2] \leftarrow L[k]$
12.          $L[k-3] \leftarrow L[k-1]$
13.          $k \leftarrow k-2$
14. return $k$
```

Figure 3.8: The procedure to remove the concavity formed by the vertex pattern **1331**

and the x-coordinate is incremented ($d = 0$) or decremented ($d = 2$) by g; and if the direction is upwards ($d = 1$) or downwards ($d = 3$), then the y-coordinate is decremented ($d = 1$) or incremented ($d = 3$) while the x-coordinate remains unchanged. In other words, for $d = 0$, $\langle i, j\rangle \leftarrow \langle i, j+g\rangle$; for $d = 2$, $\langle i, j\rangle \leftarrow \langle i, j-g\rangle$; for $d = 1$, $\langle i, j\rangle \leftarrow \langle i+g, j\rangle$; and for $d = 3$, $\langle i, j\rangle \leftarrow \langle i-g, j\rangle$. The above transformations are denoted by $(i, j) = d \circledast (i, j)$ in Step 3 for notational simplicity. The subsequent direction of traversal is determined in Step 10.

In procedure APPLY-R1, if the lengths of the vertices $L[k-3]$ and $L[k-1]$ are same (Step 1), then the length of $L[k-4]$ is modified as shown in Step 2. The tail of the list is reset to $k-4$ (Step 3) as the last four vertices are removed according to the Rule **R11**. Similarly, Rules **R12** and **R13**, are implemented in Steps 5–7 and Steps 8–13, respectively. While applying the Rule **R13**, as vertices $L[k-3]$ (v_1) and $L[k-2]$ (v_2) are removed, the vertices $L[k-1]$ (v_3) and $L[k]$ (v_4) are reassigned as $L[k-3]$ and $L[k-2]$ respectively (Steps 11–12). The tail of the list, k, is reset to $k-2$, as two vertices have been removed (Step 13). The value of k is reset to $k-4$ (Step 3) and $k-2$ (Step 7) respectively for Rules **R11** and **R12**. The procedure returns the updated value of k in Step 14.

```
Procedure Apply-R2 $(L, k, i, j, g)$
Steps:
1. if $(L[k-3].l < L[k-1].l)$
2.   then $L[k-1].l \leftarrow L[k-1].l - L[k-3].l$
3.        $L[k-4].l \leftarrow L[k-4].l + L[k-2].l$
4.        $L[k-2] \leftarrow L[k]$
5.        $L[k-3] \leftarrow L[k-1]$
6.        $k \leftarrow k-2$
7. else $\triangleright$ $L[k-3].l \geqslant L[k-1].l$
8.   then $l' \leftarrow L[k-3].l - L[k-1].l$, $l'' \leftarrow L[k].l$
9.     $d \leftarrow L[k].d$
10.    while $(l' \geqslant L[k-3].l - L[k-1].l$ or $l'' \geqslant L[k-2].l)$
11.      $\langle (d,t,l), i, j\rangle \leftarrow$ Next-Vertex $(S, i, j, d, g)$
12.      if $(d = d_{k-3})$ then $l' \leftarrow l' + l$
13.      else if $(d = d_{k-1})$, then $l' \leftarrow l' - l$
14.      if $(d = d_k)$ then $l'' \leftarrow l'' + l$
15.      else if $(d = d_{k-2})$ then $l'' \leftarrow l'' - l$
16.    if $(d = d_{k-2})$
17.      then $L[k-2].l \leftarrow L[k-2].l - l''$
18.           $L[k-3].l \leftarrow l'$
19.           $k \leftarrow k-2$
20.    else if $(d = d_{k-1})$
21.      then $L[k-3].l \leftarrow L[k-3].l - L[k-1].l$
22.           $L[k-2].l \leftarrow L[k-2].l - l''$
23.           $L[k-1].l \leftarrow L[k-3].l - L[k-1].l - l'$
24.           $k \leftarrow k-1$
25.return $\langle k, i, j\rangle$
```

Figure 3.9: The procedure to remove the concavity formed by the vertex pattern **1333**

The procedure Apply-R2 implements the second set of rules corresponding to the pattern **1333**. Rule **R21**, applied when $l_1 < l_3$, is implemented in Steps 1–6. Otherwise, when $l_1 \geqslant l_3$ (Step 7), two variables l' and l'' are initialized in Step 8. The **while** loop (Steps 10–15) is repeated as long as the condition $l' \geqslant l_1 - l_3$ or the condition $l'' \geqslant l_2$, is satisfied. In the **while** loop, the next vertex is visited and the values of l' and l'' are modified depending upon the direction of traversal. If the direction of traversal is same as the direction of traversal from $L[k-3]$, then l' is updated to $l'+l$; if it is same as that of $L[k-1]$, then l' is updated to $l'-l$ (Steps 12–13). Similarly, l'' is also modified (Steps 14–15). Thus, each time a new vertex is visited, the parameters l' and l'' are modified. When it comes out of the **while** loop, two different rules, **R22** and **R23**, are applied depending upon the current direction of traversal. If this direction of traversal is same as that from $L[k-2]$ (Step 16), then **R22** is applied (Steps 17–19). When the direction is same as that of $L[k-1]$, Rule **R23** is applied (Steps 20–24). The value of k is readjusted to $k-2$ (Step 19) for Rule **R22**, and to $k-1$ (Step 24) for Rule **R23**. The procedure returns the updated k and $\langle i,j\rangle$ (Step 25).

3.4.1 Proof of Correctness

The proof of correctness of the algorithm is based on the fact that the orthogonal hull produced by the algorithm is of minimum area with each object point strictly lying inside it, and the intersection of the hull with any horizontal or vertical line is either empty or exactly one line segment (Defn. 3.2.1).

To show that the orthogonal hull produced by the algorithm ORTHO-HULL is of minimum area, we first observe that if p is a point lying on the grid line and lying left while traversing around the object S, then $0 < d_\top(p,S) \leqslant g$[1]. That $d_\top(p,S) > 0$ is evident from our policy of object containment in a cell of the grid, as explained in Sec. 3.2.1. To show that $d_\top(p,S) \leqslant g$, let, w.l.o.g., p be any point on the horizontal grid line-segment common to Q_1 and Q_4 (Fig. 3.2). Let, for contradiction, $d_\top(p,S) = h > g$. For any point p' in Q_1 or Q_4, $d_\top(p,p') \leq g$. Since $h > g$, neither Q_1 nor Q_4 has any object containment. Hence, the traversal traces the grid-line segment common to Q_3 and Q_4, and then traces either the grid line-segment common to Q_2 and Q_3 (90^0 vertex, Fig. 3.2(a)) or the grid line-segment common to Q_1 and Q_2 (180^0). In either case, the grid-line segment common to Q_1 and Q_4 (and p, there of) lies to the right during the traversal around S — a contradiction.

Thus, each cell lying left of the traversed path around the object contour has object containment, and each free cell (without object containment) lies right of the traversed path. Free cells are included in the orthogonal hull only when the reduction rules are applied. It is evident from the rules (Sec. 3.3) and their related figures (Figs. 3.4 and 3.5)

[1]Here, $d_\top(p,q) = \max\{|i_p - i_q|, |j_p - j_q|\}$ denotes the *(orthogonal) distance between two points*, $p(i_p, j_p)$ and $q(i_q, j_q)$ (i.e., the Minkowski norm L_∞ [Klette and Rosenfeld (2004a)]). The *distance of a point p from the object S* is $d_\top(p,S) = \min\{d_\top(p,q) : q \in S\}$.

that the free cells included in the orthogonal hull are minimum in number. For, if any of these free cells (making a non-convex region) is not included, then it would create either a hole inside the hull or two consecutive vertices of Type **3**. This completes the proof that the orthogonal is of minimum area.

Now, to show that the intersection of the orthogonal hull with any horizontal or vertical (grid or non-grid) line is either empty or a single line segment, we simply observe that the final polygon does not contain two consecutive vertices of Type **3**. In accordance with the concavity-removal rules, it is evident that, excepting the Rules **R21** and Rule **R23**, the outcome of each rule contains no two consecutive vertices of Type **3** (Sec. 3.3.2). The outcome of **R21** or **R23** has a pattern **33** at the end, which is removed depending on the type of the next vertex. In the final iteration, the last vertex visited coincides with the start vertex, which is of Type **1**, and hence, the applied rule is one among **R11**, **R12**, and **R13**. Therefore, on termination, we get no two consecutive vertices of Type **3**.

3.4.2 Demonstration

A step-by-step demonstration of the algorithm is shown in Fig. 3.10, which illustrates how the different rules are applied for removal of some typical concave regions from a given digital object as the orthogonal traversal proceeds closely around the object contour without touching it. On the left hand side, the list L is shown as it grows with the orthogonal traversal or as it shrinks as a result of applying the reduction rules. The start vertex, u_1, is determined by a row-wise scan of the grid points. Note that the vertices are indicated in the figure by their indices. For simplicity of the representation, the dummy vertex is not shown here in L. The leftmost column of the table indicates the vertices as they are visited and associated list is shown to its right. Only the types of the vertices are shown in the list L. The first non-convex region is detected when u_6 is visited. On visiting u_6, the sequence of types of the last four vertices in L matches the pattern **1331** and l_3 (the length corresponding to u_3) equals l_5 (of u_5), and hence Rule **R11** (Fig. 3.4) is applied. As a result, all four vertices, namely u_3, u_4, u_5, and u_6, are removed, and the length of u_2 (equivalent to v_0 in Fig. 3.4) is modified to $l_2 + l_4 + l_6$. As the traversal continues, the next non-convex region is detected when u_9 is visited. On visiting u_9, the sequence of types of the last four vertices matches the pattern **1333**, and $l_2 \geqslant l_8$. Note that the length l_2 has been modified to $l_2 + l_4 + l_6$ in the last reduction. So, the two variables l' and l'' are initialized as $l_2 - l_8$ and l_9 respectively. When u_{10} is visited, l' is modified to $l_2 - l_8 - l_{10}$ and l'' is not changed. As the second condition in the **while** loop

u_1	1												
u_2	1	1											
u_3	1	1	1										
u_4	1	1	1	3									
u_5	1	1	1	3	3								
u_6	1	1	1	3	3	1	▷R11						
	1	1											
u_7	1	1	3										
u_8	1	1	3	3									
u_9	1	1	3	3	3	▷R22							
	1	1	3	(u_{10} and u_{11})									
u_{12}	1	1	3	3									
u_{13}	1	1	3	3	1	▷R13							
	1	3	1										
u_{14}	1	3	1	1									
u_{15}	1	3	1	1	3								
u_{16}	1	3	1	1	3	3							
u_{17}	1	3	1	1	3	3	3	▷R23					
	1	3	1	1	3	3	(u_{18}, u_{19} and u_{20})						
u_{21}	1	3	1	1	3	3	1	▷R12					
	1	3	1	1	3								
u_{22}	1	3	1	1	3	1							
u_{23}	1	3	1	1	3	1	1						
u_{24}	1	3	1	1	3	1	1	1					
u_{25}	1	3	1	1	3	1	1	1	1				
u_{26}	1	3	1	1	3	1	1	1	1	3			
u_{27}	1	3	1	1	3	1	1	1	1	3	3		
u_{28}	1	3	1	1	3	1	1	1	1	3	3	3	▷R21
	1	3	1	1	3	1	1	1	3	3			
u_{29}	1	3	1	1	3	1	1	1	3	3	1	▷R12	
	1	3	1	1	3	1	1	1	3				
u_{30}	1	3	1	1	3	1	1	1	3	1			

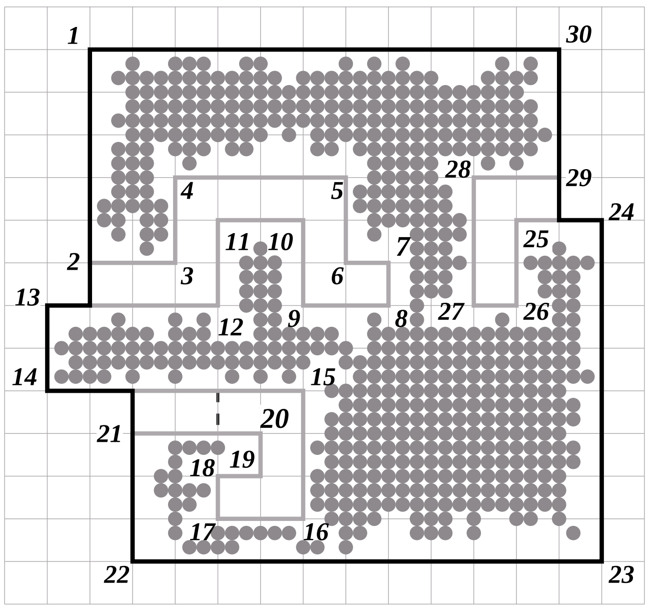


Figure 3.10: A step-by-step demonstration of the proposed algorithm on a digital object (shown in right). Each vertex u_i of the orthogonal polygon (right) has been labeled as 'i' for sake of simplicity. For each iteration of the algorithm, the last five vertices in the list L (left) have been highlighted in red before a rule is applied and in green after two consecutive vertices of Type **3** are removed by the rule.

in Step 9 of the procedure APPLY-R2 (Fig. 3.9) still holds, u_{10} is discarded (not added to the list L) and the next vertex u_{11} is visited. Now, l'' is modified to $l_9 - l_{11}$, and none of the conditions in the **while** loop holds true, and so the algorithm comes out of the

while loop. As the direction of traversal of u_{11} is same as that of u_7, the Rule **R22** is applied. The list L is left with a pattern **113**. Subsequently, when vertex u_{13} is visited, the Rule **R13** is applied. In the course of traversal, when u_{17} is reached, the procedure Apply-R2 is called as the sequence of last four vertices in L matches **1333**. The vertices u_{18} and u_{19} are visited but not added to L as one of the conditions in the **while** loop is still valid. When u_{20} is visited, none of the conditions in the **while** loop is valid, and as d_{20} equals d_{16}, reduction Rule **R23** is applied. Subsequently, Rule **R21** is applied at u_{28} and Rule **R12** at u_{29}. When the traversal finally reaches u_1, the list L contains the vertices of the orthogonal hull of the given digital object in order.

3.4.3 Time Complexity

Since the object is defined as a connected component, the containment of the object in a cell incident at a grid point q is verified from the intersection of the object with the four edges of the corresponding cell. For each edge, the intersection can be checked in $O(g)$ time, where g is the grid size. Hence, checking the object containment in any cell can be done in $4 \times O(g) = O(g)$ time.

During traversal of the grid points lying immediately outside the object contour, we visit each grid point q_i from the preceding one, q_{i-1}, using the information on intersection of the object with the edges incident at q_{i-1}. For example (Fig. 3.2(a)), if q_{i-1} is of Type **1** and has been visited along the vertical edge (from its predecessor, q_{i-2}), then q_i is visited along the horizontal edge from q_{i-1}. Thus, the number of grid points visited while traversing orthogonally along the object contour is bounded by $O(n/g)$, where n is the number of points constituting the object contour. The resultant time complexity for visiting all the vertices is, therefore, given by $O(n/g) \cdot O(g) = O(n)$. Note that, each grid point lying on the orthogonal path of traversal is visited either once (Fig. 3.2(a, b, d)) or twice (Fig. 3.2(c)). The time spent over detection and removal of concavities is associated with checking a pattern (of types of the last four vertices) in the list L in $O(1)$ time and applying the reduction rule, whenever necessary. If the pattern does not contain two consecutive **3**s, a new vertex is visited in $O(g)$ time. If the pattern undergoes a reduction, which needs $O(1)$ time, then also the next vertex is visited. Maximum number of reductions is bounded by $O(n/g) - 4$, since at most $O(n/g)$ vertices are visited and the orthogonal hull consists of at least four vertices. Thus, the total number of list operations is bounded by $(O(n/g) - 4) \cdot O(1) = O(n/g)$. Hence, the total time complexity for finding the orthogonal hull of a digital object is given by $O(n) + O(n/g) = O(n)$. This improves the earlier time complexity [Karlsson and Overmars (1988)].

3.5 Experimental Results

The proposed algorithm is implemented in C on a Sun_Ultra 5_10, Sparc, 233 MHz, the OS being the SunOS Release 5.7 Generic, and has been tested on (i) database D1 containing 1034 logo images (received on request, from Prof. Anil K. Jain and Aditya Vailya of Michigan State Univ., USA); (ii) a collected database D2 having 520 shape images; (iii) some geometric shapes; (iv) a selected database of optical characters; (v) a set of gray-scale images.

Figure 3.11 shows the result on an image of 'dragon' for $g = 8$. The orthogonal hull is shown in blue, whereas the red colored polygon indicates the smallest-area orthogonal polygon describing the object, which is traversed during the derivation of the orthogonal hull. The interior region of the orthogonal hull has been highlighted in green. The regions enclosed by the orthogonal hull (blue lines) and the isothetic polygon (red lines) show the non-convex portions detected by the algorithm. The CPU time required for computation of the convex hull for the 'dragon' image, having size 1000×1000 and consisting of 89744 object pixels, is 8.61 seconds, which is significantly high compared to 34.29 milliseconds required for the computation of its orthogonal hull for $g = 8$ (shown in Table 3.1). The convex hull is computed using the Graham scan algorithm. The time required for computation of the convex hull is very high compared to orthogonal hulls because of the fact that the computation of an orthogonal hull is based on the orthogonal traversal around the boundary of the object, and it requires only comparison and addition/subtraction in the integer domain. The time required by Graham scan algorithm to compute the convex hull could be reduced by considering the boundary points of the object, but that requires an edge detection algorithm.

In Table 3.1, the area of the convex hull and the areas of the orthogonal hulls for different grid sizes (4, 8, and 14) are presented. The number of convex hull vertices, the CPU time required for computation of the convex hull, the number of vertices of an orthogonal hull, and the CPU times at different grid sizes are also presented in the table. It can be seen from this table that the number of vertices of $OH(S)$ decreases and its area increases with the increase of grid size. Also, the computation time drops appreciably for higher grid sizes. Apart from the 'dragon' image, Table 3.1 also presents the results corresponding to four logo images, two OCR images ('P' and 'R'), and one geometric shape ('swastik').

Figure 3.14 shows four logo images and their corresponding orthogonal hulls for grid size $g = 4, 8$, and 14. The non-convex regions have been filled in yellow to depict the result

Table 3.1: Comparison of experimental results on convex hulls and orthogonal hulls.

image name and size	#pixels	$\|CH\|$	$A(CH)$	$T(CH)$	$\|OH\|$			$A(OH)$			$T(OH)$		
					$g = 4$	8	14	$g = 4$	8	14	$g = 4$	8	14
logo245 288 × 288	9868	38	39509	682	86	48	26	35681	37929	39887	3.24	2.78	2.63
logo247 288 × 288	10096	25	32419	698	66	28	14	29929	31857	33349	2.08	1.83	1.88
logo353 288 × 288	18122	33	32241	1375	120	60	32	31117	32593	35953	3.68	2.88	2.73
logo354 288 × 288	7179	33	39237	263	88	44	42	35835	37025	41469	3.40	2.91	2.62
logo415 288 × 288	13590	21	29131	894	98	56	40	23149	24601	27735	3.33	2.81	2.544
dragon 1000 × 1000	89744	37	564925	8610	314	168	94	526961	533409	546449	39.10	34.29	33.33
‘R’ 288 × 288	26369	12	35141	1087	16	12	8	37237	39441	43149	3.62	2.83	2.70
‘P’ 288 × 288	22320	24	28576	472	40	20	18	27833	28945	33741	3.28	2.72	2.58
swastik 442 × 442	24174	11	35069	3910	204	108	60	35825	38177	41329	10.25	7.81	6.82

$|CH|$ and $|OH|$ denote the respective number of convex hull vertices and number of orthogonal hull vertices; $A(CH)$ and $A(OH)$ are their respective areas; and $T(\cdot)$ indicates the CPU time in milliseconds.

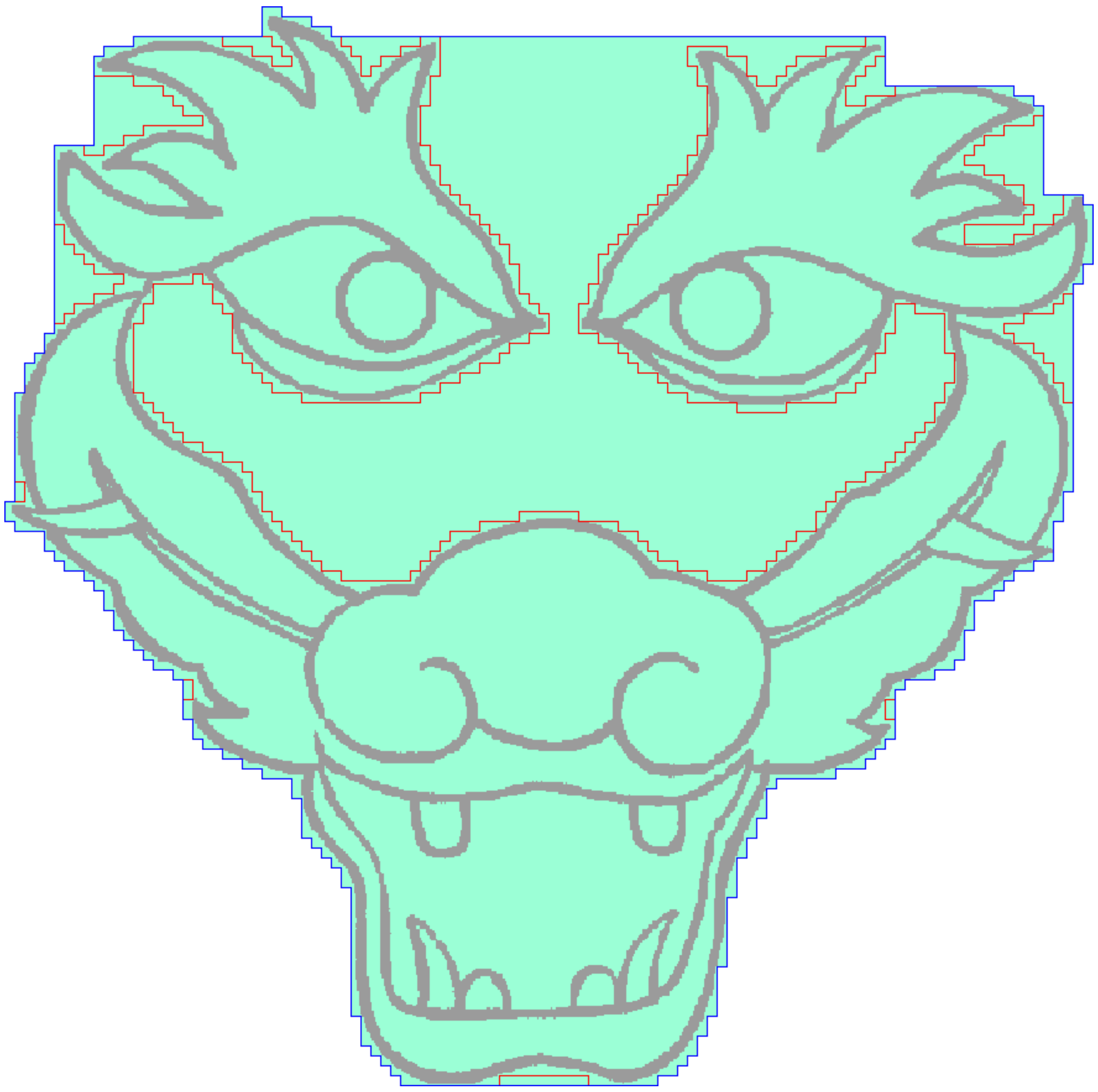
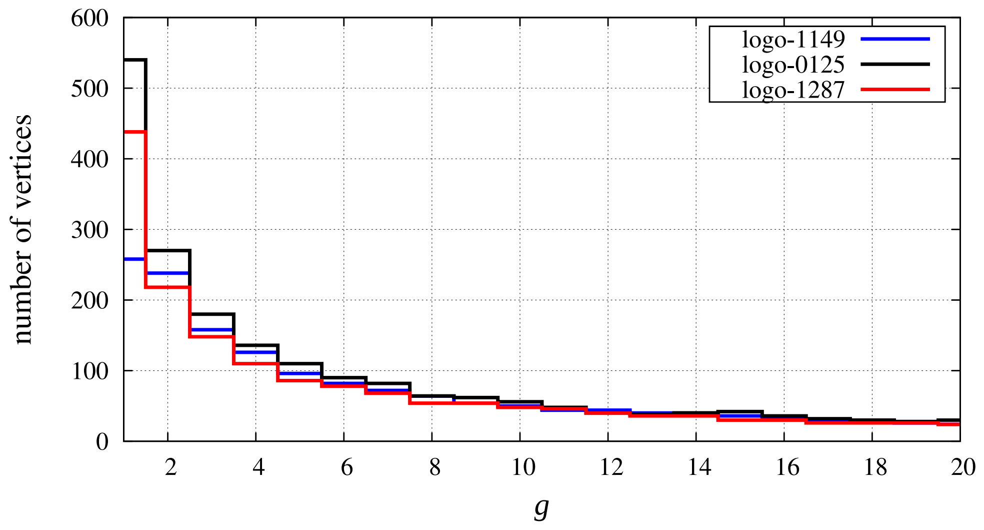
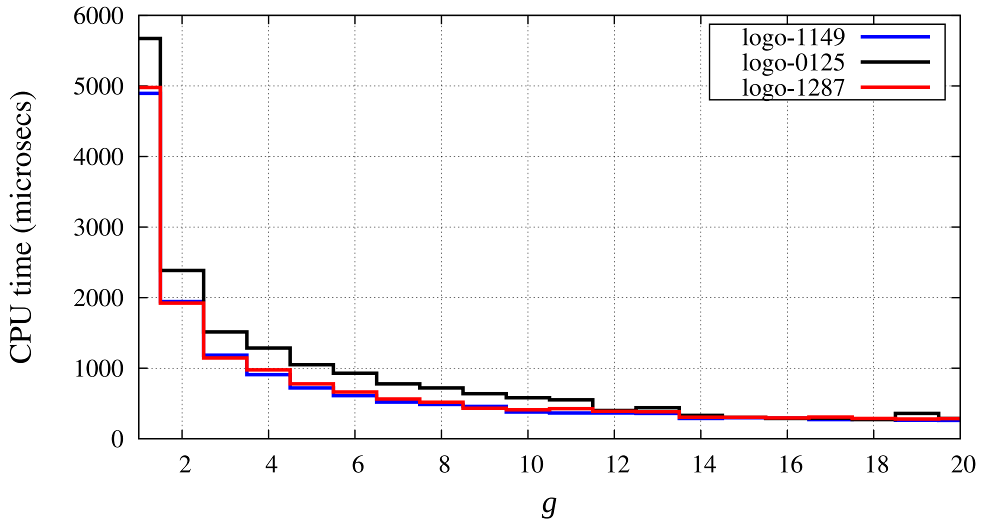


Figure 3.11: Orthogonal hull (the edges shown in blue and the interior in green) of the binary image 'dragon' for $g = 8$.

of concavity-reduction rules. First column of each row indicates the number of object pixels, and each image is accompanied with the number of vertices and the orthogonal hull area (shown in two boxes) for the corresponding grid size. It is seen that the number of vertices gets decreased and the hull area gets increased with the increase of g. The number of vertices is plotted against the grid size (g = 1 to 20) in Fig. 3.12 (a) for



(a) No. of vertices vs. grid size.



(b) CPU Time vs. grid size.

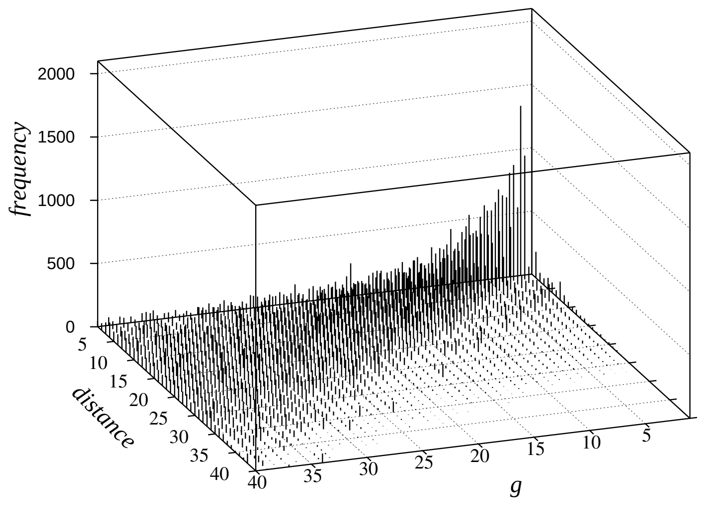
Figure 3.12: (a) Shows that the number of vertices of the orthogonal hull decreases with the increasing grid size. (b) The CPU time required for computation of the orthogonal hull decreases with the increase of grid size.

three different logo images. From these plots, it is evident that the number of vertices rapidly decreases with the increase of grid size. Figure 3.12 (b) shows the way the CPU time decreases with increasing g for the same three images. The orthogonal hulls of all four logo images in Fig. 3.14 reflect the symmetric nature of the objects. The symmetric nature of the objects remains almost unchanged with a change in the grid size. Figure 3.15 shows another set of four logo images in which, excepting the first image ('logo119'), all images are asymmetric. The similar nature of the last three objects ('logo353', 'logo354', and 'logo355') are captured by their orthogonal hulls. Figure 3.16 shows four geometric shapes and their orthogonal hulls for various grid sizes. It is seen that the algorithm detects all the non-convex regions and extracts the orthogonal hulls correctly. The 'spiral' image has a more convoluted shape, and the algorithm successfully constructs the orthogonal hull. It may be noted that, for all these images, the complexity (number of vertices) of the orthogonal hull reduces with the increasing grid size. Similar to the images in Fig. 3.14, the orthogonal hulls of the first two images have also rotational symmetry, which remains almost invariant with a change in the grid size.

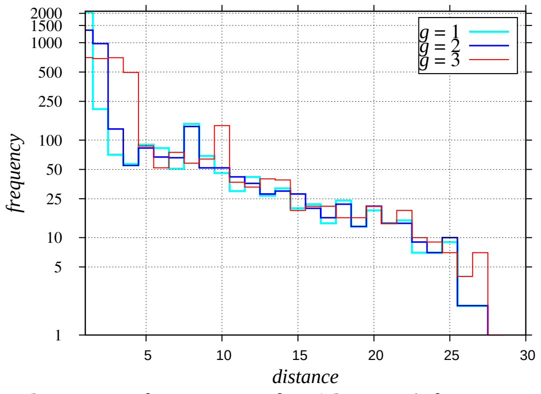
The orthogonal hulls of four optical characters are shown in Fig. 3.17. The optical characters can be classified into different sets depending upon their orthogonal hulls. A clear demarcation can be made between different characters by using the non-convex regions (marked in yellow) in conjunction with orthogonal hulls. Figure 3.18 shows results on binarized impressions of some real-world gray-scale images. In Fig. 3.13, we have furnished a 3-D plot on the frequency of errors versus g and distances of points on the orthogonal hull from the object corresponding to the image 'dragon'. The error frequency for an orthogonal distance d is given by the number of points on (the border of) $OH(S)$ for which the nearest object point is at the distance d. The other three plots (Fig. 3.13(b–d)) show the distribution of frequency versus distance for three grid sizes ($g = 1, 2, 3$) corresponding to the images 'dragon', 'logo355', and 'logo353' respectively. The frequency is shown in $\log_{10}$-scale. It is evident that the number of object points at a smaller distance is very high, whereas, the number of object points lying at a larger distance from OH(S) is low. However, this distribution is dependent on the shape of the object; if there are more concavities, then the distribution may be different, while the basic trend remains the same.

3.6 Conclusion

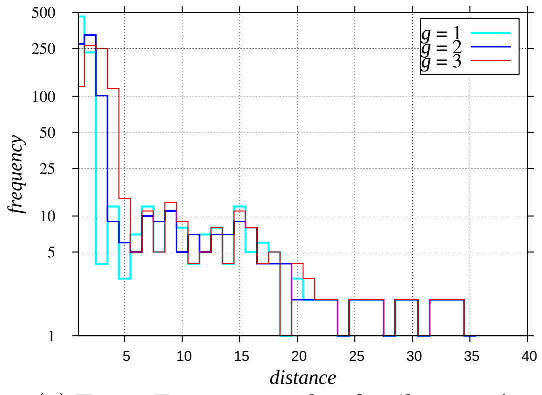
We have presented a combinatorial algorithm to construct the orthogonal convex hulls of a digital object for various grid resolutions. The worst-case time complexity of the algorithm is linear in the size of the contour. The actual runtimes on different images demonstrate its



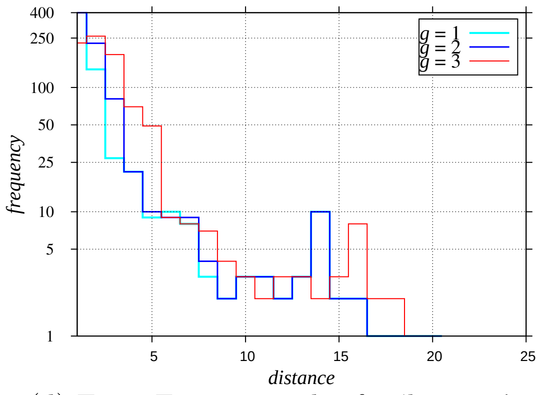
(a) Frequency of errors plotted against g corresponding to 'dragon'.



(b) Plot on the error frequency for 'dragon' for $g = 1, 2,$ and 3.



(c) Error Frequency plot for 'logo355'



(d) Error Frequency plot for 'logo355'

Figure 3.13: Plots on frequency of errors versus g for different images.

effectiveness for speedy execution. The algorithm is a single-pass algorithm, and outputs the (types and outgoing edge-lengths of) vertices of the orthogonal hull in order. Hence, we can utilize the orthogonal hull description in some suitable application like shape analysis, shape-based image retrieval, etc. For example, the number of times the reduction rules are applied during the traversal for removal of a concavity can be used to measure the distribution of shape complexity over a large and complex object. Also, the non-convex regions along with the orthogonal hull can be used for an appropriate shape analysis.

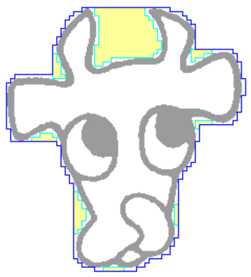
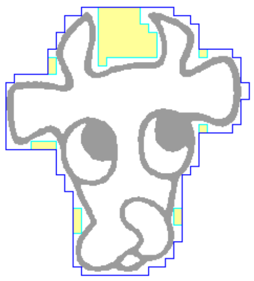
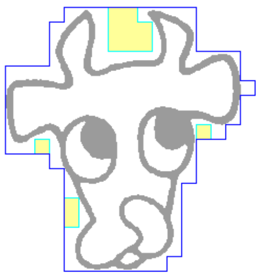
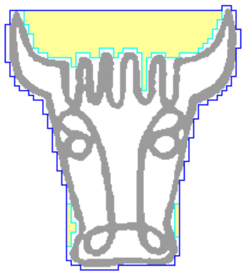
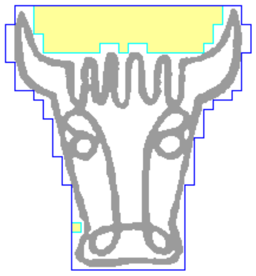
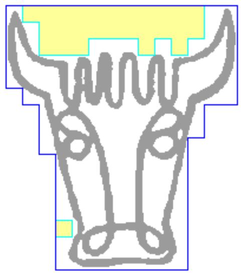
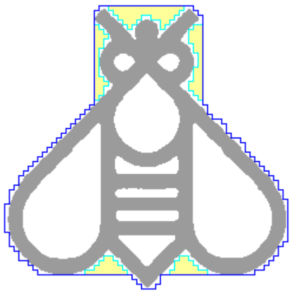
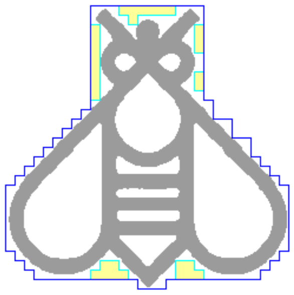
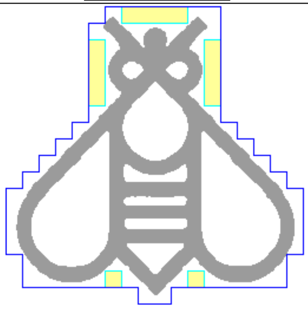
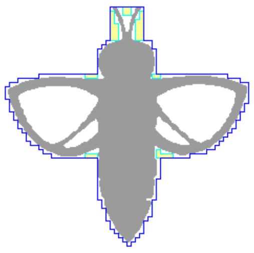
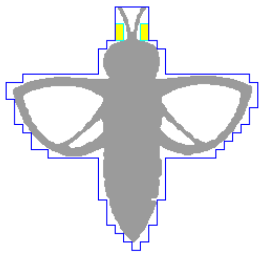
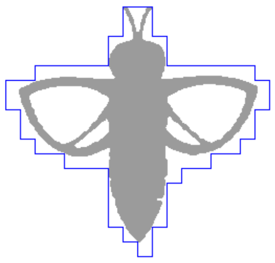
n	$g = 4$	$g = 8$	$g = 14$
logo245 9868	 86 | 35681	 48 | 37929	 26 | 39887
logo247 10096	 66 | 29921	 28 | 31857	 14 | 33349
logo416 16367	 114 | 36561	 52 | 38497	 34 | 42253
logo415 13590	 98 | 23149	 56 | 24601	 40 | 27735

Figure 3.14: The orthogonal hulls (edges shown in blue) of a set of logo images for grid size $g = 4, 8$, and 14. The non-convex regions detected by the algorithm are shown in yellow. The number of vertices of the orthogonal hull is shown in the left box and the area of the hull in the right box below the concerned image.

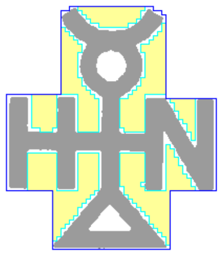
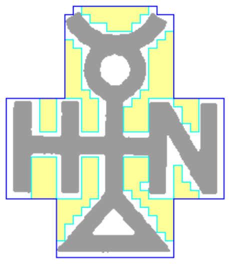
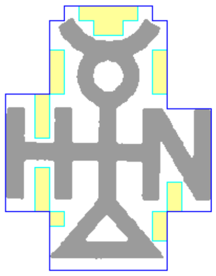
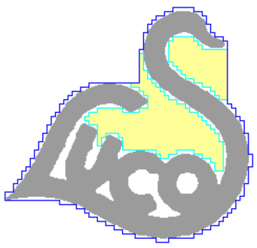
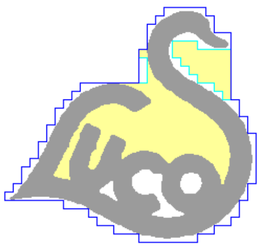
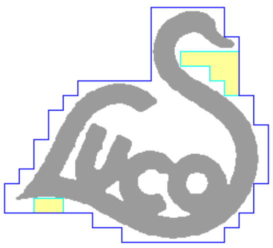
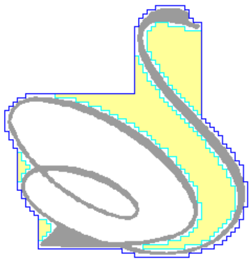
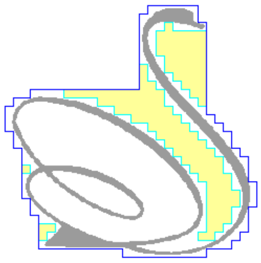
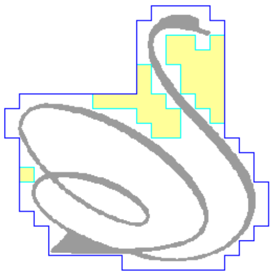
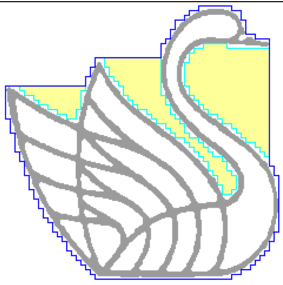
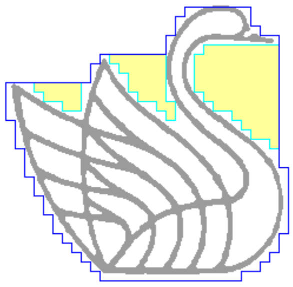
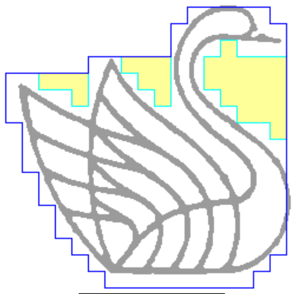
n	$g = 4$	$g = 8$	$g = 14$
logo119 16305	 18 33217	 16 35649	 16 37101
logo353 18122	 120 31117	 60 32593	 32 35953
logo354 7179	 88 35385	 44 37025	 32 41469
logo355 10541	 90 39665	 48 41625	 28 43793

Figure 3.15: The orthogonal hulls for another set of logo images corresponding to $g = 4$, 8, and 14 (color codes and numerical figures as in Fig. 3.14).

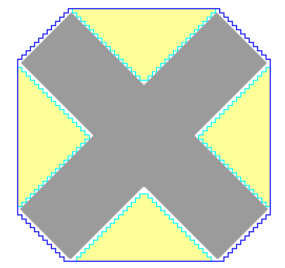
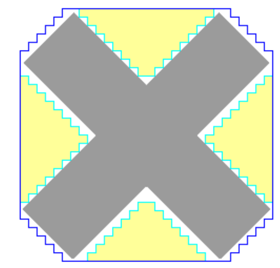
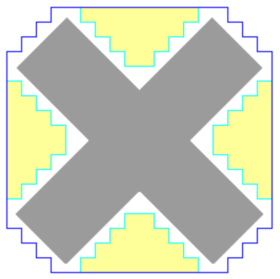
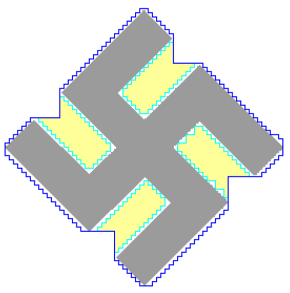
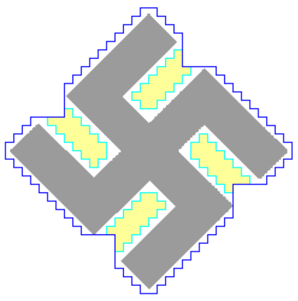
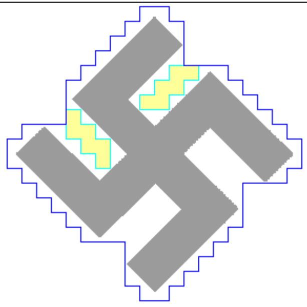
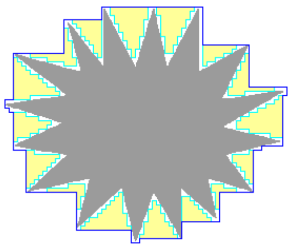
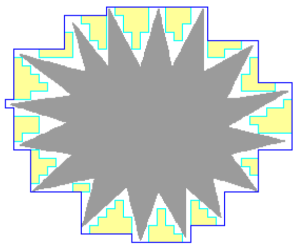
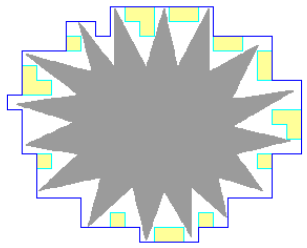
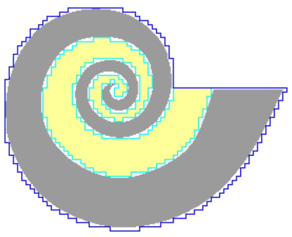
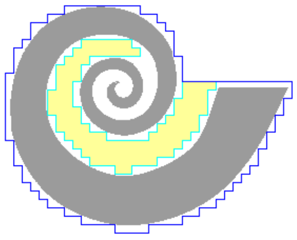
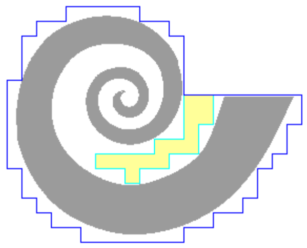
n	$g = 4$	$g = 8$	$g = 14$
32583	 96 53473	 44 54241	 28 59305
24174	 204 35825	 108 38177	 60 41329
28732	 50 42445	 34 44657	 30 46173
23776	 126 38033	 64 40233	 36 43429

Figure 3.16: The orthogonal hulls of a set of geometric shapes for $g = 4$, 8, and 14 (color codes and numerical figures as in Fig. 3.14).

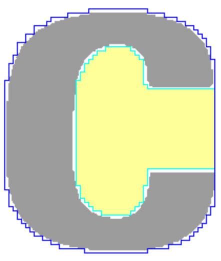
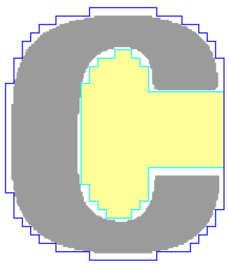
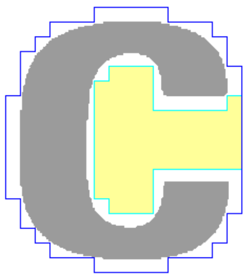
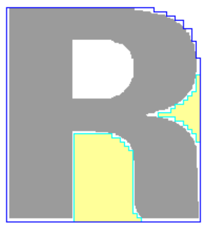
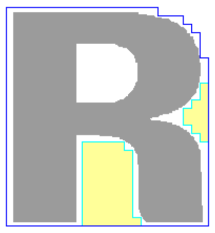
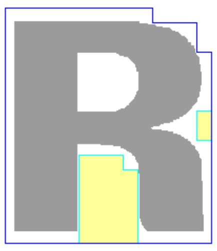
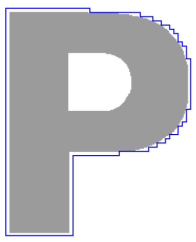
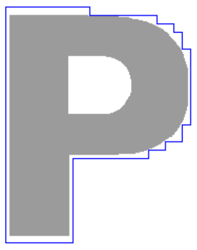
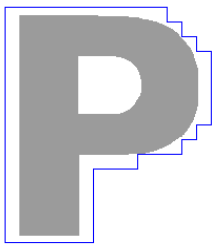
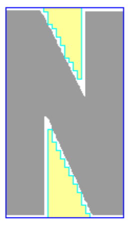
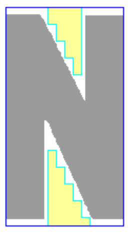
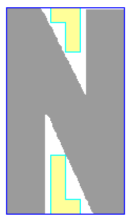
n	$g = 4$	$g = 8$	$g = 14$
24170	 80 41953	 48 44161	 32 48105
26369	 16 37237	 12 39441	 8 43149
22320	 40 27833	 20 28945	 18 33741
17592	 4 22713	 4 23617	 8 22261

Figure 3.17: Results on some optical characters (color codes and numerical figures as in Fig. 3.14).

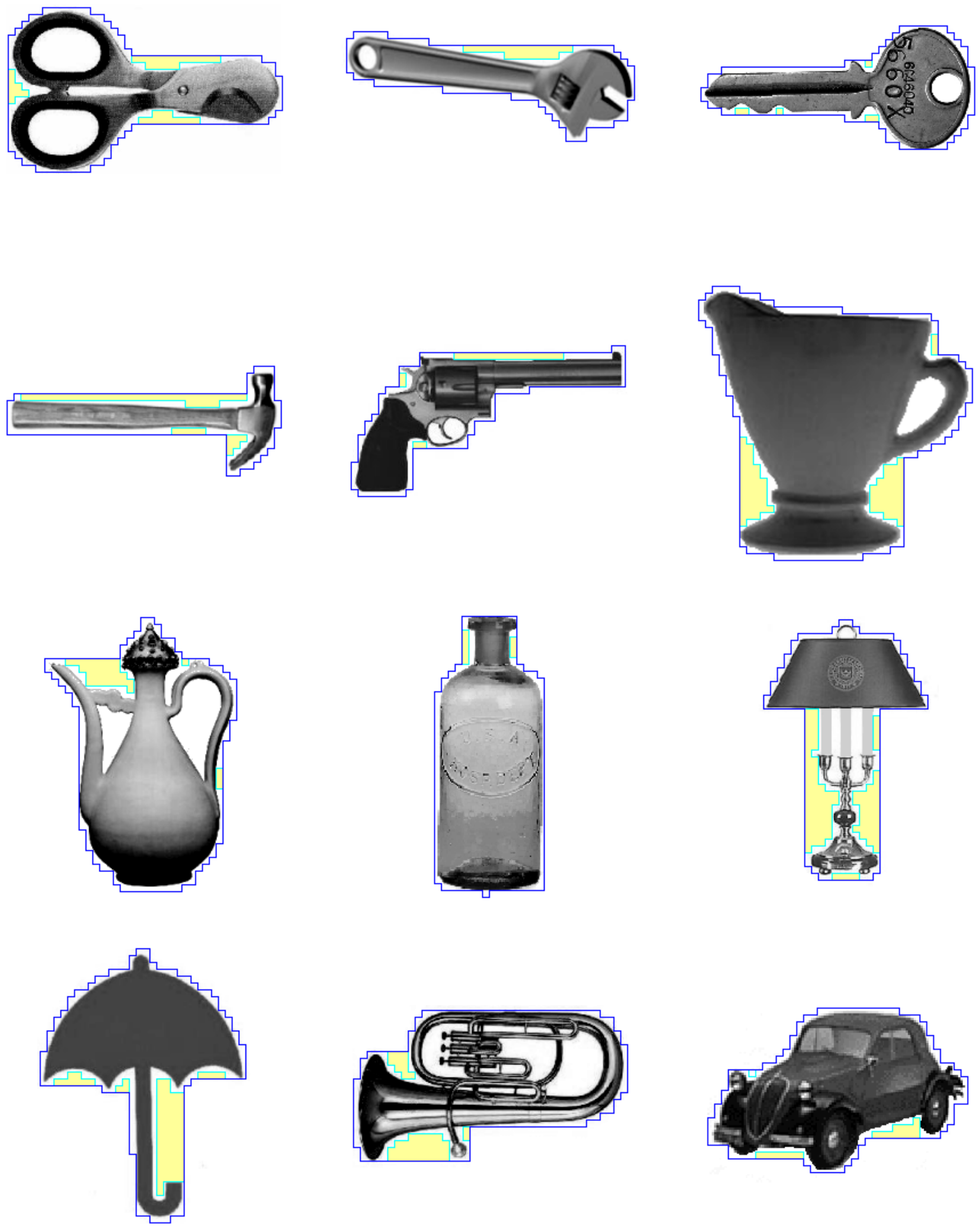


Figure 3.18: The orthogonal hulls (color codes and numerical figures as in Fig. 3.14) of the binarized images corresponding to some real-world gray-scale objects for $g = 6$.

Chapter 4

Shape Analysis using Isothetic Covers

4.1 Introduction

As mentioned in Chapter 2, isothetic covers may find several areas of application including image analysis. In this chapter, we present further applications of isothetic covers in shape analysis. An approximate shape of an object is captured by the isothetic cover from which a shape code is derived. The use of shape codes in representing an object and subsequent retrieval of similar images from a database is discussed in Sec. 4.2. The isothetic cover can also be used to find a measure of the shape complexity of an arbitrary object. Section 4.3 presents the derivation of a shape complexity measure. Section 4.4 presents a mechanism to rank optical character prototypes in a large database using isothetic chord lengths of the isothetic covers corresponding to these character prototypes. In Sec. 4.5, we have presented the experimental results corresponding to these applications.

4.2 Multigrid Shape Analysis using Shape Code

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) [Katsaggelos *et al.* (1998)], runlength encoding [Shuster and Katsaggelos (1998)], and polygon-based encoding [O'Conell (1997)] 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-distortion [Katsaggelos *et al.* (1998)], 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 [Mokhtarian and Mackworth (1992), Mokhtarian *et al.* (1997)], geometric moments [Hu (1962)], 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 [CCITT (1994), ISO (1992, 1999)] where an object is encoded by its support map, and the other being contour based (where the object boundary is represented efficiently), e.g., chain code [Freeman (1961a), Kaneko and Okudaira (1985)], polygonal approximation [Kaup and Heuer (2000), O'Conell (1997), Shuster and Katsaggelos (1998)], high-order curve fittings viz. splines [Katsaggelos *et al.* (1998)], combined polygon-spline approximation [Gerken (1994)], approximation of polytopes [Bemporad *et al.* (2004)], etc.

In this work, a fast and efficient shape coding technique based on outer isothetic cover describing the digital object on a multigrid background, is proposed. This method differs from other polygonal approximation methods [Kaup and Heuer (2000), O'Conell (1997), Pal and Mitra (2004b), Shuster and Katsaggelos (1998)] in the fact that it does not require the contour to be extracted before constructing the outer isothetic cover. 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. 4.1. For retrieval, at each finer level, the size of the effective search database is diminished, which favours encouraging matching results.

4.2.1 Proposed Work

We have shown in Chapter 2 how the outer isothetic cover (OIC) of a digital object, having more than one connected component, can be constructed. Each polygon of the OIC is uniquely described, starting from a suitable vertex (start vertex), and the corresponding higher level description of the polygons of the OIC is coded properly to generate a shape

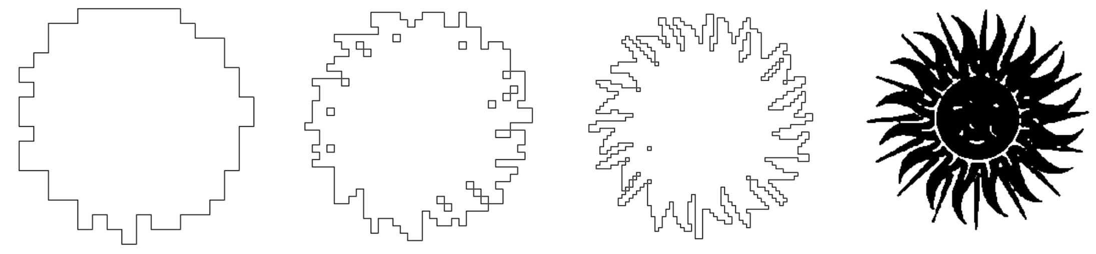


Figure 4.1: The isothetic polygons of an image are shown from left to right with increasingly finer resolution. The original image is shown in extreme right.

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 varying resolution environment turn out to be powerful features for the subsequent image retrieval process.

Given a binary image S containing one or more connected components[1], and a set of uniformly spaced horizontal and vertical grid lines, $\mathcal{G} = (\mathcal{H}, \mathcal{V})$ (Defn. 2.2.5), the corresponding outer isothetic cover, $\overline{P}(S) = \{P_i\}_{i=1}^{n}$, consists of n outer/outer hole polygons (Defn. 2.2.8). Let V_i represent the ordered set of vertices of P_i.

It may be noted that, two polygons in $\overline{P}(S)$ do not have any edge intersection, although one outer polygon may contain a outer hole 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. 4.2, the outer polygon P_1 contains the outer 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. It may be also noted that, if a connected component in S 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 occurring in an image is not perceivable in a coarse grid, then there would be no isothetic polygon in $\overline{P}(S)$ that would contain the object.

[1]We have considered 8-connectivity in this work.

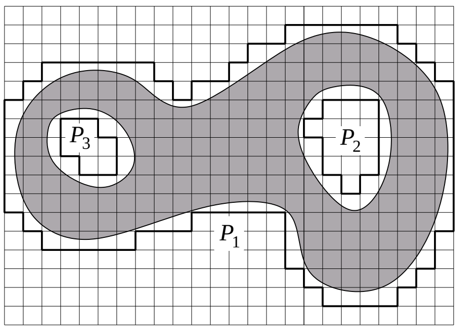


Figure 4.2: Outer isothetic cover consisting of an outer polygon, P_1, and two (outer) hole polygons, P_2 and P_3.

4.2.2 Shape Code

Let V_{ri} be the ordered set of vertices for the ith polygon P_{ri} in $\overline{P}(S_r)$ corresponding to the rth digital image S_r in a binary image database $\mathcal{D} = \{S_r\}_{r=1}^{N}$ 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:

$$\begin{aligned} SC(P_{ri}, g) = x[v_{ri}^{(1)}]y[v_{ri}^{(1)}] \quad & (b_8b_7)_{ri}^{(2)}|e(v_{ri}^{(1)}, v_{ri}^{(2)})| \\ & (b_8b_7)_{ri}^{(3)}|e(v_{ri}^{(2)}, v_{ri}^{(3)})| \cdots \\ & (b_8b_7)_{ri}^{(m_{ri})}|e(v_{ri}^{(m_{ri}-1)}, v_{ri}^{(m_{ri})})|, \end{aligned} \tag{4.1}$$

where, $(x[v_{ri}^{(1)}], y[v_{ri}^{(1)}])$ gives the start (first) vertex, $(b_8b_7)_{ri}^{(2)}$ 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.

4.2.3 Multigrid Shape Code

The shape code of the outer isothetic cover 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 outer isothetic cover (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 OIC 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 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 OIC are generated for an image for different grid sizes. Outer isothetic cover, as interpreted and realized from their shape codes, have been shown for a sample logo image in Fig. 4.3 for illustration. It may be observed in Fig. 4.3 how the underlying objects become more and more revealing as the grid size (separation of grid lines) goes on decreasing from 16 in Fig. 4.3(a) to 4 in Fig. 4.3(c).

Now, the **Mu**ltigrid **S**hape **C**ode (MuSC) for the image S_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:

$$MuSC(S_r) = \quad SC(S_r, g_1)\, 0^{16}\, SC(S_r, g_2)\, 0^{16} \cdots 0^{16}\, SC(S_r, g_\alpha), \tag{4.2}$$

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:

$$SC(S_r, g) = \quad SC(P_{r1}, g)\, 0^8\, SC(P_{r2}, g)\, 0^8 \cdots 0^8\, SC(P_{rn_r}, g). \tag{4.3}$$

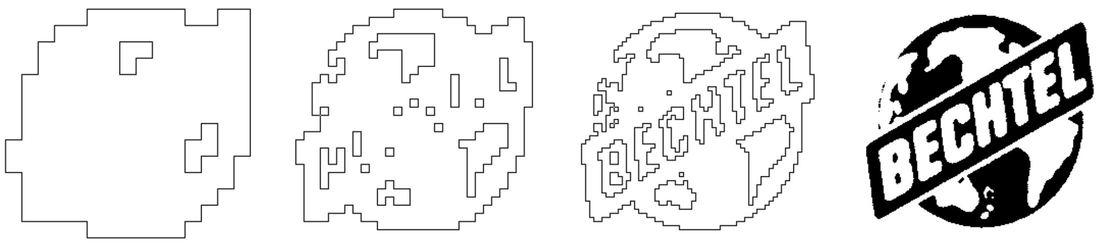
It may be noted that, since images in a database $\mathcal{D}$ may have nonuniform sizes, all images in $\mathcal{D}$ are normalized to the size of $\mu \times \mu$ before finding their MuSCs. In our experiments, we have considered $\mu = 256$ pixels.

4.2.4 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. 4.2. It may be noted that, for any image S, $MuSC(S)$ represents the highest level description of all the OICs corresponding to the object(s) present in the image S 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 S_r, an ordered list, namely $L(S_r, g_1)$, 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. 4.1). It may be noted that $SC(P_r, g_1)$, in turn, is extracted from its multigrid shape code, $MuSC(S_r)$ (given in Eqn. 4.2). The list $L(S_r, g_1)$ 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 $L(S_r, g_1)$ are as follows:

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


(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 4.3: A sample logo image and its OIC 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.

– each node in the linked list pointed from $L(S_r, g_1)[h]$ 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 retrieval process.

– in the linked list that $L(S_r, g_1)[h]$ 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 S_r for grid size g_1. In order to do this, we construct a binary matrix $M(S_r, g_1)$, having size $H_1 \times H_1$, considering $g = g_1$ and $H = H_1$ in Eqn. 4.4. A similar matrix, namely $M(\mathcal{Q}, g_1)$, is also being constructed for $\mathcal{Q}$ using Eqn. 4.4, and these two matrices, $M(S_r, g_1)$ and $M(\mathcal{Q}, g_1)$, are used to find the Hamming distance $L\left(M(S_r, g_1), M(\mathcal{Q}, g_1)\right)$ between the isothetic polygons of database image S_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. 4.5.

$$M(S_r, g)[w][h] = \begin{cases} 1, & \text{if } |\{e(u,v) \in L(S_r, g) : y[u] \leq h < y[v] \text{ AND } x[u] \leq w\}| \text{ is odd ;} \\ 0, & \text{otherwise;} \end{cases}$$
$$\text{for } 1 \leq w, h \leq H. \tag{4.4}$$

$$L\left(M(S_r, g), M(\mathcal{Q}, g)\right) = \sum_{w=1}^{H} \sum_{h=1}^{H} |M(S_r, g)[w][h] - M(\mathcal{Q}, g)[w][h]| \tag{4.5}$$

It is evident from Eqn. 4.4 and Eqn. 4.5, if the Hamming distance $L\left(M(S_r, g_1), M(\mathcal{Q}, g_1)\right)$ is small, then the image S_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.

4.3 Shape Complexity Measure

Shape complexity has been a very important measure in areas such as computer vision [Siddiqi and Kimia (1993)], satellite imagery [Oddo (1992)], geographic information systems [Brinkhoff *et al.* (1995)], and medical imaging [Cesar and Costa (1997)]. A shape complexity measure also plays a crucial role in designing computationally efficient shape classification algorithms [Barutcuoglu and DeCoro (2006), Tsai *et al.* (2005)].

A number of prior works on shape complexity are present in the literature. G. Toussaint (1991) has proposed a method based on polygon triangulation. In another approach [Chazelle and Incerpi (1984)], the sinuosity of the polygon boundary is measured to depict the shape complexity. Shape complexity is also calculated in terms of entropy of the curvature of the object contour [Page *et al.* (2003)]. In another approach, the shape context of each selected point on the boundary is calculated [Belongie *et al.* (2002)]. In fact, a large number of works has been done using either the boundary of the silhouette image, disregarding the hole or internal boundaries, or with a set of points on the extracted edge of the objects. Also, methods have been proposed based on Hausdorff distance [Huttenlocaher *et al.* (1993)], eigenvector or modal-matching based approach [Sclaroff and Pentland (1995)], Kolmogorov complexity [Chen and Sundaram (2005)]. Since these methods are formulated in the Euclidean domain, they involve complex calculations involving floating point operations, and are therefore, computationally expensive.

On the contrary, we propose a novel algorithm to capture the complexity of an image in the digital plane using some digital geometric properties [Klette and Rosenfeld (2004a), Yip and Klette (2003)] of an isothetic polygon. The algorithm uses outer isothetic cover [Bhowmick *et al.* (2005b), Biswas *et al.* (2005a,b)], which uses only comparison and addition operations in the integer domain, and is therefore, very fast, efficient, and robust. Shape complexity measure in this algorithm is derived using the properties of the tight isothetic polygon that contains the object. Since a square shape is the simplest figure in the realm of 2D digital geometry, our shape complexity technique considers a square as the simplest possible object. That is, the shape complexity measure of an object (polygon) is zero if it is a square. Further, in our method, the complexity measure also takes into account the complexity due to the presence of internal contours in an object, which is not considered in some of the existing works [Gdalyahu and Weinshall (1999), Sharvit *et al.* (1998)].

4.3.1 Shape Complexity of Objects using Outer Isothetic Cover

Once the outer isothetic cover of an object S is extracted, it is encoded with a string, $\mathbf{s}$, which consists of the different types of grid points (including the vertices) depicting the isothetic cover. There can be three types of such grid points on the polygonal envelope, namely **1**, **2**, and **3**. If $\mathbf{i}$ denotes the type of the grid point on the polygon, then it indicates that the internal angle of the polygon at that grid point is given by $\mathbf{i} \times \frac{\pi}{2}$. Clearly, type **1** and **3** denote the vertices with internal angles 90^0 and 270^0 respectively, while type **2**, with internal angle 180^0, indicates a simple edge point where the polygon edge propagates in the same direction. The three types of grid points are illustrated in Fig. 4.4. Thus, occurrence of **3** in $\mathbf{s}$ indicates a concavity, whereas **1** signifies a convex contour of the polygon, and thereof, of the object. Also, **2** indicates a straight edge of the polygon, thereby contributing no significant information regarding the shape complexity.

For an outer polygon, it can be shown that each concavity (**3**) on the polygon contour matches with a convexity (**1**), leaving four 90^0 vertices [Sur-Kolay and Bhattacharya (1988)], i.e. $\langle \mathbf{1111} \rangle$. Thus each polygon, encoded in string $\mathbf{s}$, can be reduced to $\langle \mathbf{1111} \rangle$ in case of an outer polygon, and to $\langle \mathbf{3333} \rangle$ in case of an (outer) hole polygon by replacing the

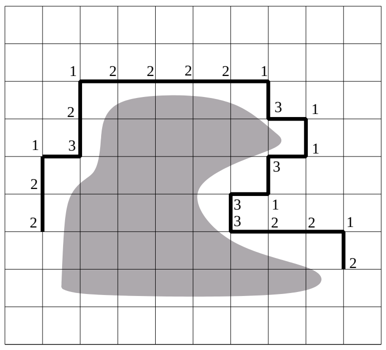


Figure 4.4: Grid point types **1**, **2**, and **3** on the outer ploygon (shown partially).

consecutive occurrences of **3** followed by **1**. For example, a square will have the polygon code $\mathbf{s} \equiv \langle \mathbf{1111} \rangle$.

The string **s** is reduced by applying the following reduction rules:

(1) *Edge Reduction Rule:* $\mathbf{2}^+ \rightarrow \epsilon$

(2) *Vertex Reduction Rule:* $(\mathbf{31})^+ \rightarrow \epsilon$

The *edge reduction rule* does not contribute to the complexity of the polygon, so these reductions are not included in the total number of reductions. This is evident from Fig. 4.4 in which the top edge of the isothetic polygon consists of several consecutive **2**s that indicates no significant variation on the contour of the object. On the contrary, in the right side of the polygon, the number of consecutive **2**s is less and also, the occurrence of **31**s is frequent, which indicates a complex nature of the associated part of the contour.

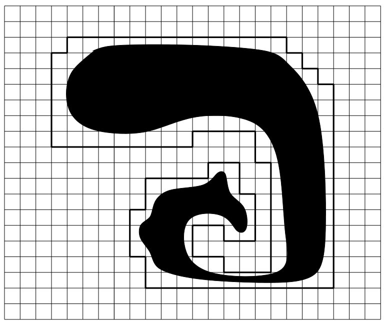
Let there be m polygons in $\overline{P}(S)$, and each polygon encoded in string s_i. Then each s_i will be reduced to either $\langle \mathbf{1111} \rangle$ or $\langle \mathbf{3333} \rangle$, i.e. to a substring of length 4, say, in at most n iterations. Then the shape complexity (SC), which is given by

$$SC = \frac{\sum_{k=1}^{n} \sum_{i=1}^{m} r_i^{(k)}}{\sum_{k=1}^{n} \sum_{i=1}^{m} l_i^{(k)}},$$

is computed by the algorithm SCOPE as follows.

Algorithm SCOPE

1. Extract the OIC $\overline{P}(S)$
2. Encode $\overline{P}(S)$ to string $s \equiv \langle s_i \rangle_{i=1}^{m}$



Itn.	String ($\mathbf{s}$)
1	$1^2(31)^3 1^2(31)1(31)^2 1(31)1^2 3^2(31)3^2(31)3(31)1^2 3$
2	$1^7 3^4(31)1$
3	$1^7 3^3(31)$
4	$1^6 3^2(31)$
5	$1^5 3(31)$
6	$1^4(31)$
7	1111

Figure 4.5: The outer polygon is encoded by the string $\mathbf{s} \equiv 1^2(31)^3 1^2(31)1(31)^2 1(31)1^2 3^2(31)3^2(31)3(31)1^2 3$ (itn. 1), after the application of the edge reduction rule on $\mathbf{s}$. The substring(s) is reduced at each subsequent iteration.

```
3. Apply rule (1) on $s$ to remove all **2**s from $\mathbf{s}$
4. $r \leftarrow 0, l \leftarrow 0$
5. Until each substring $s_i$ has length 4
   5.1. Apply rule (2) on $s_i$
   5.2.      $r \leftarrow r + \sum r_i$
   5.3.      $l \leftarrow l + \sum l_i$
6. $SC = \frac{r}{l}$
```

In Step 5.2, r_i indicates the number of occurrences of **31** those are reduced in the ith iteration, and l_i (Step 5.3) indicates the corresponding string length which has been reduced. In the expression of the shape complexity **SC**, as each $\mathbf{r_i} < \mathbf{l_i}$ for $i = 1 \ldots n$, **SC** is always less than 1. It should also be mentioned here that for the hole polygons (if any), the number of reductions and the corresponding string lengths also contribute to the computation of the final shape complexity. It may be noted here that while merging the concavities and convexities, the string $\mathbf{s}$ may be treated as circular.

4.3.2 Demonstration

Fig. 4.5 shows an object, the corresponding OIC, and the way the string is reduced in each iteration. For example, in iteration 1, r_1=8 and l_1=36. This is also clear from the demonstration that, higher the complexity of an object, larger is the number of iterations needed for the reduction.

4.4 Ranking of Optical Character Prototypes in a Large Database

Offline optical character recognition (OCR) is a well-researched problem as on now, and over the last few years, handwritten character recognition (HCR) has spurred intense research interest in the concerned group of researchers. However, due to comparatively low performance of an automated HCR system caused by the inevitable cursive variations in handwritten characters, it has been applied with restrictions to the size of lexicon, restrictions on writing styles, etc. [Kang and Kim (2004)], [Plamondon and Srihari (2000)].

In order to achieve high-performance HCR, therefore, large databases have been gradually developed throughout the years. Some of these are

1. AHDB made from 100 different writers, including words used for numbers and in bank checks [Almáadeed *et al.* (2002)];
2. CEDAR database collected from handwritten postal addresses [Hull (1994)];
3. IFN/ENIT database consisting of 26,459 images of the 937 names of cities and towns in Tunisia, written by 411 different writers [Märgner *et al.* (2005)];
4. ISI database of Bangla handwritten basic characters consisting of 20187 isolated basic character images [Bhattacharya *et al.* (2005)];
5. KU-1 Hangul character image database [Kim and Lee (1998)];
6. MNIST of handwritten digits [LeCun *et al.* (1998)];
7. NIST special database 3 [Garris and Wilkinson (1992), Geist *et al.* (1994)];
8. PE92 handwritten Korean character Image Database [Kim *et al.* (1993)];
9. Reuters-21578 clean text database [Lewis (1992)];
10. UNIPEN online handwriting database [Guyon *et al.* (1994)]; etc.

Each of these databases contains a large number of prototypes for each character in the corresponding alphabet. Recognizing a handwritten character, therefore, involves matching the character with the prototypes or their features, which is associated with a large overhead. The process can be made faster if we can arrange the prototypes in some order so that the recognition time may be reduced. For a given character, a subset of the prototypes, which are almost similar in structure, may be replaced by a smaller subset (ideally by a single prototype), if possible, depending on the tradeoff between speed and precision. Similarly, to include a new prototype in the database, we can think of checking its degree of dissimilarity with the existing prototypes corresponding to the concerned character in the database, and take the decision of whether or not include that prototype, accordingly. Further, to check the diversity of a database, we need to check how sparsely the prototypes are located in their respective feature space.

The proposed work introduces the novel concept of ranking the prototypes based on a single feature, namely the *total isothetic chord length* (Def. 4.4.2). For each character in the database, we consider all the prototypes and find their OICs (Def. 4.4.1) to derive the corresponding isothetic chord lengths. Using these chord lengths, we find the rank of each prototype from the corresponding measure of dissimilarity (Eqn. 4.10), which, in turn, is used to estimate the goodness/relevance of the existing set of prototypes of the concerned character in the database, and the database as a whole, thereof.

The rest of this section is organized as follows. Characterization of an isothetic polygon using the isothetic chord lengths is given in Sec. 4.4.1. The proposed method that includes dissimilarity measure between two isothetic polygons, policy of ranking the prototypes, database property, etc., are given in Sec. 4.4.2. To demonstrate the novelty and elegance of the proposed method, some preliminary results have been given in Sec. 4.5.3.

4.4.1 Characterization of an Isothetic Polygon using Isothetic Chord Lengths

The following definitions are newly introduced in this work for characterization of an optical character using the OIC that tightly encloses the character.

Definition 4.4.1 (isothetic chord)**.** Let p and p' be two grid points that lie on the isothetic polygon P. Then the line segment $\mathbf{L}(p, p')$, joining p and p', is said to be an *isothetic chord of* P, provided conditions (c1) and (c2) are simultaneously satisfied.

(c1) $\mathbf{L}(p, p')$ is either horizontal or vertical;

(c2) either $\mathbf{L}(p, p')$ is an edge of P or each point on $\mathbf{L}(p, p')$ (excepting p and p') lies inside P.

Note that, for a *vertical chord* joining p and p' having same x-coordinate, we use the notation $\mathbf{L}_x(p, p')$, whereas, for a *horizontal chord* joining p and p' with same y-coordinate, we use $\mathbf{L}_y(p, p')$.

Definition 4.4.2 (total isothetic chord length).
Let $S_x = \left\{\mathbf{L}_x\left(p^{(2i)}, p^{(2i+1)}\right)\right\}_{i=0}^{k_x-1}$ be the set of all vertical chords lying on the vertical grid line with abscissa x, and $S_y = \left\{\mathbf{L}_y\left(p^{(2j)}, p^{(2j+1)}\right)\right\}_{j=0}^{k_y-1}$ be the set of all horizontal chords lying on the horizontal grid line with ordinate y, corresponding to the isothetic polygon P. Then the *total vertical chord length at abscissa x for P* and the *total horizontal chord length at ordinate y for P* are given by

$$L_x = \sum_{i=0}^{k_x-1} L_x\left(p^{(2i)}, p^{(2i+1)}\right), \tag{4.6}$$

$$L_y = \sum_{j=0}^{k_y-1} L_y\left(p^{(2j)}, p^{(2j+1)}\right), \tag{4.7}$$

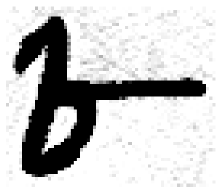
respectively, where, $L_x\left(p^{(2i)}, p^{(2i+1)}\right)$ (or, simply $L_x^{(i)}$) denotes the length of the ith vertical chord, namely $\mathbf{L}_x\left(p^{(2i)}, p^{(2i+1)}\right)$, and $L_y\left(p^{(2j)}, p^{(2j+1)}\right)$ (or, simply $L_y^{(j)}$) that of the jth horizontal chord, namely $\mathbf{L}_y\left(p^{(2j)}, p^{(2j+1)}\right)$, corresponding to P.

4.4.2 Proposed Method

Let c be any digital (optical/handwritten) character, and $\overline{P}(c)$ be the OIC constructed using the algorithm MAKE-OIC as mentioned in Chapter 2 (see Sec. 2.6.1). Note that, $\overline{P}(c)$ is subject to change with changes in the grid parameters, especially the separation g between the (equi-spaced — horizontal or vertical) grid lines. Let c_u^v be the uth prototype of the vth optical character in an alphabet. Let P_u^v be the isothetic polygon corresponding to the prototype c_u^v. Let $\mathcal{X}_u^v = \left\langle L_{u,x}^v : 1 \leqslant x \leqslant w_u^v + 1\right\rangle$ and $\mathcal{Y}_u^v = \left\langle L_{u,y}^v : 1 \leqslant y \leqslant h_u^v + 1\right\rangle$ be the respective ordered list of the total vertical chord lengths and that of the total horizontal chord lengths corresponding to the polygon P_u^v, where w_u^v and h_u^v denote the width and height of the isothetic polygon P_u^v, which has, therefore, intersections with exactly w_u^v+1 number of vertical grid lines and h_u^v+1 number of horizontal grid lines. Few sample lists of total horizontal chord lengths corresponding to the prototype of a Bangla numeral '8' for different grid sizes have been shown in Fig. 4.6 as a ready reference.

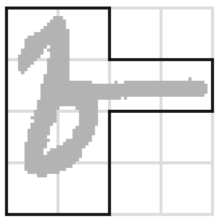
4.4.3 Dissimilarity Measure

Let c_u^v and $c_{u'}^{v'}$ be two prototypes. Let they are normalized w.r.t. their sizes, such that the widths and the heights of their respective isothetic polygons, namely P_u^v and $P_{u'}^{v'}$, become identical (i.e., $w_u^v = w_{u'}^{v'} = w$ and $h_u^v = h_{u'}^{v'} = h$) for a given grid of a given size, g. Then



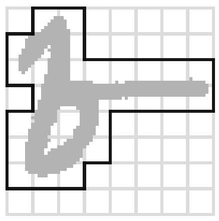
(a)

a prototype of the Bangla numeric character ‘**8**’.



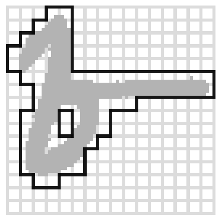
(b)

$g = 16$.
$\mathcal{Y} = \langle 2, 4, 4, 2, 2 \rangle$.



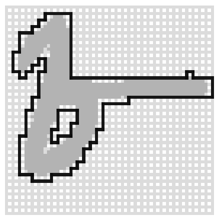
(c)

$g = 8$.
$\mathcal{Y} = \langle 2, 3, 8, 8, 8, 4, 4, 3 \rangle$.



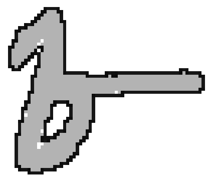
(d)

$g = 4$.
$\mathcal{Y} = \langle 2, 3, 4, 5, 5, 16, 15, 14, 9, 6, 7, 6, 5, 5, 2 \rangle$.



(e)

$g = 2$.
$\mathcal{Y} = \langle 4, 5, 6, 8, 8, 8, 9, 9, 8, 9, 28, 27, 27, 28, 15, 12, 9, 10, 10, 11, 12, 11, 10, 10, 9, 8, 3 \rangle$.



(f)

$g = 1$.
$\mathcal{Y} = \langle 4, 6, 7, 8, 10, 12, 14, 15, 15, 15, 16, 16, 17, 16, 16, 15, 14, 13, 13 13, 44, 53, 53, 53, 53, 53, 53, 29, 20 21, 17, 16, 16, 16, 17, 18, 16, 16, 18 19, 21, 21, 20, 19, 19, 19, 18, 17, 16 14, 11, 4 \rangle$.

Figure 4.6: Outer polygons and their total horizontal chord lengths (elements of $\mathcal{Y}$) for a typical prototype of the Bangla numeral ‘**8**’ (813 object/black pixels) for different grid sizes, g.

we define the dissimilarity measure between the two prototypes by considering the total isothetic chord lengths of their corresponding OICs. The dissimilarity measure, $d_x(c_u^v, c_{u'}^{v'})$, between c_u^v and $c_{u'}^{v'}$, w.r.t. their vertical chords, is given by the sum over the difference of each element (total chord length) in $\mathcal{X}_u^v$ with the corresponding element (that has 1-1 correspondence owing to the normalized width) in $\mathcal{X}_{u'}^{v'}$, which is given in Eqn. 4.8. The dissimilarity measure, $d_y(c_u^v, c_{u'}^{v'})$, between c_u^v and $c_{u'}^{v'}$, w.r.t. their horizontal chords can be defined in a similar way by considering $\mathcal{Y}_u^v$ and $\mathcal{Y}_{u'}^{v'}$, as shown in Eqn. 4.9.

$$d_x(c_u^v, c_{u'}^{v'}) = \sum_{x=1}^{w+1} \left| L_{u,x}^v - L_{u',x}^{v'} \right|. \tag{4.8}$$

$$d_y(c_u^v, c_{u'}^{v'}) = \sum_{y=1}^{h+1} \left| L_{u,y}^v - L_{u',y}^{v'} \right|. \tag{4.9}$$

Using d_x and d_y, the *overall dissimilarity* between P_u^v and $P_{u'}^{v'}$ (between c_u^v and $c_{u'}^{v'}$, thereof) is given by

$$d(c_u^v, c_{u'}^{v'}) = d_x(c_u^v, c_{u'}^{v'}) + d_y(c_u^v, c_{u'}^{v'}). \tag{4.10}$$

Thus, higher the value of $d(c_u^v, c_{u'}^{v'})$, lesser is the similarity between the prototypes c_u^v and $c_{u'}^{v'}$, and lesser the value of $d(c_u^v, c_{u'}^{v'})$, higher is the similarity between them.

4.4.4 Ranking of Prototypes

Using the overall dissimilarity, given in Eqn. 4.10, the prototypes available in a particular database D for a given alphabet are ranked as follows.

Let $\mathcal{C}_u^v = \left\langle c_{u'}^{v'} : c_{u'}^{v'} \in D; c_{u'}^{v'} \neq c_u^v \right\rangle$ be the (ordered) list in the nonincreasing order of the dissimilarity measures ($d(c_u^v, c_{u'}^{v'})$) of the prototypes in D as measured from the prototype c_u^v. Let the database D contains n_v number of vth prototype, out of which $m_u^v (\leqslant n_v)$ prototypes are found to be among the first n_v entries in $\mathcal{C}_u^v$. It is evident that, if $m_u^v = n_v$, then we may consider the prototype c_u^v as an obvious (and hence less mistakable) representative of the vth character captured in D; whereas, if m_u^v is (appreciably) less than n_v, then the prototype c_u^v is an offbeat (and hence less unmistakable) representative of the concerned character. In other words, if $m_{u_1}^v$ corresponding to a prototype $c_{u_1}^v$ is greater than $m_{u_2}^v$ corresponding to a prototype $c_{u_2}^v$ (both the prototypes being in representation of the same (vth) character), then the former has more redundancy than the latter; and we consider that $c_{u_1}^v$ has a higher rank than $c_{u_2}^v$. That is, in this work, lower the rank, better the prototype.

Further, if the prototypes $c_{u_1}^v$ and $c_{u_2}^v$ have $m_{u_1}^v = m_{u_2}^v = m_u^v$ (say), then we consider the mean dissimilarity measures of their first m_u^v correct prototypes in $\mathcal{C}_{u_1}^v$ and $\mathcal{C}_{u_2}^v$ respectively. Hence, if $\overline{d}_{u_1}^v$ and $\overline{d}_{u_2}^v$ be the respective means (of *dissimilarity measures*) of $c_{u_1}^v$ and $c_{u_2}^v$, and if, w.l.o.g., $\overline{d}_{u_1}^v < \overline{d}_{u_2}^v$, then $c_{u_1}^v$ has a higher rank than $c_{u_2}^v$. In case of a tie (i.e., $\overline{d}_{u_1}^v = \overline{d}_{u_2}^v$), we consider that the two prototypes have same rank.

Using the above ranks of n_v prototypes for the vth character, we get an ordering of them (in nondecreasing order of their ranks), which is stored in the (ordered) set

$\mathcal{A} = \left\langle c_{u_1}^v, c_{u_2}^v, \ldots, c_{u_{n_v}}^v \right\rangle$. Note that, in $\mathcal{A}$, c_u^v occurs before $c_{u'}^{v'}$ if and only if either $m_u^v < m_{u'}^v$ or $\overline{d}_u^v \geqslant \overline{d}_{u'}^v$.

4.4.5 Goodness of the Database

It is evident from Sec. 4.4.4 that an ideal database of prototypes should be such that for each prototype (c_u^v) corresponding to each (vth) character of the alphabet, each other prototype $c_{u'}^v$ of that character class should have $d(c_u^v, c_{u'}^v)$ greater than $d(c_u^v, c_{u''}^{v''})$ corresponding to any prototype $c_{u''}^{v''}$ of any other class. That is, for an ideal database, we have $m_u^v = 0, \forall u, \forall v$. For a non-ideal case, the goodness of the database is related with the departure of the prototypes from the ideal situation. The goodness of the collection of all the prototypes corresponding to the vth character is, therefore, given by

$$GI_v = \frac{1}{n_v} \sum_{u=1}^{n_v} \left(1 - \frac{m_u^v}{n_v}\right) = 1 - \frac{1}{{n_v}^2} \sum_{u=1}^{n_v} m_u^v, \tag{4.11}$$

since $(n_v - m_u^v)/n_v$ signifies the goodness of the uth prototype of the vth character present in D.

Hence, the goodness index of a database D containing the prototypes of an alphabet of N characters is defined by

$$\begin{aligned} GI_D &= \tfrac{1}{N} \sum_{v=1}^{N} \left(1 - \tfrac{1}{{n_v}^2} \sum_{u=1}^{n_v} m_u^v\right) \\ &= 1 - \tfrac{1}{N} \sum_{v=1}^{N} \tfrac{1}{{n_v}^2} \sum_{u=1}^{n_v} m_u^v, \end{aligned} \tag{4.12}$$

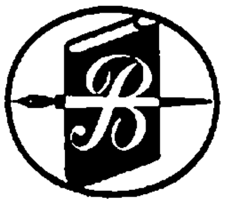
where, $GI_D = 1$ implies maximum/ideal goodness of D, and its goodness declines as GI_D falls short of 1.

4.5 Experimental Results

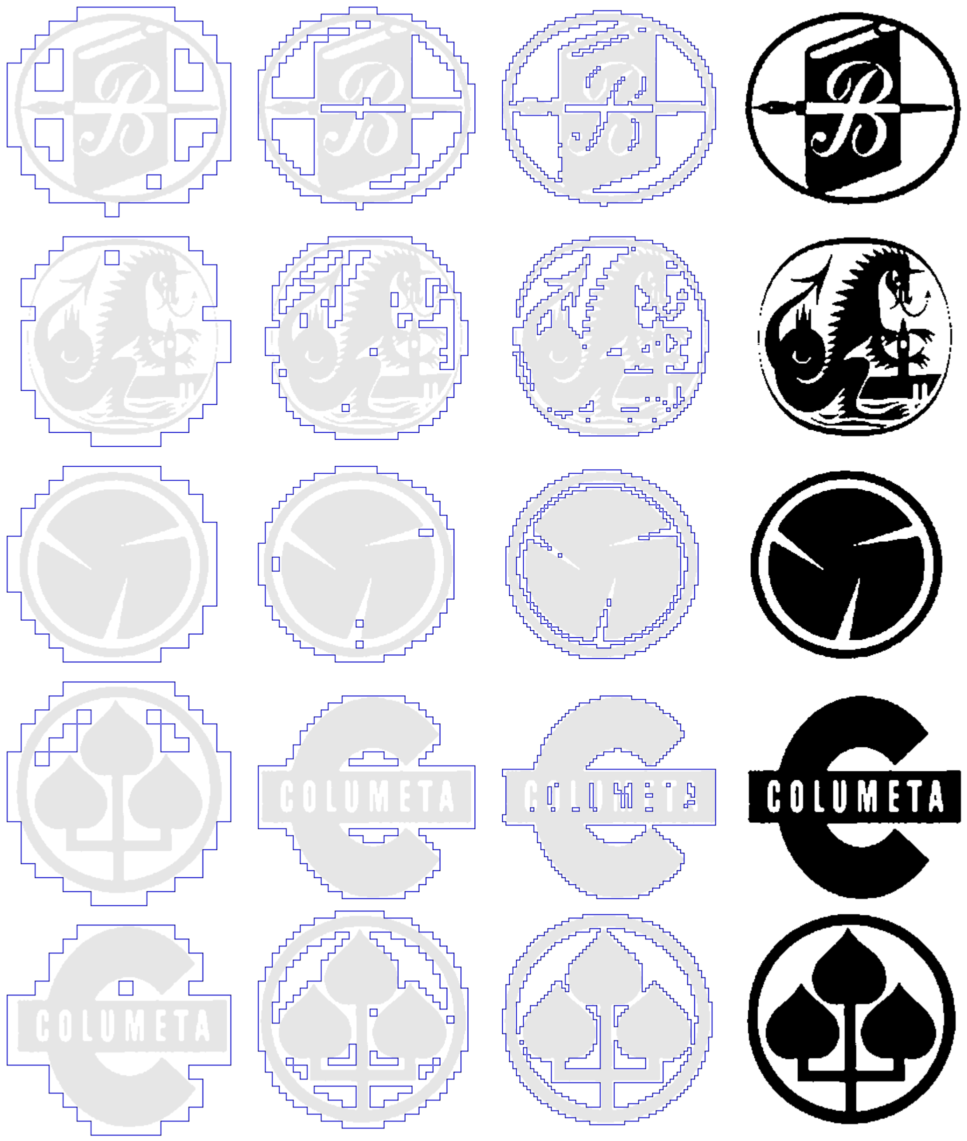
4.5.1 Multigrid Shape Code

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 4.1. All the images are normalized to



Query image



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

Figure 4.7: The results of querying the database with the query image are shown. The images in each column are ordered from top to bottom in increasing hamming distance while different columns correspond to different grid sizes.

Table 4.1: Results (average storage and CPU time per image) for databases D1 and D2

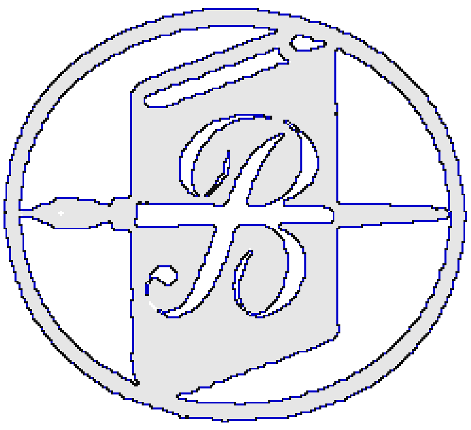
	D1 (1034 images)			D2 (110 images)		
	Storage required		CPU Time	Storage required		CPU Time
grid size	bytes	%	millisecs.	bytes	%	millisecs.
16	51363	0.60	22.10	456	0.05	25.17
8	126113	1.49	31.04	12338	1.37	34.79
4	298943	3.53	52.40	32038	3.55	64.85
2	677208	8.00	105.97	71532	7.93	123.32
1	1422700	16.80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

size 256×256. The table shows the amount of storage (in bytes 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. 4.7, 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. 4.8.

4.5.2 Shape Complexity Measure

We have tested the algorithm SCOPE on two sets of images: 1) test images, which are arbitrarily drawn objects, and 2) logo images, obtained from Prof A.K. Jain on request. As shown in Fig. 4.9, the complexity of a square (a) is 0, whereas that of a convoluted shape (f) is 0.74. It is evident from the shape of the isothetic polygon of the square image that there are no 270^0 vertices, so there are zero reductions, thereby assigning zero


(a) $g = 1$.

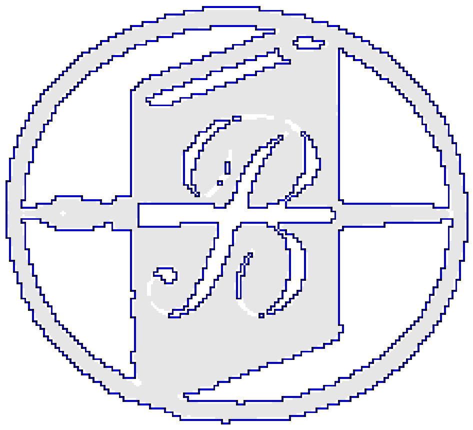

(b) $g = 2$.

Figure 4.8: Shape codes extracted at low grid sizes (i.e., 1 and 2) aid in object visualization.

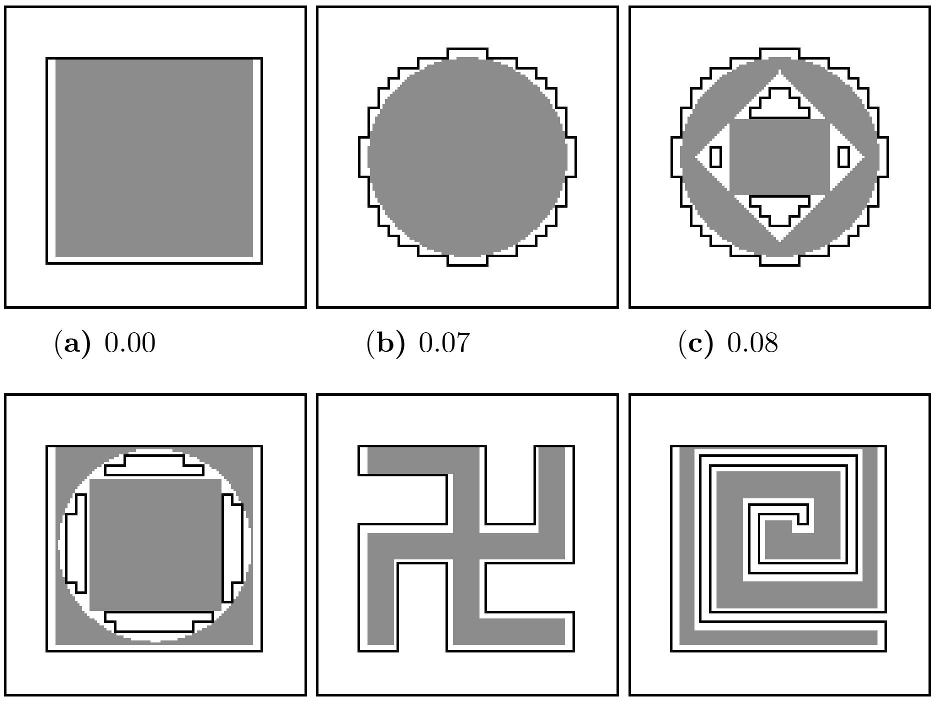
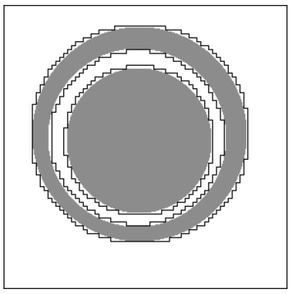
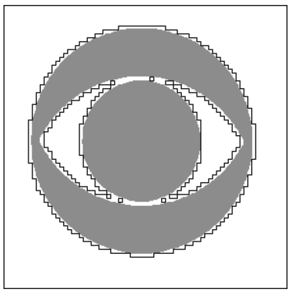
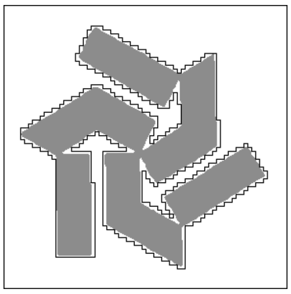
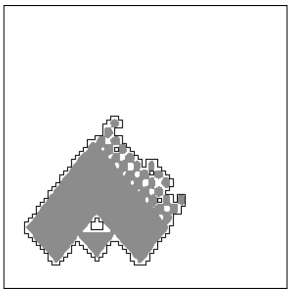

(a) 0.00
(b) 0.07
(c) 0.08
(d) 0.15
(e) 0.22
(f) 0.74

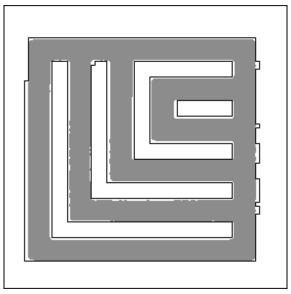
Figure 4.9: The shape complexity values for a set of synthetic images.


(**a**) 0.077


(**b**) 0.082


(**c**) 0.123


(**d**) 0.130


(**e**) 0.150

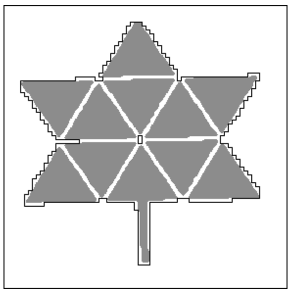

(**f**) 0.167

Figure 4.10: The shape complexity values for a set of logo images.

complexity to it. Also, for the logo images, as shown in Fig. 4.10, the shape complexity increases with the increasing structural complexity. This method is robust as compared to the methods where the shape contour is used, because the contour extraction is prone to noise in the image. The complexity of the algorithm is output sensitive, and depends upon the number of grid points lying on the OIC corresponding to the object.

4.5.3 Ranking of Optical Character Prototypes

We have implemented the algorithm for finding the isothetic chord lengths and hence ranking the prototypes of a database in C in SunOS Release 5.7 Generic of Sun_Ultra 5_10, Sparc, 233 MHz. We have experimented on the ISI database of isolated Bangla characters [ISI (2002)], and a part of CEDAR database [Hull (1994)], and some results on ISI Bangla database are presented here.

For a particular database, we consider the width and height of the bounding box (BB) of all the prototypes of each (vth) character, and take their averages to get the average width (w_v) and average height (h_v) of the BB corresponding to the vth character. The

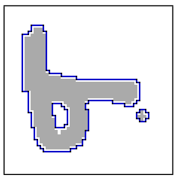


Figure 4.11: Instance of a prototype that needs two outer polygons to cover it completely (for $g = 2$). The chord lengths associated with such a prototype is obtained by adding the respective chord lengths of the two polygons.

BB of each prototype c_u^v has been normalized to width$= w_v$ and height$= h_v$, and then binarized to derive their chord lengths and subsequent ranking.

It should be mentioned here that, in the stage of finding the OIC, we get a single outer polygon of a prototype in most of the cases due to the implicit property of an isothetic polygon in covering small perturbations and spurious errors (that crept in during acquisition of prototypes and subsequent processing, especially binarization) present in a prototype. However, if there are multiple outer polygons (in case of too much noisy distractions), then we add the corresponding chord lengths (L_x and L_y, Def. 4.4.2) of all the polygons[1] to get the total chord lengths corresponding to that character. Fig. 4.11 illustrates such an unusual case for $g = 2$.

We have executed our scheme for different possible values of grid size, from $g = 1$ to $g = 16$, and have found that the ranking is dependent on g. Average width and height of the BB of a prototype are the major parameters that decide a proper value of g, and the ranking output thereof. From Fig. 4.6, it is evident that a small value of g picks up too much details of a prototype, whereas a large value neglects/overlooks some essential part (such as a loop/hole), thereby producing unexpected impression of its structural information.

Few sample prototypes (normalized and binarized), their isothetic polygons, and their ranked order based on total isothetic chord lengths have been shown in Fig. 4.12 for grid size $g = 4$. We have also given their mean dissimilarity measures, namely $\overline{d}$ (Sec. 4.4.4), to give an idea of the numerical values in the output of our experimentation. As explained in Sec. 4.4.4, since for the vth character, we have considered only the other prototypes of vth character which are within the first n_v prototypes (of minimum d_u^v), the $\overline{d}$'s in Fig. 4.12 are not increasing with the ranks. In Fig. 4.13, the best (unobvious/offbeat) prototypes

[1]Actually, a large polygon is there that represents the prototype and the small one(s) just cover(s) its noisy part(s) (Fig. 4.11).

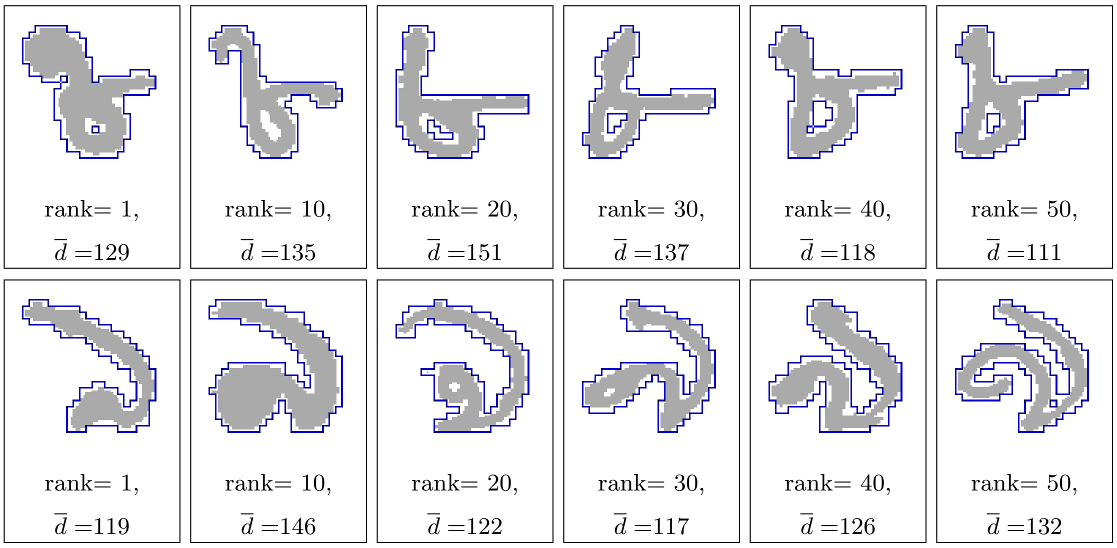

rank= 1, $\overline{d}$ =129
rank= 10, $\overline{d}$ =135
rank= 20, $\overline{d}$ =151
rank= 30, $\overline{d}$ =137
rank= 40, $\overline{d}$ =118
rank= 50, $\overline{d}$ =111
rank= 1, $\overline{d}$ =119
rank= 10, $\overline{d}$ =146
rank= 20, $\overline{d}$ =122
rank= 30, $\overline{d}$ =117
rank= 40, $\overline{d}$ =126
rank= 50, $\overline{d}$ =132

Figure 4.12: Sample prototypes of Bangla numerals '8' (top row) and '9' (bottom row) [ISI (2002)] shown with their ranks (for $g = 4$) in increasing order from left to right. For each numeral, number of prototypes is 50. Note that, a prototype with lower rank carries more significance in the database, since it is "uncommonly found" as evident in this enumeration. (See text for further explanations.)

and the worst prototypes for some Bangla vowels [ISI (2002)] corresponding to $g = 4$ have been given to illustrate the elegance and effectiveness of the proposed method.

The quantitative measures of our method, such as change of goodness index (GI_D) (Sec. 4.4.5) of the database (ISI Bangla) and CPU times versus grid-size g, have been given in Table 4.5.3. Note that, the total isothetic chord lengths (horizontal/vertical) of a prototype are found during the construction stage of the corresponding isothetic polygon. Hence the CPU time of construction of their lengths (L_x and L_y) is included in that of finding the isothetic polygon (t_P). Further, to have a glance of the distribution of mean dissimilarity measures and the number of prototypes, m_u^v (of same character in first n_v ones, see Sec. 4.4.4), we have given a plot in Fig. 4.14 corresponding to few numeric characters of ISI Bangla database, which justifies the goodness level of the database.

4.6 Conclusion

In this chapter, we have presented three different applications of outer isothetic cover of a digital object for analysis of the shape information contained in it. The multigrid

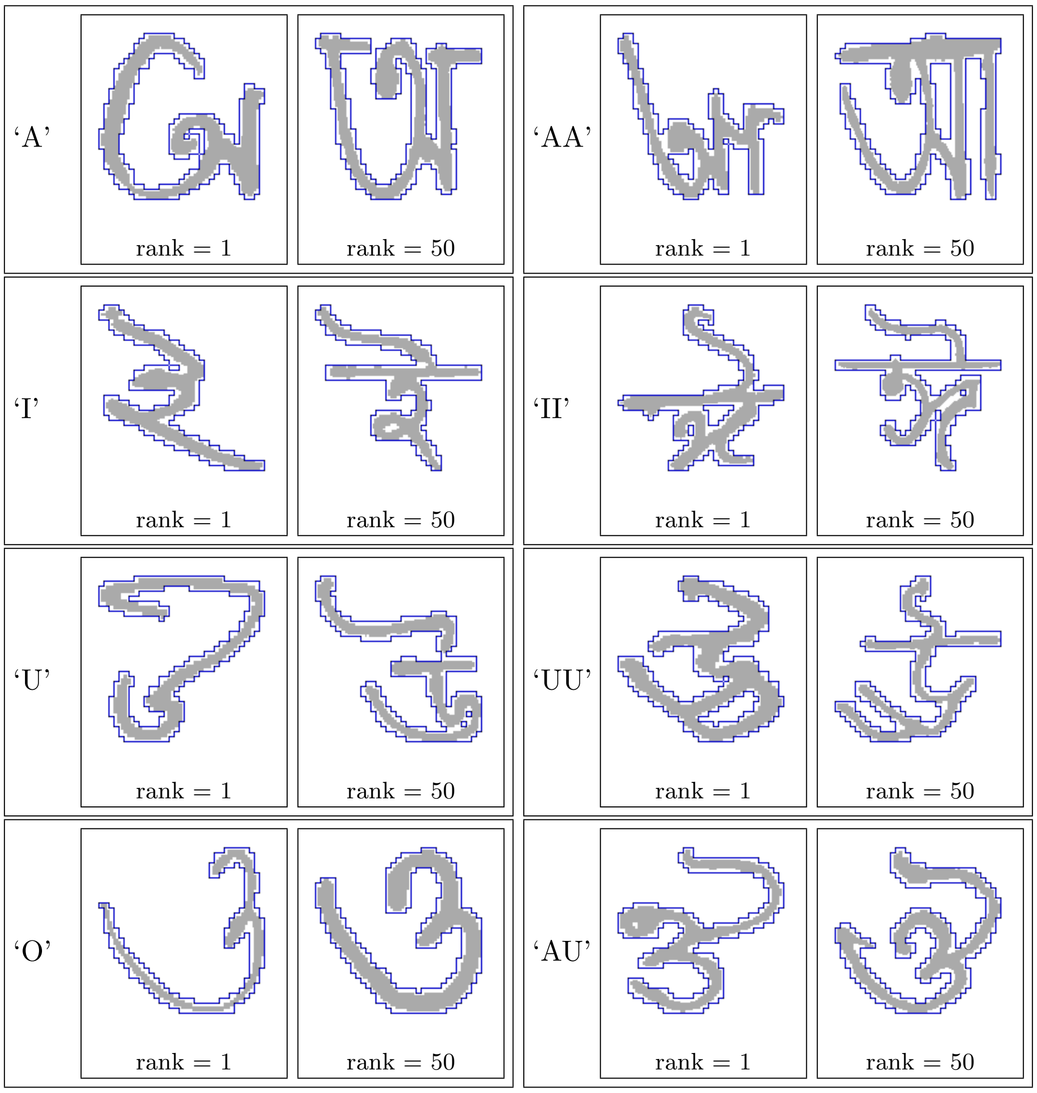
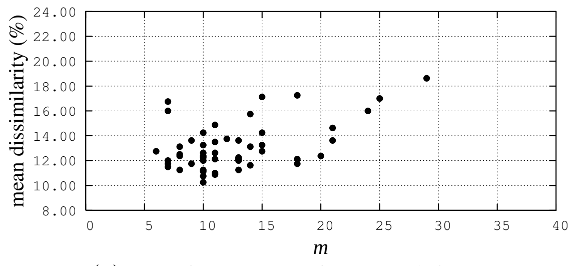


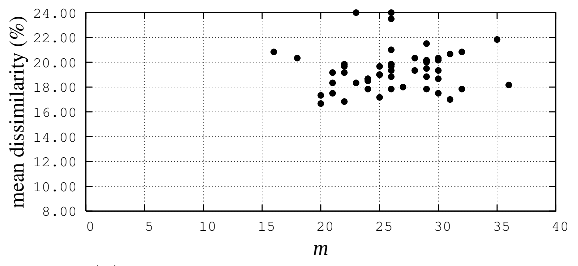
Figure 4.13: Best offbeat prototypes (rank = 1) and worst offbeat prototypes (rank = 50) corresponding to some vowels of Bangla alphabet [ISI (2002)].

shape code derived from the OIC captures the topological feature of an object. The shape complexity measure based on the reduction rules applied on the OIC is a unique measure of complexity of digital objects. It has also been shown that how the isothetic chord lengths of an OIC describing an optical character can be used for ranking of optical character prototypes.

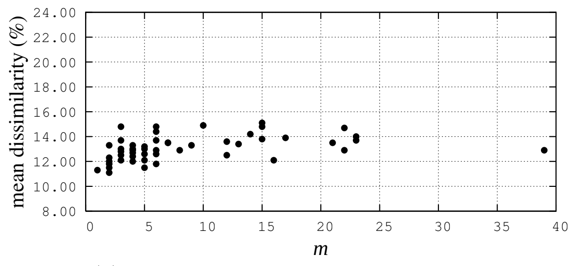
In Sec. 4.2.1, we have introduced a novel technique for determining the varying reso-



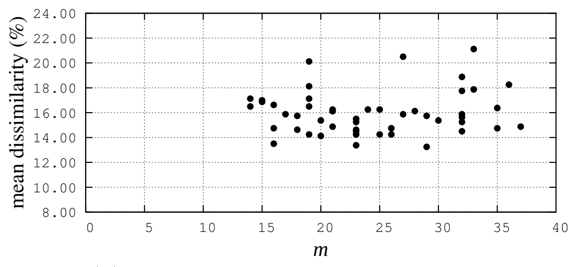
(a) Bangla numeric character ‘**0**’.



(b) Bangla numeric character ‘**1**’.



(c) Bangla numeric character ‘**8**’.



(d) Bangla numeric character ‘**9**’.

Figure 4.14: Distribution of mean dissimilarity measure and m_u^v (Sec. 4.4.4) for all prototypes of vth character corresponding to four typical numeric figures of ISI Bangla database (with $g = 4$).

g	GI_D	Average CPU time (millisecs.) per prototype		
		t$_P$	t$_d$	total
2	0.427	8	53	61
4	0.662	6	42	48
6	0.630	5	34	39
8	0.549	5	26	31
10	0.411	3	17	20
12	0.396	2	11	13

Table 4.2: Results on ISI Bangla database.

lution shape codes of a binary image using the classical properties of isothetic polygons. An efficient retrieval scheme is designed and implemented 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 [Bishnu *et al.* (2006)], 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.

In this chapter, we have also presented a shape complexity measure of objects in the digital geometry domain without using any Euclidean measure. The computations being in the integer domain, apart from one division operation[1] in the shape complexity expression, turn out to be very fast. As with the variation of the grid size the desired compactness of the object is obtained, this readily lends itself to multiscale treatment. Also as a corollary to this work, a number of questions arising in the emerging domain of digital geometry,

[1]The 0-1 scale of SCOPE needs the division, which can be avoided if we consider the upper limit of the measure as max $\sum L_k$ for a given (normalized) background grid.

such as, how many distinct isothetic polygons can be drawn on a given grid plane, how difficult it is to generate the exhaustive set of isothetic polygons in the $n \times n$ grid, etc. Another interesting problem may be to generate the string (s) that represent the isothetic polygon(s) of a given complexity.

The idea of ranking the prototypes in a large database of optical/handwritten characters based on isothetic chord lengths has a novel and prospective potential in the domain of OCR/HCR, which is exhibited by the results and the quantitative measures shown in this chapter. Since deriving the structural information of a character by thinning or edge detection is vulnerable to the impurity of the character, the feature of isothetic chord length may serve as an alternative solution. However, instead of a single grid size, g, multiple grid sizes may be considered at a time for a particular database to derive the multi-valued features of chord lengths for subsequent ranking. Using the ranked prototypes for recognition of handwritten characters is another interesting problem that needs intense study and experimentation.

Chapter 5

Polygonal Approximation of Thick Digital Curves

5.1 Introduction

Exploration of properties, characterizations, and representations of digital curves (DC) has been studied over the years since the debut of digitization of graphical objects and visual imageries [Klette and Rosenfeld (2004a,b), Rosenfeld and Klette (2001)]. Nevertheless, in the abundance of various problems and their algorithms related with digital objects, polygonal approximation of a digital curve/object has received special attention for its efficient representation and potential applications in connection with analysis of digital images [Aken and Novak (1985), Attneave (1954), Imai and Iri (1986)]. The set of straight edges of the polygons carries a strong geometric property of the underlying objects that have been approximated. Such information can be used for efficient high-level description of the objects and for finding the similarity among different objects in the digital plane.

Since an optimal solution of polygonal approximation targeted to minimize the number of vertices, and space thereof, is computationally intensive, several heuristic and meta-heuristic approaches based on certain optimality criteria have been proposed over the last few decades, and some of these that have come up in recent times may be seen in the literature [Bhowmick and Bhattacharya (2007), Perez and Vidal (1994), Schröder and Laurent (1999), Schuster and Katsaggelos (1998), Tanigawa and Katoh (2006), Teh and Chin (1989), Yin (2003)]. Further, there also exist various studies and comparisons of the proposed techniques, e.g., [Bhowmick and Bhattacharya (2007), Rosin (1997), Teh and Chin (1989), Yin (1998)], to cite a few. This entire collection of polygonal approximation algorithms, however, consider the input digital curve to be strictly "irreducible"[1] (and

[1]A digital curve $\mathcal{C}$ is said to be "irreducible" if and only if removal of any grid point p in $\mathcal{C}$ makes $\mathcal{C}$ disconnected.

connected thereof), failing which the algorithm may produce undesired results pertaining to polygonal approximation.

In the case of a thick DC, thinning is required to ensure the property of "irreducibility" so that it can qualify for the subsequent process of polygonal approximation. A thinning procedure, being plagued by asymmetric erosion in the thick regions and shifting of junction/end points, and being liable to slow down the overall run time of the approximation process, is susceptible to deteriorate the results of approximation. Furthermore, the result goes on worsening if there occur some missing grid points (pixels) in the input DC — which splits, therefore, into multiple DC's — producing several approximate polygons instead of a single polygon, thereby giving rise to misleading impression, and more specifically, posing severe problems in the subsequent applications.

5.1.1 Existing Methods

Algorithms for approximating a given digital curve or contour, which is one-pixel thick, had been proposed since the early period of digitization [Aken and Novak (1985), Attneave (1954), Imai and Iri (1986)]. Efficient and suboptimal algorithms of several variants had been proposed later [Bhowmick *et al.* (2005b, 2006), Biswas *et al.* (2005b), Perez and Vidal (1994), Schröder and Laurent (1999), Schuster and Katsaggelos (1998)]. The problem of polygonal approximation of a digital curve persists to be engrossing even today, and with the emergence of new paradigms that open up new possibilities, a number of algorithms have been proposed in recent times [Asano and Kawamura (2000), Asano *et al.* (2003), Chen and Chung (2001), Climer and Bhatia (2003), Guru *et al.* (2004), Xie and Ji (2001)]. All these methods are meant for polygonal approximation of a strictly one-pixel thick digital curve and mostly based on the conventional techniques, such as an appropriate distance criteria, usage of masks, eigenvalue analysis, Hough transform, etc. In general, all these algorithms can be broadly classified into two categories — one in which the number of vertices of the approximate polygon(s) is specified, and the other where a distortion criterion (e.g., maximum Euclidian distance) is used. The principles and salient features of some of these algorithms are mentioned below.

Amongst the earlier algorithms, the scan-along algorithm proposed by Wall and Danielsson (1984) needs a mention here for its high speed of execution owing to a simple yet effective concept of *area deviation* for each line segment. The algorithm outputs a new line segment when the area deviation per unit length of the current approximating segment exceeds a prescribed threshold. The provision for simplifying the associated computations is also available, e.g., by replacing $\sqrt{x^2 + y^2}$ with $(|x| + |y|)$ or with $\max\{|x|, |y|\}$. However, the algorithm lacks the scope of producing the desired result for (self-)intersecting or branching curves when the input set of points constituting the curve-set is not given in the order that defines the curve.

Another procedure proposed by Teh and Chin (1989) detects the *dominant points* in a closed digital curve, given in the chain-coded representation as input. A dominant point corresponds to *curvature maxima* in the local neighborhood of the concerned curve. The procedure first determines the *region of support* for each point based on its local properties, computes measures of relative significance (e.g., cosine curvature, k curvature, 1 curvature, etc.) of each point, and finally detects dominant points by *non-maxima suppression.* However, since the procedure needs trigonometric functions for curvature finding and performs non-maxima suppression in multiple iterations (4 passes), its runtime is quite high.

Later, the concept of *perceptual organization* was introduced by Hu and Yan (1997), which attempts to match the human performance. The approximation process is divided into three stages. In the first stage, points are grouped together using a *linking-merging* approach based on the principles of proximity, similarity, and symmetry. In the second stage, the linked curve is smoothed so as to suppress noise and delete visually insignificant points. Finally, a rule-based strategy is applied to preserve the feature points while reducing the number of segments in the approximated polygon. The procedural complexity and the runtime are high due to three stages, each using multiplications for the entire set of points on the curve.

Another method of polygonal approximation using *ant colony search* technique is proposed by Yin (2003). A directed graph is used to represent the problem with the objective of finding the *shortest closed circuit* on the graph under the problem-specific constraints. A number of artificial ants are distributed on the graph, which communicate with one another through the *pheromone trails* as a long-term memory guiding the future exploration of the graph through certain *node transition rules* and *pheromone updating rules.* The procedural complexity is very high because of its inherent recursive nature and complex calculations, one of which is exponentiation in selection problem of a node.

A comparative study of the above algorithms can be found in some of the recent papers [Bhowmick and Bhattacharya (2007), Yin (1998)]. Since most of these algorithms require intensive floating-point operations in order to analyze the discrete curvature [Anderson and Bezdek (1984), Fischler and Wolf (1994), Freeman and Davis (1977), Teh and Chin (1989), Wuescher and Boyer (1991)], their runtime for a complex digital curve is

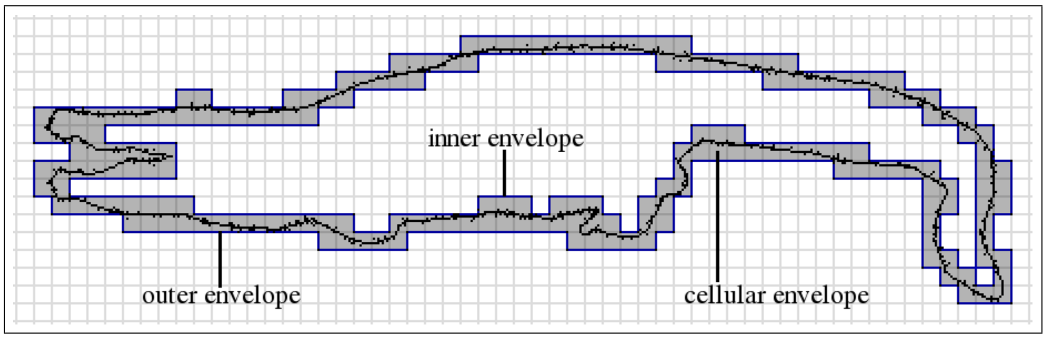


Figure 5.1: Cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ of a real-world (thick, rough, and reducible) curve-shaped object $\mathcal{C}$ for cell size $g = 8$.

quite high. For other details, the related procedures in several other works [Bezdek and Anderson (1985), Dunham (1986), Pavlidis (1980), Rosin (1997), Teh and Chin (1989), Wall and Danielsson (1984), Wu (1984), Yin (2003, 2004)] may be looked at. Further, for thick, rough, and weakly disconnected digital curves (which are usually representatives of the corresponding real-world objects), the algorithms are susceptible to produce unexpected results. On the contrary, the proposed algorithm can deal with such non-ideal curves, and yields a suboptimal solution of polygonal approximation by using primitive integer operations only.

5.1.2 Main Results

These above-mentioned problems have been investigated in this chapter, using the novel concept of cellular envelope of an arbitrary digital curve whose thickness may vary non-uniformly.

A brief outline of the work is as follows. In Sec. 5.2, we present the concept of cellular envelope of an arbitrary DC[1] using its inner and outer isothetic polygons [Bhattacharya and Rosenfeld (1990), Biswas *et al.* (2005b), Yu and Thonnat (1992)]. Sec. 5.3 discusses some digital geometric properties of cellular straight line segments (CSS), followed by the motivation and underlying principle for their extraction (stage II) from the cellular envelope of the input DC obtained in stage I. In Sec. 5.4, we present our method **PACE** (**P**olygonal **A**pproximation of Thick Digital Curves using **C**ellular **E**nvelope) along with

[1]In this work, we use the term “DC” to denote a digital curve (reducible or irreducible) as well as a curve-shaped object that may contain multiple disconnected segments producing the impression of a single object.

a brief analysis. Sec. 5.5 reports some test results on some objects with variable thickness values. Finally in Sec. 5.6, we summarize the strength of our method, comment on its future scope and further works, and point out the possibilities of polygonal approximation in the cellular plane.

5.2 Cellular Envelope

Let $\mathcal{C}$ be a given DC, and $\mathcal{G} = (\mathcal{H}, \mathcal{V})$ be the digital grid (Defn. 2.2.5). Then the cellular envelope of $\mathcal{C}$, corresponding to the cellular plane defined by $\mathcal{G}$, is given by

$$\mathcal{E}(\mathcal{C}, \mathcal{G}) \quad = \mathcal{E}_{out}(\mathcal{C}, \mathcal{G}) \smallsetminus \mathcal{E}_{in}(\mathcal{C}, \mathcal{G}) \tag{5.1}$$

where $\mathcal{E}_{out}(\mathcal{C}, \mathcal{G})$ and $\mathcal{E}_{in}(\mathcal{C}, \mathcal{G})$ represent the respective outer and inner envelopes corresponding to $\mathcal{C}$ w.r.t. $\mathcal{G}$. Please note that $\mathcal{E}_{out}(\mathcal{C}, \mathcal{G})$ consists of one *outer polygon* and $\mathcal{E}_{in}(\mathcal{C}, \mathcal{G})$ may consist of zero or more *outer (pseudo) hole polygon* (see Defn. 2.2.8) depending on the type of the curve $\mathcal{C}$.

The cellular envelope of a DC (curve-shaped object) $\mathcal{C}$, which is rough, not irreducible, and disconnected (since it has uneven thickness and stray pixels) has been shown in Fig. 5.1. Note that, the cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ shown in this figure is for the cell size $g = 8$, and the envelope "tightly encloses" all the points of $\mathcal{C}$ with no points lying outside $\mathcal{E}(\mathcal{C}, \mathcal{G})$. The cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ can be computed with the help of the algorithm MAKE-OIP which outputs the outer polygon and outer (pseudo) hole polygons, as mentioned in Sec. 2.5.3. However, for the digital curve $\mathcal{C}$ we have computed the cellular envelope, which is the set of cells that tightly enclose $\mathcal{C}$, by the method mentioned in Sec. 5.4.1.

5.3 Cellular Straight Segments

There exist several works on constructs, properties, and applications of cell complexes and cellular straight segments (CSS) in which the primal as well as many alternative definitions of CSS are found [Fam and Sklansky (1977), Geer and McLaughlin (2003), Kim (1982), Klette (2000), Klette and Rosenfeld (2004b)]. For example, as indicated in [Klette and Rosenfeld (2004b)], a CSS $\mathbf{C}$ can be defined as the minimal set of cells c specified by a straight line segment $\mathbf{L} \in \mathbb{R}^2$ such that

$$\mathbf{L} \cap c \neq \emptyset, \ \forall \ c \in \mathbf{C}; \tag{5.2}$$

$$\text{and} \qquad \mathbf{L} \subset \mathbf{C}, \tag{5.3}$$

which makes its primal definition.

Another definition of CSS involving the Euclidean metric space is given by Fam and Sklansky (1977), in which it has been shown that a cellular curve **C** is a CSS if and only if there exist a direction θ and a pair of (parallel) lines in $\mathbb{R}^2$ (tangential to and) containing **C**, such that the distance between, and measured in the direction (say, $\theta_\perp$) perpendicular to this pair of lines, does not exceed the distance (along $\theta_\perp$) between the closest pair of parallel lines containing the square formed by $(2 \times 2 =)\, 4$ cells sharing a common vertex.

In a recent work [Geer and McLaughlin (2003)], an Euclidean-free definition of CSS has been given in terms of "fully partitioned (finite) strings" ($S^{(0)}$) and "higher order derived strings" ($S^{(j)} : j \geq 1$), the latter being derived iteratively from the preceding string (i.e., $S^{(j-1)}$) by replacing the majority symbol substrings of $S^{(j-1)}$ by its length, and by deleting the minority symbols of $S^{(j-1)}$. Subsequently, it has been shown that a string $S\,(= S^{(0)})$ represents a CSS, provided the jth order derived string of S exists for all $j \geq 0$.

Alternatively, in the perspective of digital straightness, if we consider the center points of these edge-connected cells as grid points, then it follows that a family of cells is edge-connected if and only if the set of center points of these cells is 4-connected. Thus CSS provides a suitable option — apart from that provided by digital straight line segments (DSS) [Rosenfeld (1974)] — for adjudging the straightness of a curve in the digital plane, as indicated in a contemporary work [Klette and Rosenfeld (2004b)]. A linear off-line algorithm for CSS recognition, based on convex hull construction, is briefly sketched in an earlier work [Kim (1982)]. In this chapter, we have designed an online algorithm to derive the set of CSS's from the cellular envelope (Sec. 5.2) of a curve-shaped object, which cannot be subjected to direct DSS extraction/polygonal approximation owing to its inherent nature of possessing varying thickness, as mentioned in Sec. 6.1.

We have considered the center of each cell for extracting the longest line segment iteratively in (a part of) a cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ corresponding to the given curve $\mathcal{C}$ and given cell size g imposed by the grid $\mathcal{G}$. We have used some digital geometric properties of DSS formulated by earlier authors [Klette and Rosenfeld (2004b), Rosenfeld (1974)]. Before explaining our algorithm, the DSS properties (defined w.r.t. chain codes [Freeman (1961a,b)]) relevant to our work, which were established earlier [Rosenfeld (1974)], and later (see [Klette and Rosenfeld (2004b)]) correlated with the other straightness options such as cellular straightness, are mentioned below[1].

[1]In our work, we have considered 4-connectivity of a DSS, i.e., having chain codes lying in the set $\{0, 2, 4, 6\}$, since the cells in the cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ obtained for the curve $\mathcal{C}$ (Sec. 5.2) are connected

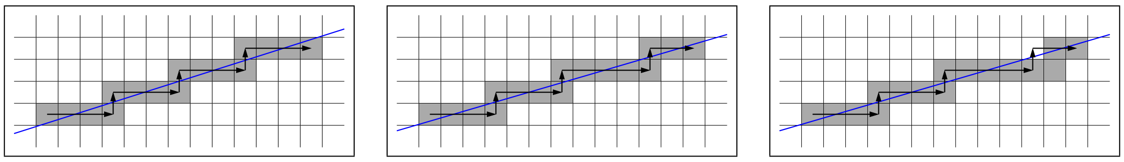


Figure 5.2: Examples of cellular curves explaining the significance of straightness properties (R1)–(R4). Note that the directed black path that traces the ordered set of centers of the cells shows the digital curve (DC) corresponding to a cellular curve. The curves in (a) and (b) are CSS's (the corresponding real lines being shown in blue); but the curve in (c) is not, since there does not exist any real line that can pass through the set of cells defining this curve (see text for explanation).

(R1) The runs have at most two directions, differing by 90^0, and for one of these directions, the run length must be 1.

(R2) The runs can have only two lengths, which are consecutive integers.

(R3) One of the run lengths can occur only once at a time.

(R4) For the run length that occurs in runs, these runs can themselves have only two lengths, which are consecutive integers; and so on.

Few examples of cellular curves/envelopes are shown in Fig. 5.2 to explain the significance of properties (R1)–(R4). For the curve in (a), if we consider the center of each cell as a grid point, as mentioned earlier, then its chain code is $\mathsf{000200020002000} = \mathsf{0}^3\mathsf{2}\mathsf{0}^3\mathsf{2}\mathsf{0}^3\mathsf{2}\mathsf{0}^3$, which consists of codes $\mathsf{0}$ and $\mathsf{2}$ only, and contains consecutive $\mathsf{0}$'s but no two consecutive 2's, thereby satisfying property (R1). Regarding (R2), (R3), and (R4), since there is only one run length (of $\mathsf{0}$'s), this curve trivially satisfies these three properties, and becomes a CSS. Similarly, since the curve in (b) has chain code $\mathsf{0}^3\mathsf{2}\mathsf{0}^3\mathsf{2}\mathsf{0}^3\mathsf{2}\mathsf{0}^2$, which obeys (R1)–(R4), it is a CSS. On the contrary, the curve in (c) has chain code $\mathsf{0}^3\mathsf{2}\mathsf{0}^3\mathsf{2}\mathsf{0}^5\mathsf{2}\mathsf{0}^1$, which satisfies (R1), but violates (R2) as $\mathsf{0}$ has non-consecutive run lengths (3 and 5) — even if we do not consider the leftmost and the rightmost run lengths (which are 3 and 1, respectively), and so it is not a CSS.

In the proposed method for extraction of CSS from the cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$, we have adhered to the properties (R1–R4). In addition, we have considered that also the leftmost and the rightmost run lengths of a CSS should follow property (R2) (which is not

in 4-neighborhood. In a DSS with 8-connectivity, however, the runs would have directions differing by 45^0 as stated in [Rosenfeld (1974)].

```
STEP 1. Initialize each entry in $A_e$ and each entry in $A_c$ with '0'.
STEP 2. DFS-VISIT on $\mathcal{C}$ starting from $p$ using 8-connectivity to reach the nearest cell edge $e_p$ of $\mathcal{G}$.
STEP 3. DFS-VISIT on $A_e$ starting from the entry $A_e(e_p)$ corresponding to $e_p$ in $A_e$ using 4-connectivity (of '1's in $A_e$) to assign:
        '1' to the entry in $A_e$ corresponding to each cell edge $e$ intersected by $\mathcal{C}$, and
        '1' to the entry in $A_c$ corresponding to each of the two cells with $e$ as the common edge.
STEP 4. DFS-VISIT on $A_c$ starting from some cell (e.g., $c_p$, the left adjacent cell of $e_p$) of the cellular envelope formed by the '1's obtained in step 3 using 4-connectivity (of '1's in $A_c$); and check whether the entry $A_c(c)$ corresponding to the cell $c$ currently under DFS-VISIT satisfies at least one of the following two conditions:
        (i) both the left and the right adjacent entries of $A_c(c)$ are '1's;
        (ii) both the bottom and the top adjacent entries of $A_c(c)$ are '1's.
        If (i) or/and (ii) is/are true, then terminate the DFS-VISIT, since the current cell $c$ lies either on a horizontal edge/part (when (i) satisfies) or on a vertical edge/part (when (ii) satisfies) of the cellular envelope of $\mathcal{C}$; and declare $c$ as the seed cell $c_0$ for stage II.
STEP 5. If no seed cell $c_0$ is found in step 4, then the cell size is not sufficiently large compared to the (minimum) thickness of the input curve $\mathcal{C}$. Hence the user may be asked to increase the cell size (i.e., grid separation $g$); alternatively, an arbitrary cell of the envelope may be considered to be the seed cell $c_0$.
```

Figure 5.3: Algorithm FIND-CELLULAR-ENVELOPE $(\mathcal{C}, \mathcal{G}, p)$ in stage I.

mandatory as suggested by Rosenfeld (1974)).

5.4 Polygonal Approximation Using Cellular Envelope

The method of finding the (cellular) polygonal approximation of a curve-shaped object $\mathcal{C}$ consists of two stages, namely stage I and stage II. In stage I, we construct the cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ based on the novel concept of combinatorial arrangement of the cells containing $\mathcal{C}$. In stage II, we analyze the cells of $\mathcal{E}(\mathcal{C}, \mathcal{G})$ to extract the straight pieces from $\mathcal{E}(\mathcal{C}, \mathcal{G})$, considering the center of each cell of $\mathcal{E}(\mathcal{C}, \mathcal{G})$ as a grid point and using the properties (R1)–(R4), as mentioned and explained in Sec. 5.3. We have designed and implemented two algorithms, one for each stage, which are briefed up next.

5.4.1 Stage I: Finding the Cellular Envelope

We consider any point $p \in \mathcal{C}$ as the start point defining the object $\mathcal{C}$. For the time being, consider that $\mathcal{C}$ is connected in 8-neighborhood. Then using DFS-VISIT (Depth First

Search algorithm [Cormen *et al.* (2000)]), we can reach the nearest edge e_p of a cell that intersects $\mathcal{C}$. Starting from e_p, using DFS-VISIT on the edges of the cells, we visit those cell edges that are intersected by $\mathcal{E}$; this procedure helps us in constructing the edge matrix A_e and the cell matrix A_c (Sec. 5.2), which are finally used to obtain $\mathcal{E}(\mathcal{C}, \mathcal{G})$. The major steps of the algorithm FIND-CELLULAR-ENVELOPE $(\mathcal{C}, \mathcal{G}, p)$ to find the cellular envelope of a connected (and of uniform or non-uniform thickness) object $\mathcal{C}$ w.r.t. the cellular array imposed by the grid $\mathcal{G}$ are given in Fig. 5.3.

In the case $\mathcal{C}$ has some missing points/pixels, i.e., if it suffers from disconnectedness, then it may happen that none of the edges of a cell is intersected by $\mathcal{C}$, although $\mathcal{C}$ is contained in that cell. To circumvent this problem, we have to directly construct the cell matrix A_c, without constructing A_e, which would, however, increase the time complexity (and the run time, thereof) of stage I. It may be noted that, if the curve possesses too much gap/disconnectedness, so that the gap is even larger than the cell size, then this gap may result in creating a discontinuity (in the edge-connectivity) of the cells constituting the envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$, which is then fragmented into two or more pieces, thereby producing faulty results. Choosing an appropriate cell size is, therefore, necessary to obtain the desired cellular envelope of a disconnected DC in stage I.

5.4.2 Stage II: Finding the Cellular Straight Segments

In stage II, the algorithm FIND-CSS $(\mathcal{E}, c_0)$[1], given in Fig. 5.4, extracts the ordered set of CSS's from the cellular envelope $\mathcal{E}$, as follows. W.l.o.g., since in stage I, the seed cell c_0 lies on a horizontal part (or on a vertical part, or on a thick part) of $\mathcal{E}$, we negotiate two traversals (STEP 1) — one towards left and the other towards right of (center point of) c_0 — to obtain two CSS's with complying (cellular) straightness such that the sum of their lengths is maximal, and merge these two to get the first CSS, C_1, to be included in the ordered set T of terminal cells (STEP 2). The starting cell for extracting the next CSS (STEP 3) from the cellular envelope is, therefore, considered to be the right terminal cell c_1 of C_1. We use the algorithm DFS-VISIT [Cormen *et al.* (2000)] to explore the cells constituting the envelope and to extract the CSSs, whose terminal cells are finally reported in T.

Time complexity. If N be the number of points defining the curve $\mathcal{C}$, then its envelope $\mathcal{E}$ consists of $O(N/g)$ cells. As we have used DFS-VISITs, the time complexity in stage I

[1]Now onwards, we denote the cellular envelope of $\mathcal{C}$ by $\mathcal{E}$ for simplicity.

```
STEP 1.  Traverse (cell-wise) towards left and towards right from $c_0$ to extract all possible pairs of CSS
         starting from $c_0$, such that
         (i) the chain code of each CSS, and
         (ii) the combined chain code of the two CSS's
         in each pair are in conformity with properties (R1)–(R4);
STEP 2.  Find a/the pair of CSS that has maximum sum of lengths;
         merge this pair into a single CSS, namely $C_1$;
         declare $c_0$ and $c_1$ as the left and the right terminal cells of $C_1$;
         store (the centers of) $c_0$ and $c_1$ in the ordered set $T$.
STEP 3.  Start from $c_1$ to extract the next (longest) CSS, $C_2 := (c_1, c_2)$, with terminal cells $c_1$ and $c_2$;
         store $c_2$ in T; and mark the cells defining $C_2$ as VISITED.
STEP 4.  Repeat STEP 3 starting from the last entry (i.e., terminal cell) in $T$ to get the CSS's defining
         $\mathcal{E}$ until all cells of $\mathcal{E}$ are VISITED (using DFS-VISIT).
         Note: (i) If a CSS has both its terminal cells in the 4-neighborhood of another (longer)
         CSS, then the former (shorter) CSS is not included in $T$ (Fig. 5.5(a)). (ii) For a
         bifurcating/branching CSS, we store both its terminal cells in $T$ (Fig. 5.5(b)).
STEP 5.  Declare $T$ as the polygonal approximation of the cellular envelope $\mathcal{E}$.
```

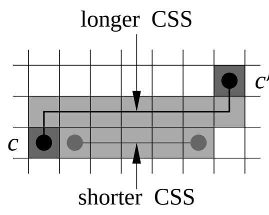
Figure 5.4: Algorithm FIND-CSS $(\mathcal{E}, c_0)$ in stage II.

is bounded by $O(N/g)$. In stage II, extraction of each CSS C_i takes $O(|C_i|)$ time, where $|C_i|$ is the number of cells defining C_i. Hence, the time complexity to extract all CSS's in step II is $O\left(\sum |C_i|\right) = O(N/g)$, which gives the total time complexity of **PACE** as $O(N/g)$.

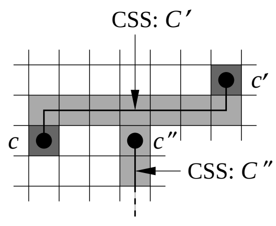
5.4.3 Efficiency of the Algorithm

The deviation of the approximate polygon(s) (or polychain(s)) from the input set of digital curves determines the efficiency of a polygonal-approximation algorithm. Two such measures to assess the quality of approximation corresponding to a curve $\mathcal{C}$ are (i) *compression ratio* CR $= N/M$ and (ii) *the integral square error* (ISE) between $\mathcal{C}$ and P, N being the number of points in the (thinned) curve $\mathcal{C}$ and M being the number of vertices in the approximate polygon P. It may be noted that there is always a trade-off between CR and ISE [Held *et al.* (1994), Rosin and West (1995), Sarkar (1993)].

The proposed algorithm is not constrained by the number of vertices of the output polygon. Hence, we cannot use a measure of approximation with M as an input parameter. Since the precision of the cellular envelope $\mathcal{E}$ corresponding to $\mathcal{C}$ depends on the *cell*



(a) A short CSS with each of its terminal cells lying at 4-N of a longer CSS is not considered as a valid CSS (*Note* (i) of STEP 4 in Fig. 5.4).



(b) For a branching CSS, C'', each of its terminal cells (one is c'' and the other not shown) is stored in T (*Note* (ii) of STEP 4 in Fig. 5.4).

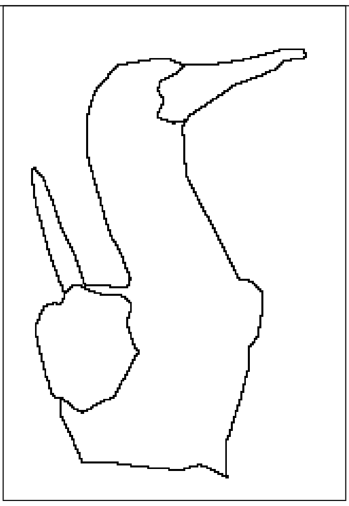
Figure 5.5: Inclusion and exclusion of terminal cell(s) of CSS in T.

size, namely g, which, in turn, decides the quality of approximation, the *approximation parameter* of our algorithm is considered as g, depending on which M would change. A high value of g is likely to produce a slacked approximation, whereby M decreases accordingly, whereas a low value of g is expected to output a tight approximation requiring a higher number of vertices to constitute P. Thus, the number of vertices, M, is a measure of the quality of approximation for a given grid size, g. Hence, we measure the efficiency of our algorithm using the compression ratio (CR) versus g.

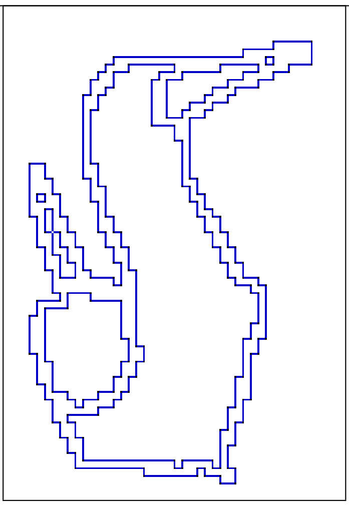
As stated earlier, only CR does not reflect the overall efficiency of approximation, since a trade-off lies between CR and ISE. Hence, apart from CR, we also find the deviation, $d_\perp(p \rightarrow p') := \max\{|x - x'|, |y - y'|\}$, of each point $p(x, y) \in \mathcal{C}$ to its corresponding (nearest) point $p'(x', y') \in P$ for the chosen grid size, g. For all points in $\mathcal{C}$, the overall error of approximation is measured by the frequency $f(d_\perp)$ of the number of points having deviation $d_\perp$ versus $d_\perp$. Further, since $d_\perp$ depends on g in our algorithm, the fraction of the number of points in $\mathcal{C}$ with deviation $d_\perp$ varies with g. Hence, for a given value of g, the *error frequency* is estimated as

$$f(\delta) = \frac{1}{N} \left|\{p \in \mathcal{C} : d_\perp(p \rightarrow p') = \delta\}\right|. \tag{5.4}$$

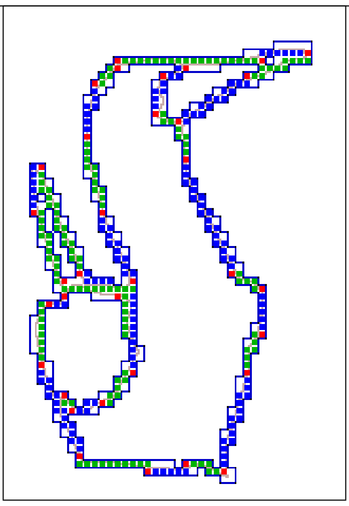
The variation of $f(d_\perp)$ versus $d_\perp$, considering the given value of g, acts as the second measure that provides the error distribution for the polygonal approximation of $\mathcal{C}$. The plots on (i) CR versus g and (ii) $f(d_\perp)$ versus $d_\perp$ on polygonal approximation of some real-world digital curves are given in Sec. 5.5.



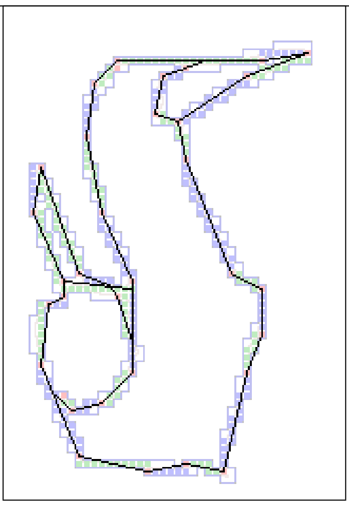
(a) A curve-shaped digital object $\mathcal{C}$ of nonuniform thickness representing the edge map of "duck" image. Since the curve is not one-pixel thick, the conventional algorithms on polygonal approximation cannot be applied on it.



(b) Cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ obtained in stage I of the algorithm **PACE**. Note that, the cells of the envelope are connected in 4-neighborhood, which are, therefore, 4-cells.



(c) The set of CSS's extracted in stage II from the envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$ shown in (b). The CSS's have been alternately colored in blue and green with the terminal cell of each CSS in red.



(d) Final polygonal approximation (in thin black lines) superimposed on the (faded) cellular envelope $\mathcal{E}(\mathcal{C}, \mathcal{G})$.

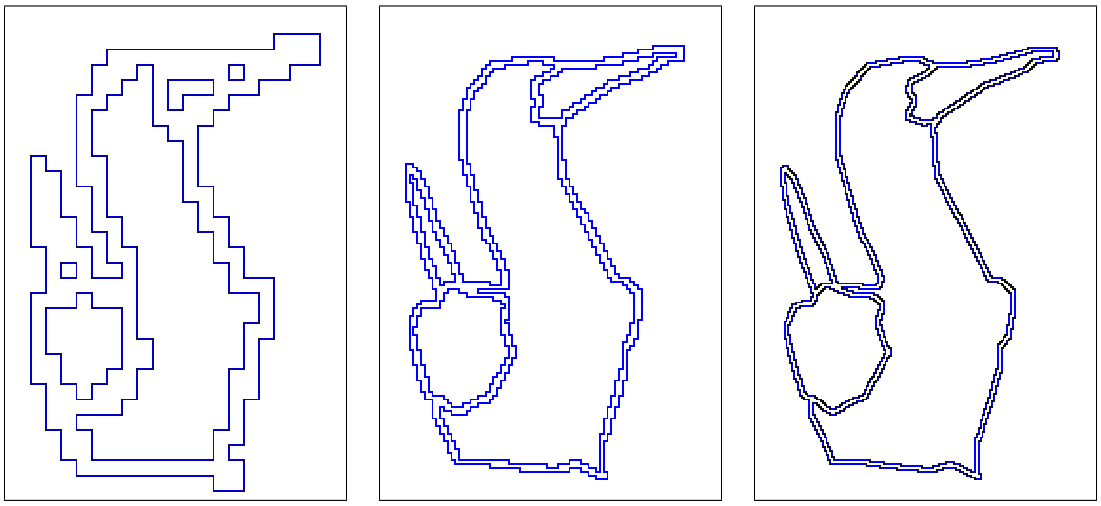
Figure 5.6: Results of algorithm **PACE** for cell size $g = 4$ on "duck".

5.5 Experimental Results

We have implemented the two algorithms, namely FIND-CELLULAR-ENVELOPE and FIND-CSS, that make the proposed method **PACE** for polygonal approximation of an arbitrarily thick DC, in C in SunOS Release 5.7 Generic of Sun_Ultra 5_10, Sparc, 233 MHz. We have tested the algorithms on various digital curves of arbitrary shape, changing thickness, and irregular connectedness, some of which are presented in this thesis. It may be mentioned that, no other earlier work on cellular polygon appears to exist, and hence, we could not have performed a comparative study of our method with a thinning-free method. However, to demonstrate the strength and efficiency of our algorithm, we have given the results of polygonal approximation (on thinned curves) by a couple of existing methods [Teh and Chin (1989), Wall and Danielsson (1984)]. In order to do this, we have first applied the thinning procedure [Rosenfeld and Kak (1982)] on a thick curve (or a set of curves) so that the thinned curve can be used as input in these existing algorithms (Fig. 5.9).

The result for an edge map (non-thinned) of "duck" is shown in Fig. 5.6, which testifies the elegance of **PACE** in deriving the cellular polygon corresponding to a DC. It may be noticed in this figure that, some of the cells in the envelope $\mathcal{E}$ have not been included in any CSS; because in the algorithm FIND-CSS, we have considered the (terminal cells of) each locally longest CSS to be included in P (see the *Note* in STEP 4). But when there is a bifurcation/self-intersection (e.g., in and around the root of its tail) or a sharp bend (e.g., at the tip of its beak), the cellular envelope (Fig. 5.6(b)) contains several cells across its thickness, which may cause error in the polygonal approximation as observed in Fig. 5.6(d) in the part of the polygon corresponding to the region in and around the tail root. Hence a proper value of the cell size, g, is mandatory to ensure a good cellular envelope corresponding to a DC, and a good polygonal approximation thereof.

To illustrate the role and significance of g, few other results on the image "duck" have been given in Fig. 5.7. It is evident from this figure that, for $g = 8$, the approximation deteriorates relative to the one corresponding to $g = 4$ (Fig. 5.6), owing to the fact that a larger value of g imparts a greater tolerance of approximation by slackening the cellular envelope corresponding to the digital curve. A larger value of g is advantageous when fewer vertices are desired in order to reduce the output complexity, by compromising with a lower quality of approximation. For example, the number of output vertices corresponding to the image "duck" for $g = 8$ is $M(g = 8) = 19$ (Fig. 5.7), which is significantly less than that corresponding to $g = 4$, i.e., $M(g = 4) = 34$ (Fig. 5.6) — a fact that certifies the desired behavior of a polygonal-approximation algorithm [Bhowmick and Bhattacharya (2007),



Cellular envelopes of the set of thick curves (Fig. 5.9) representing the image “duck”.

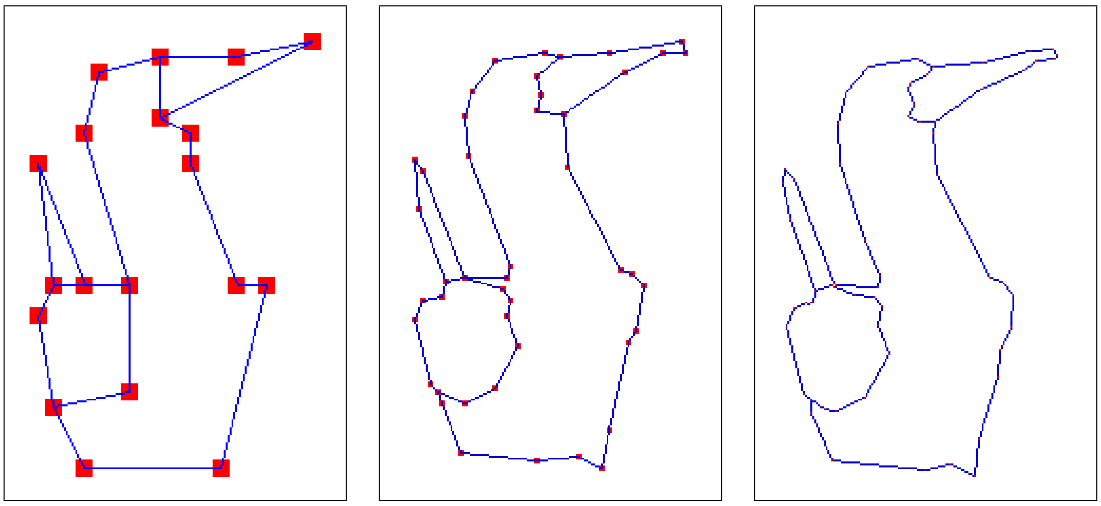


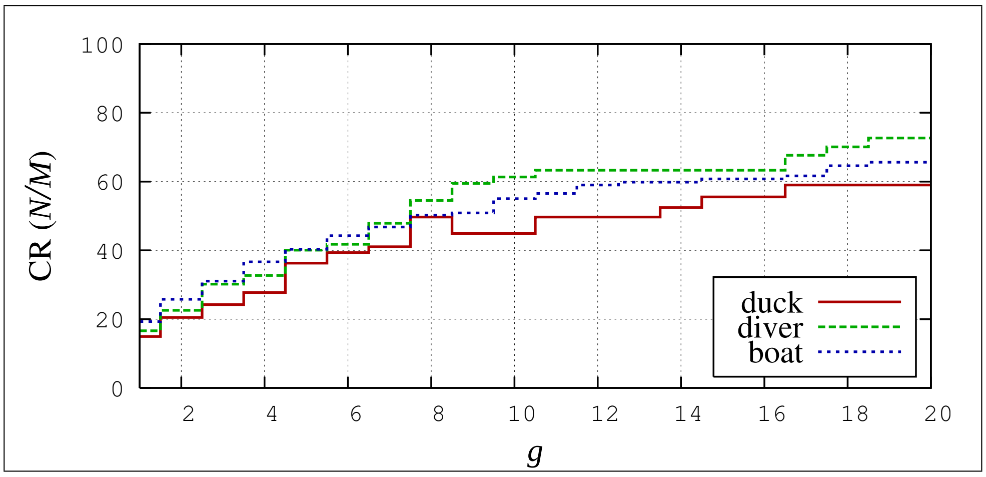
Figure 5.7: Polygonal approximation of the image “duck” for few other grid sizes ($g = 8, 2, 1$ from left to right) shows how the quality of the output polygon goes on improving with decreasing grid size, although at the cost of increasing number of vertices of the polygon. Note that the vertices of a polygon are highlighted as a red square, the square-size being g, and the edges are shown in blue.

Rosin (1997)]. For $g = 2$, the quality of approximation (measured in terms of the deviation of an approximate polygon from the original curve) improves relative to that corresponding to $g = 4$; and for $g = 1$, the quality of an approximate polygon improves further. However, as evident from Fig. 5.7, the number of vertices ($M(g = 2) = 45, M(g = 1) = 63$) of the approximate polygon also increases with its improving quality.

Comparisons of the algorithm **PACE** with two well-known algorithms, namely **AD** based on *area deviation* [Wall and Danielsson (1984)] and **CM** based on *curvature maxima* [Teh and Chin (1989)], are presented in Fig. 5.9 and in Table 5.1. Since these algorithms need irreducible/thinned digital curves as input, a thinning algorithm [Rosenfeld and Kak (1982)] is applied on the set of thick curves constituting the image "duck" as shown in the figure. The set of thinned curves are then considered as input to these algorithms. It is evident from this figure and Fig. 5.7 that for a high value of g (e.g., 4), the approximate polygons produced by the algorithm **PACE** are qualitatively inferior to those obtained by area deviation or by curvature maxima. However, the quality of the approximate polygons by the proposed algorithm improves on lowering the value of g. In particular, for $g = 1$ the resultant polygons are qualitatively comparable with the ones produced by the two other algorithms, as evident from Fig. 5.9.

The efficiency of approximation is measured in terms of the compression ratio and the error frequency as explained in Sec. 5.4.3. The related plots in the form of two sets of histograms for the image "duck" and two other real-world thick curve sets, namely the images "diver" and "boat", are shown in Fig. 5.8. The respective results of polygonal approximation on "diver" and "boat" are presented in Fig. 5.11 and Fig. 5.12 to demonstrate the goodness and efficiency of the algorithm in the case of digital curves with varying thickness values and arbitrary intersections. For each of the three curve sets, the trade-off between the compression ratio (CR) and the approximation parameter (i.e., the grid size, g) is evident from the set of plots on CR versus g. The set of histograms on the error frequency $f(d_\perp)$ versus the error/deviation $d_\perp$ corresponding to a digital curve $\mathcal{C}$ depicts that majority of the erroneous points (with $d_\perp > 0$) have their errors/deviations in the lower range of $[0, g]$, indicating that the approximate polygon lies close to the original (thinned) curve. A relatively much smaller fraction of these erroneous points have high deviations, i.e., deviations nearing g. For example, the polygonal approximation of the image "duck" for $g = 1$ produces $f(d_\perp = 0) = 89\%$ (no error), $f(d_\perp = 1) = 11\%$, and $f(d_\perp > 1) = 0\%$; for $g = 2$, the corresponding error frequencies are $f(d_\perp = 0) = 72\%$, $f(d_\perp = 1) = 26\%$, $f(d_\perp = 2) = 2\%$, and $f(d_\perp > 2) = 0\%$; and so forth (Fig. 5.8). More importantly, this is true for each of the three images, which characterizes the non-dependence of the algorithm on the structure and composition of the set of input curves.

One of the salient features of the algorithm **PACE** is its ability to produce effective polygonal approximation even for a noisy digital curve, which, being affected by noise, is likely to be disconnected. A noisy curve can be thought of as a pattern of points located



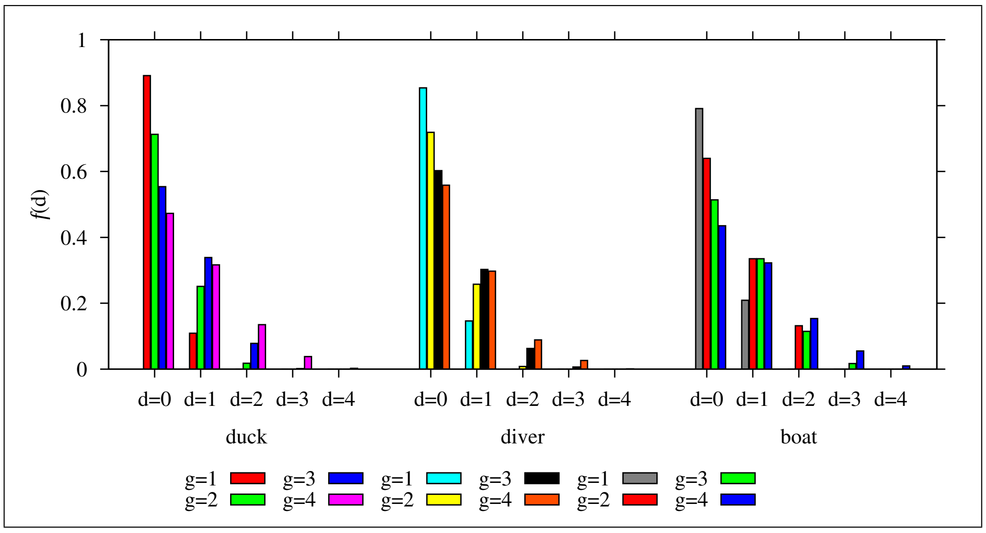


Figure 5.8: Plots on quality of approximation for the images “duck”, “diver”, and “boat”. The top set of plots shows three histogram profiles on the compression ratio, $CR = N/M$, versus the approximation parameter, i.e., cell size g. The bottom set is made of the histograms on the error frequency $f(d_\perp)$ versus the error of approximation, $d_\perp$, for $g = 1, 2, 4$.

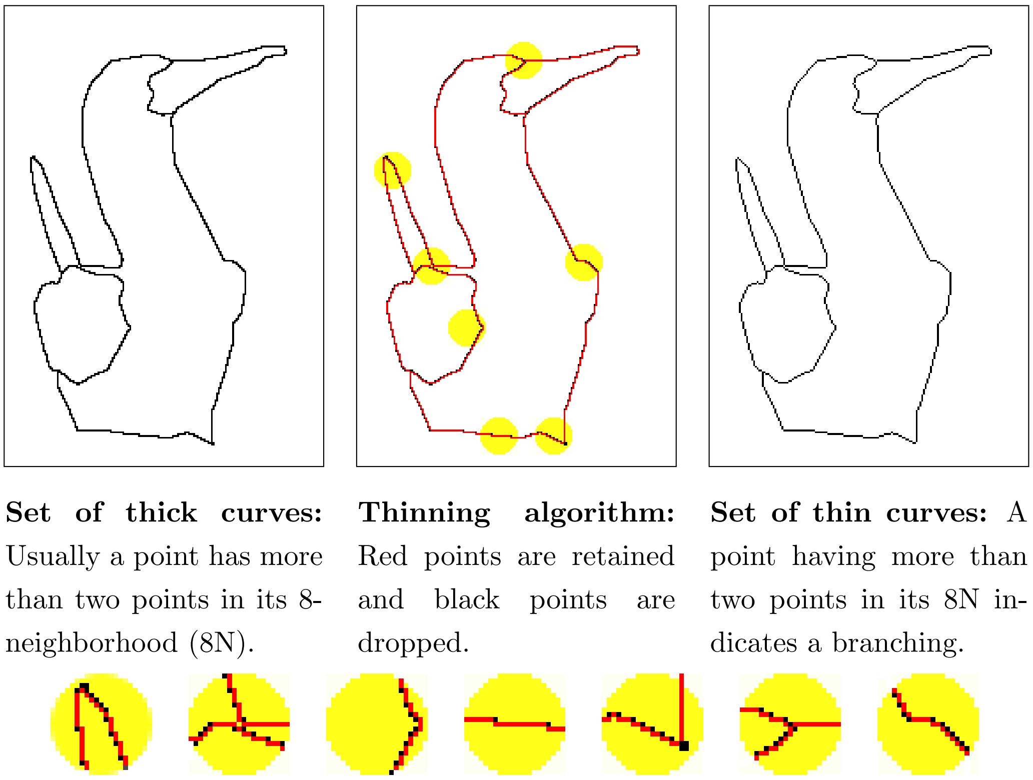
densely along a curve, which are usually disconnected at irregular intervals, although giving the impression of a curve-like pattern due to distribution of the constituting points on or near the actual (noise-free) curve. The existing algorithms for polygonal approximation do not have the provision of working with such noisy curves as input. The set of noisy curves corresponding to the image “duck” and the results of polygonal approximation on it are shown in Fig. 5.5. Since, for a small value of g (e.g., $g = 1$ or 2), the resultant polygonal envelopes are likely to possess small polygons, which contain the noisy/spurious points of the curve as evident from the figure, a proper value of g is required for applying the proposed algorithm. For example, for $g = 4$, the corresponding polygonal envelope successfully covers all points of the noisy curve, including the spurious points, and this property enables the algorithm to derive the polygonal approximation to the desired level of precision.

Another feature of the proposed method is the inherent nature of Euclidean-free metrics and operations involved in both the stages. This leads to high execution speed to the implementation of **PACE**, which is reflected in the respective CPU times shown in Table 5.1. As evident from this table, the algorithms **AD** and **CM** are considerably slower than **PACE** owing to the usage of only primitive integer operations (comparison, addition, and increment) in the latter. On the contrary, the former algorithms use more complex operations, e.g., multiplication and division, which need computation in the real domain, for realization of certain geometric and trigonometric functions as explained in Sec. 5.1.1. Further, with the increase in the cell size g, the number of output vertices of **PACE** decreases significantly. As a result, the compression ratio (CR) improves consistently, but the quality of approximation deteriorates, as evidenced by the quality measures in Fig. 5.8. As stated earlier, this indicates that the cell size g should be suitably chosen to get an acceptable trade-off in the approximation.

5.6 Conclusion

We have presented a novel concept of approximating a curve-shaped digital object by its cellular envelope. The algorithm is marked by its **(i)** insensitivity with regard to thickness variation of the input DC, **(ii)** use of a combinatorial approach to construct the optimum cellular envelope for the given DC, **(iii)** use of straightness properties inherited from digital geometry, **(iv)** non-dependence on the Euclidean paradigm, and **(v)** implementation without using any floating point operation, which collectively make it robust, speedy, and efficient.

Although the cellular envelope does not remain entirely unaffected when a different registration of the curve $\mathcal{C}$ w.r.t. the grid (translation) is chosen, the cellular polygon produced by the subsequent CSS extraction process remains almost invariant. Experimenting on the nature of variation of the cellular envelope and the resulting polygon of a DC with

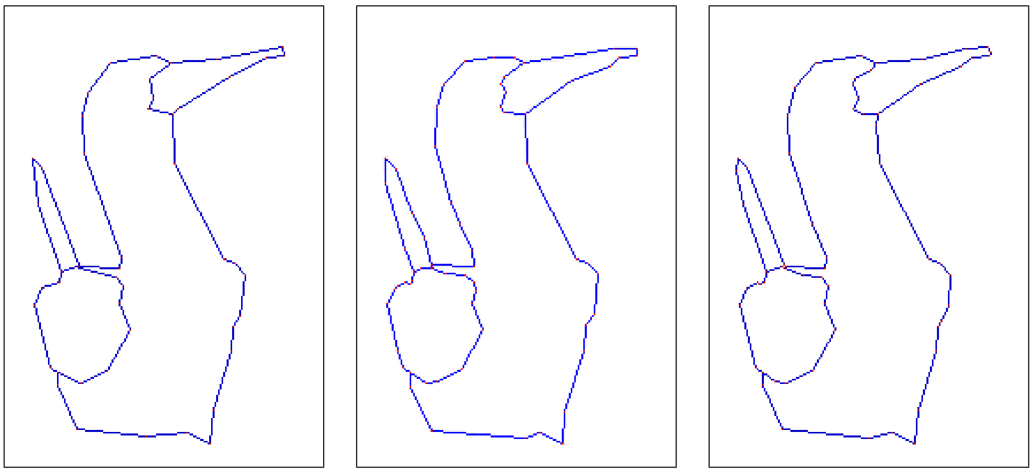


Set of thick curves: Usually a point has more than two points in its 8-neighborhood (8N).

Thinning algorithm: Red points are retained and black points are dropped.

Set of thin curves: A point having more than two points in its 8N indicates a branching.

Magnified views of thinning results corresponding to seven portions (colored yellow).



area deviation (Wall and Danielsson (1984)): $M = 47$.

curvature maxima (Teh and Chin (1989)): $M = 64$.

thinning-free method (proposed): $M(g = 1) = 63$.

Figure 5.9: Result-wise comparison of the proposed thinning-free method with the existing thinning-based methods for "duck" image.

Table 5.1: Comparison of the proposed algorithm **PACE** with the algorithms **AD** using *area deviation* [Wall and Danielsson (1984)] and **CM** using *curvature maxima* [Teh and Chin (1989)].

Image name & size	$\lvert\mathcal{C}\rvert$ = # points		$\lvert P\rvert$ = # vertices					Total CPU time (secs.)				
			AD	**CM**	**PACE**			**AD**[a]	**CM**[a]	**PACE**[b]		
	thick	thin			$g=1$	$g=2$	$g=4$			$g=1$	$g=2$	$g=4$
duck 145×227	**1267**	944	47	64	63	45	34	2.717	4.201	0.493	0.283	0.163
diver 333×224	**1789**	1336	69	82	87	63	46	3.733	5.557	0.552	0.328	0.192
boat 364×133	**3319**	2565	97	114	121	92	69	5.470	7.539	0.892	0.507	0.297
India 245×276	**1471**	1173	116	132	137	96	71	5.882	8.421	0.863	0.541	0.300
pyramid 480×198	**2361**	1795	63	67	74	57	43	3.980	4.955	0.672	0.370	0.285

[a]Total CPU time includes the time required for thinning the original curve plus the time required for polygonal approximation of the thinned curve.
[b]Total CPU time includes the time for computing the cellular envelope plus the time required for polygonal approximation the cellular envelope.

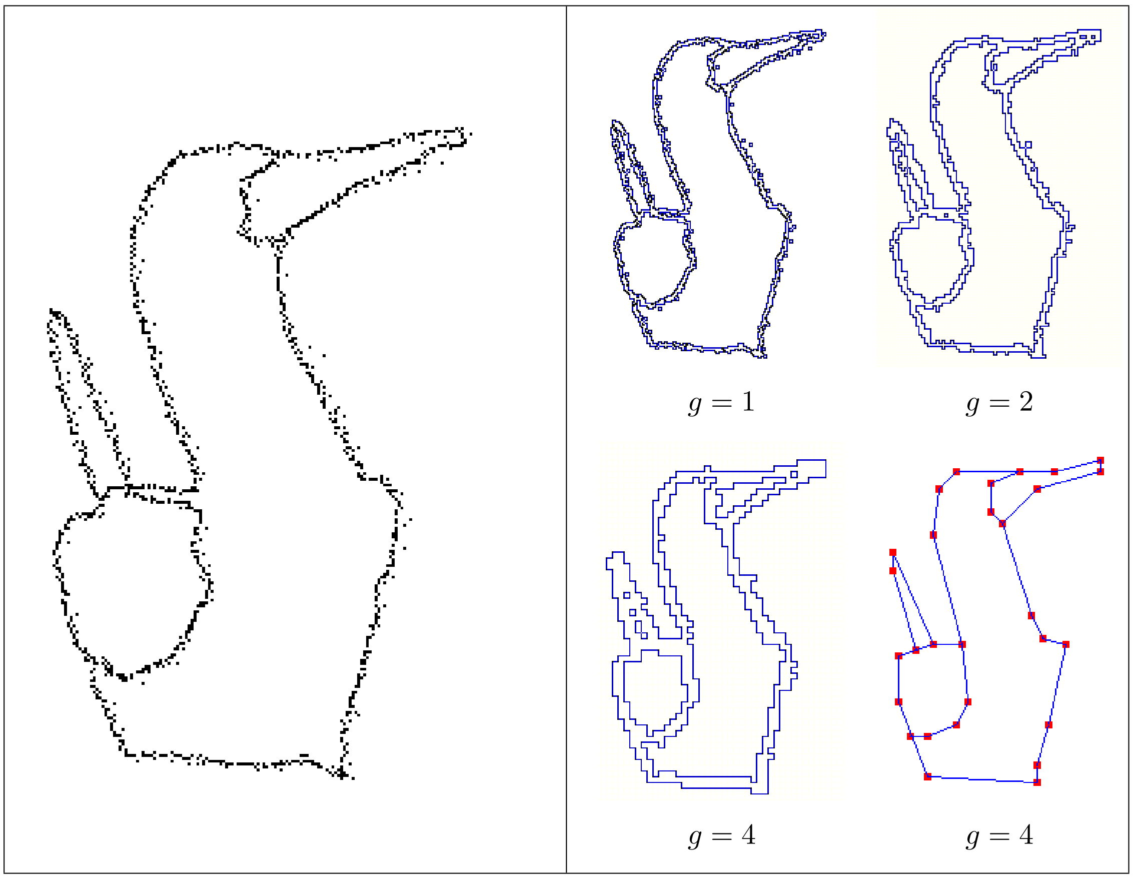


Figure 5.10: Polygonal approximation of the image "duck" after being injected by salt-and-pepper noise. For a small grid size ($g = 1$ or 2), the resultant cellular envelope cannot cover the underlying object, which is no longer a single connected component. For an appropriately large value of g, however, the object lies entirely in the cellular envelope, thereby making it suitable for polygonal approximation by the proposed method.

its registration (both translation and rotation) w.r.t. the underlying grid, therefore, stands as a possible extension of this work. Placement of a DC with proper orientation w.r.t. the grid in order to obtain its optimal cellular envelope is really a challenging problem, which is not yet addressed. The anisotropic nature of the digital plane, and the apparently unpredictable behavior of the cellular envelope of a shifting/rotating object, calls for innovative digital-geometric techniques to solve these problems.

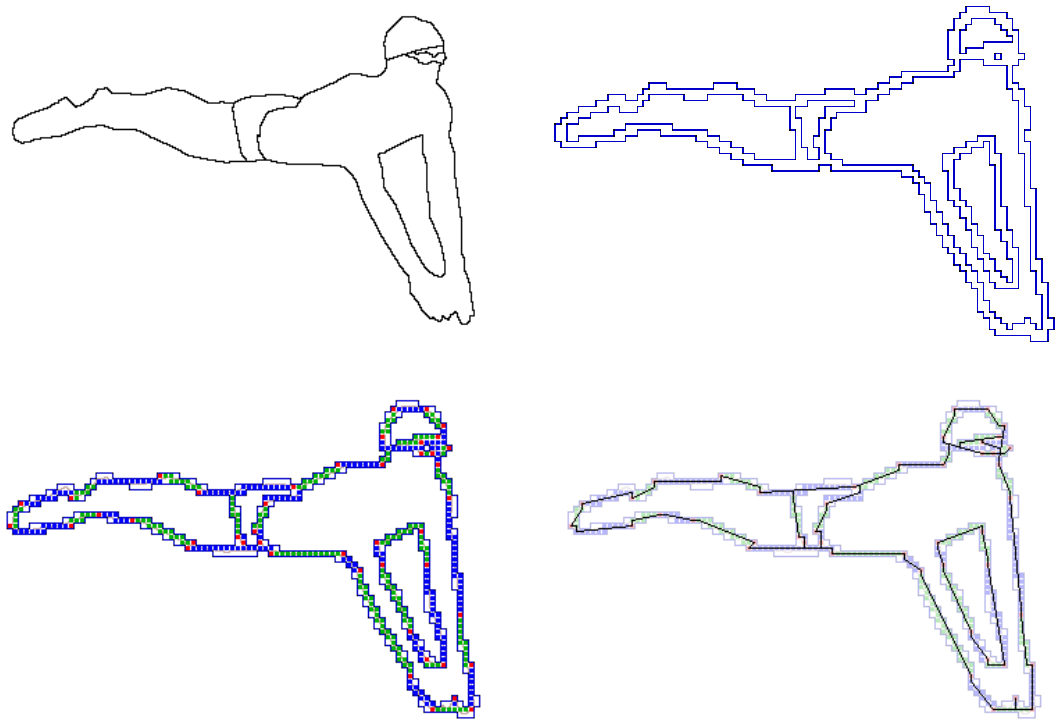


Figure 5.11: Results on “diver” for $g = 4$. See text and Fig. 5.6 for explanation.

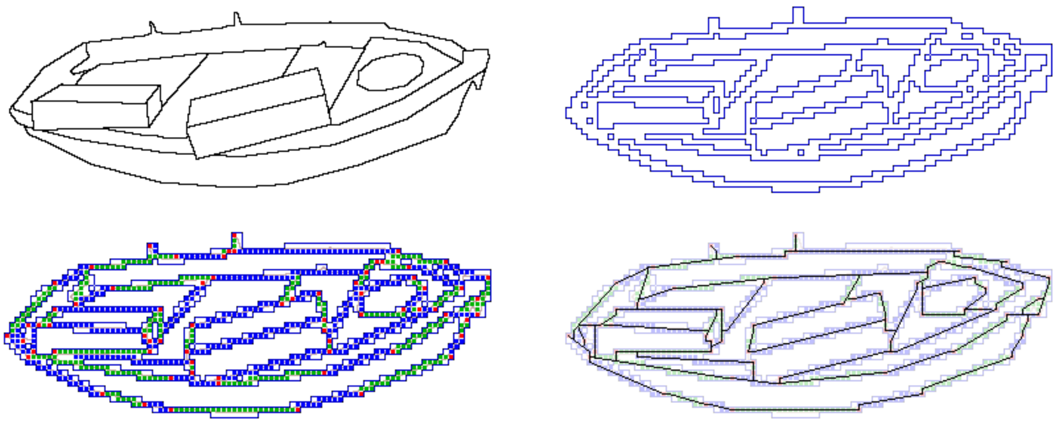


Figure 5.12: Results on “boat” for $g = 4$. See text and Fig. 5.6 for explanation.

Chapter 6

From a Digital Object to its Pointillist Ensemble

6.1 Introduction

Object representation is an essential and crucial task required in many areas of scientific analysis related with experimental psychology and computer vision. Proper representation of an object aids and eases the subsequent application involving its shape and associated information [Feldman (2000), Hyde *et al.* (1997), Zunic and Rosin (2003)]. Typical applications of description, analysis, and matching of objects of diversified nature and of various shapes can be found in a wide range of literature that includes art, architecture, cartography, cell biology, neuron morphology, psycholinguistics, qualitative reasoning, robotic vision, satellite imagery, etc. [Irvin and McKeown (1989), Ling and Jacobs (2007), Liow and Pavlidis (1990), Noronha and Nevatia (2001), Sonka *et al.* (1993), Willats (1997)]. Hence, for sampling a shape in general, and a (real/digital) curve in particular, and for reconstruction of the shape from its sample, various general and application-specific algorithms had been proposed over the last three decades for solving various scientific and engineering problems. Apart from sampling and reconstruction algorithms for curves in 2D and for surfaces in 3D, several definitions, procedures, and methods have been also proposed in recent times to describe, identify, and measure shape-related information present in a digital image [Brimkov and Klette (2008), Díaz-Baqez and Mesa (2001), Lachaud *et al.* (2007), Latecki *et al.* (1995), Ling and Jacobs (2007), Rosin (1999), Zunic and Rosin (2003)].

The work in this chapter addresses the problem of describing a 2-dimensional digital object by an *unordered set* of points in the digital plane, in order to retain the underlying shape information associated with the object from a visual perspective, and at the same time, to enable an unambiguous process for reconstructing the original object. For a

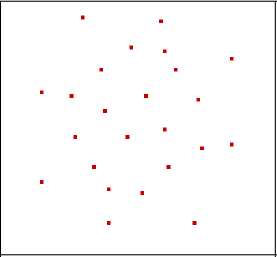
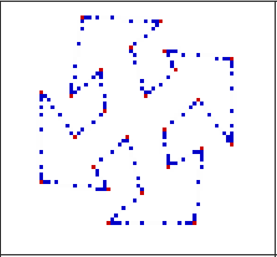
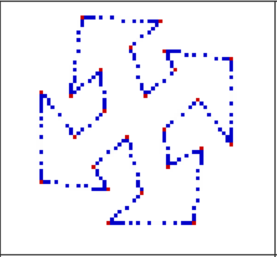
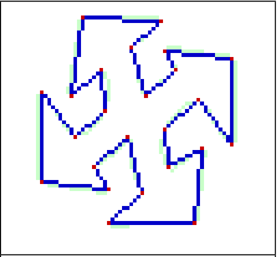
			
(a) Vertex set, $\mathcal{P}$, of a digital polygon.	(b) Ensemble for minimum pointillist factor ($\phi = 1$).	(c) Ensemble for a larger pointillist factor ($\phi = 2$).	(d) Actual polygon.

Figure 6.1: The vertices of a digital polygon, as a mere point set as in (a), cannot create a definite impression of the underlying object/polygon (d). However, with the proposed method, the (blue) pseudo-vertices along with the (red) polygon vertices (b) reflect the actual object, which becomes more prominent (c) when the pointillist factor, ϕ, is increased. The method is more effective when the polygon is sufficiently large in size and complexity.

digital object, given as a set C of digital curves, each constituent digital curve $\mathcal{C}$ in the input set C is first decomposed into its polygonal (closed curve) or poly-chain (open curve) approximation, thereby obtaining an ordered set of vertices $\mathcal{P}$ corresponding to the curve $\mathcal{C}$. Without the order — which is not reflected when the vertices are shown as mere points in $\mathbb{Z}^2$ — the set $\mathcal{P}$, however, carries almost no information about the actual object that it represents, as shown in Fig. 6.1. In particular, the shape information of the object is out of the preceptive ability of a visionary mechanism — whether a human or a computer — if the set $\mathcal{P}$ is sufficiently large in size. The underlying object and its shape details are cognizable only if the definite order is imposed on these points/vertices to define the polygon(s) corresponding to the digital curve(s) that constitute the object. Thus, an unordered set of vertices of the polygon hardly begets any idea about its shape and related geometric structure and topological orderliness, thereby making it unusable for describing a curve-shaped digital object in a meaningful way.

In order to circumvent the above problem and to map a digital object to an optimal or a suboptimal set of points in the digital plane, we explore the idea of representing a digital polygon $\mathcal{P}$ not as an ordered set of vertices[1], but as an unordered set of points, which is called the *pointillist ensemble* of $\mathcal{P}$ and denoted by $\widehat{\mathcal{P}}$. The ensemble $\widehat{\mathcal{P}}$ contains points selected appropriately from the set of digital edges of $\mathcal{P}$ using the *pointillistic approach*

[1]For sake of notational simplicity, we use $\mathcal{P}$ to denote a polygon as well as the (ordered) set of its vertices, depending on the context.

in an algorithmic way (Fig. 6.1). The pointillistic approach is newly introduced in this thesis, which, when used scientifically and efficiently, is shown to produce the desired impression of a digital object with the reduced set of object points. More importantly, the ensemble $\widehat{\mathcal{P}}$ is such that its points can be used to retrace the original polygon $\mathcal{P}$ using a reconstruction algorithm. It may be mentioned here that, in the history of the visual art, *pointillism* refers to the Neo-Impression genre of painting with points/dots with an aim to achieve an artistic-cum-scientific way of conveying the desired visual impression. The technique, accredited for its development to certain serious pointillists like Seurat and his contemporaries late in the 19th century France [Gage (1987), Seurat (1966)], therefore, relies on the perceptive ability of the eye and the subconscious analytical mind of the viewer to perceive the collection of dots as a fuller form.

The proposed scheme of digital-object representation is based on the polygonal decomposition of the object followed by the aforesaid pointillistic approach. It has several advantages, some of which are as follows:

- Polygonal decomposition of an arbitrary digital curve $\mathcal{C}$ smoothens any unwanted jaggedness of $\mathcal{C}$ as per the desired limit without losing its overall geometric orderliness, and represents the curve as an ordered set, $\mathcal{P}$, of its vertices. The set $\mathcal{P}$ is used to find the pairwise nearest edge (and hence the geometric configuration of the underlying curve ($\mathcal{C}$)), which is required to find the corresponding pointillist ensemble.

- The pointillist ensemble of $\mathcal{C}$, as obtained from $\mathcal{P}$, using the proposed algorithm, consists of an optimal or suboptimal number of points judiciously selected from the digital edges of $\mathcal{P}$, which, when visualized, gives a correct impression about the original curve.

- The algorithm for reconstruction of the curve from its ensemble finds the nearest ensemble point corresponding to each ensemble point using an appropriate data structure. The nearest neighbor rule has two-fold advantages: (i) it mimics our psycho-visual mechanism; (ii) it picks an optimal or suboptimal number of points for which the reconstructed curve resembles the original curve quite accurately.

- The algorithm is fast, robust, and efficient. It outputs the desired ensemble for an arbitrary digital curve — whether open or closed — whether a synthetic curve or a curve set representing a real-world digital object — as implemented and tested by experiments, some of which are presented in Sec. 6.4.

The rest of the chapter is organized as follows. In Sec. 6.2, we discuss briefly some of the existing works related with curve sampling and reconstruction. We start with the problem formulation and its physical interpretation in Sec. 6.3.1. In Sec. 6.3.2, we briefly explain the proposed method to obtain a polygonal/poly-chain approximation of a digital curve, $\mathcal{C}$, constituting an object in the digital plane. Sec. 6.3.3 puts forth the algorithmic approach of deriving the pointillist ensemble of the digital polygon $\mathcal{P}$ corresponding to $\mathcal{C}$, driven by its factor of pointillism and keeping in view a natural reconstruction procedure that resembles the human psycho-visual aspect. To demonstrate the strength and effectiveness of the algorithm, we have shown the results on some typical data sets of various sources and with different shapes and complexities in Sec. 6.4. Finally, in Sec. 6.5, we summarize the work and mention its future possibilities.

6.2 Related Works

Over the last three decades, several algorithms have been proposed on appropriate sampling and efficient reconstruction of curves and surfaces, most of which are in two- and three-dimensional Euclidean (real) paradigm. The algorithms have been mainly developed to serve diverse applications requiring optimal/suboptimal sampling/representation of real/digital curves, subject to prescribed efficiency or reconstruction error. Hence, in this section, the algorithms on curve sampling and reconstruction, and the applications on sampled curve sets are discussed separately: the former in Sec. 6.2.1 and the latter in Sec. 6.2.2.

6.2.1 Works Related with Sampling and Reconstruction

In order to design various curve sampling and reconstruction algorithms with theoretical guarantees, a number of works have been proposed based on *digital-geometric* techniques and *geometric graphs*. In [Latecki and Rosenfeld (2002)], a method has been presented to recover an approximation of an unknown polygon from noisy digital data, which is obtained by digitizing either an image of the polygon or a sequence of points on its boundary. A digitization scheme based on scaling proposed by Brimkov (2009) scales an original continuous real object appropriately so that the resultant magnified object and its digitization have analogous geometric properties. It has been shown that it allows faithful reconstruction of the plane figures from the considered general class. The work presented by Brimkov and Klette (2008) defines digital manifolds of arbitrary dimension, provides the theoretical basis for curve or surface tracing in digital images. These theoretical foundations are fully based on the concept of adjacency relation. The study is useful towards generalization of digital curves in arbitrary dimension, skeletonization of digital objects, and determination of object boundary. The work by Lachaud *et al.* (2007) deals with a new tangent estimator to digitized curves based on digital line recognition. A tangent estimator is useful for estimating many geometric quantities and useful for many applications.

Geometric graphs, meant for connecting a set of points in a way to capture the underlying pattern, have been proposed since 1980's in several forms. In the domain of computational (Euclidean) geometry, one such elementary graph is the *nearest neighbor graph* (NNG) obtained by joining each point to its nearest neighbor(s) [Preparata and Shamos (1985)]. A similar geometric graph is the *relative neighborhood graph* (RNG) [O'Rourke (1982)], which is obtained by joining each pair of relative neighbors. Two points p and q are said to be relative neighbors in a set S if and only if there is no other point (in S) closer to both p and q than they are from each other. Interestingly, the RNG of a point set S is a subgraph of the *Delaunay triangulation* (DT), and hence, can also be computed from the DT of S [Jaromczyk and Kowaluk (1987)].

Another graph is the *sphere-of-influence graph* (SIG) [Avis and Horton (1985), Dwyer (1995), Toussaint (1988)] defined for a set of points/sites, which is constructed by identifying the nearest neighbor of each site, centering a ball at each site so that its nearest neighbor lies on the boundary, and joining two sites by an edge if and only if their balls intersect. Toussaint (1988) proposed this graph as a good primal sketch of a dot pattern, suitable for low-level vision tasks. The asymptotic behavior of the expected number of edges of an SIG is investigated by Dwyer (1995), considering that the sites are independent and uniformly distributed.

Kirkpatrick and Radke (1985) introduced the notion of *β-skeleton*, which can be obtained by joining pairs of points whose β-neighborhoods are empty. The neighborhood can be circle-based or lune-based (for $\beta \in [0, 1]$, the lune-based neighborhood is identical with the circle-based), and for the former, the β-skeleton can also be computed from the corresponding DT [Kirkpatrick and Radke (1985)]. Algorithms to construct the β-skeleton can be seen in [Mukhopadhyay and Rao (1998), Preparata and Shamos (1985)]. A continuous spectrum of β-skeletons can be obtained by varying the (real) parameter β. This can be used to extract the boundary of an object [Amenta *et al.* (1998)].

Veltkamp proposed another geometric graph, named as *γ-skeleton*, that captures the external as well as the internal shape of a point set [Veltkamp (1992, 1995)]. It is defined in terms of two parameters, namely $c_0 \in [-1, 1]$ and $c_1 \in [-1, 1]$, where $|c_0| \leq |c_1|$. The γ-

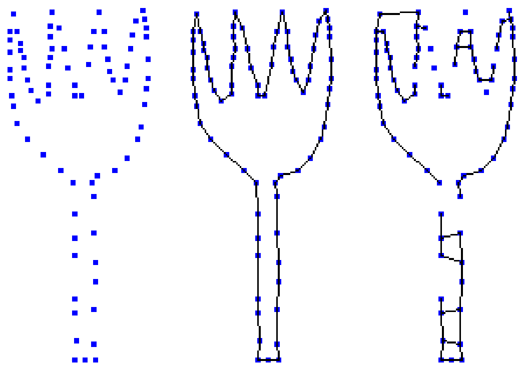


Figure 6.2: A sample/set of points (left) representing a "fork" [Althaus and Mehlhorn (2002), Althaus *et al.* (2000)]. The Traveling Salesman tour of the points produce the desired output (middle), whereas, the reconstruction algorithm of [Amenta *et al.* (1998)] deviates at some portions from the actual object.

neighborhood of a pair of points p and q is defined by two circles of radii $d(p,q)/2(1-|c_0|)$ and $d(p,q)/2(1-|c_0|)$ passing through p and q. Different possible domains of c_0 and c_1 result in two γ-neighborhoods of p and q, which define p and q as γ-neighbors if and only if at least one of their γ-neighborhoods is empty. This is used to obtain the γ-skeleton by joining all the γ-neighbors of the point set.

There also exist several other geometric graphs, such as *α-shapes* [Edelsbrunner *et al.* (1983)], *minimum spanning tree* [de Figueiredo and Gomes (1995)], *r-regular shapes* [Attali (1997)], *Gabriel graph* [Matula and Sokal (1984)], etc. The earlier methods [Attali (1997), Bernardini and Bajaj (1997), de Figueiredo and Gomes (1995), Kirkpatrick and Radke (1985)] consider the curve to be closed, smooth, and uniformly sampled. Surveys on these techniques appear in some of the contemporary works [Althaus and Mehlhorn (2002), Althaus *et al.* (2000), Dey (2007), Edelsbrunner (1998)]. Algorithms on reconstructing non-uniformly sampled open/closed smooth/non-smooth curves have been suggested in the later period [Althaus and Mehlhorn (2002), Amenta *et al.* (1998), Dey and Kumar (1999), Dey *et al.* (1999, 2000), Giesen (2000), Gold and Snoeyink (2001)].

6.2.2 Applications Based on Curve Sampling

Shape analysis and shape coding based on curve samples find a wide range of applications in today's digital world. Some of these in the context of our work are briefly discussed below to show the significance of sampling of digital curves.

6.2.2.1 Video coding

One state-of-the-art technology dealing with digital curves/shapes is object-oriented video coding, which has received a lot of attention in recent times, since it facilitates retrieval, interactive editing, and manipulation of videos [Wang *et al.* (2003)]. In line with the MPEG-4 standardization [Katsaggelos *et al.* (1998)], several contour-based shape coding methods have been proposed of late pertaining to a video sequence, represented through the evolution of video object planes. Some of these use vertex-based *polygonal approximations* for lossy shape coding [Gerken (1994), Hotter (1990), O'Connell (1997)]. In [Lee *et al.* (1999)], a *baseline shape coder* places the shape into a 2D coordinate system such that the projection of the shape onto the x-axis (baseline) is the longest. From the baseline, y-distances of points (sampled clockwise) on the shape boundary are measured. The boundary points at which the direction changes are called turning points. The boundary is represented by one-dimensional distance data with turning points, followed by *entropy coding*. In another work [Schuster and Katsaggelos (1997, 1998)], a framework for the *operational rate-distortion* (ORD) optimal encoding of shape information in the intra- and inter-modes was discussed. First order (polygons) and higher order (splines) approximation techniques are adopted to represent the boundary, and the control points of these curves are encoded to achieve the optimal result.

Apart from the above methods that operate directly on the boundary vertices, alternative approaches, which first analyze the shape information before processing, have also been proposed. The shape description algorithms can be classified into external and internal [Pavlidis (1978)]. The former ones deal with the description of the shape boundary, e.g., *Fourier descriptor* [Persoon and Fu (1977), Zahn and Roskies (1972)], *time series* [Kartikeyan and Sarkar (1989)], and *shape matrices* [Goshtasby (1985)]. The latter ones are mainly area-descriptor algorithms, e.g., *moment-based* approaches [Prokop and Reeves (1992)], skeletons or *medial axis transform* [Blum (1967), Maragos and Schafer (1986), Serra (1982)], and *shape decomposition* [Pitas and Venetsanopoulos (1990)].

To achieve a flexible tradeoff between the approximation error and the bit budget, the idea of decoupling the shape information into two independent signal data sets, namely the skeleton and distances of the boundary-sampled points from the skeleton, has been proposed [Wang *et al.* (2003)]. A given bit budget for a video frame decides the number and location of sample points for all skeletons and distance signals of all boundaries within a frame, so that the overall distortion is minimized. An ORD-optimal approach based on the *Lagrangian multiplier* and a *shortest path* algorithm in a directed acyclic graph are used for solving the problem.

6.2.2.2 Biometrics

In biometric authentication based on fingerprints and face images, there exist various works based on sampling of digital points. Broadly stated, a set of (higher-level) feature points, extracted meaningfully and judiciously from a set of (lower-level) digital curves, is used to find the approximate similarity with another set. For example, in fronto-parallel images, *chin contour* can be considered as a stable feature to preserve more details in face recognition, since from the physiological point of view, the part beneath the mouth of chin contour remains almost unaffected by a facial expression. The possible points of chin contour are obtained by using an algorithm based on prior distribution and local decision [Wang and Su (2003)]. Then the false points are removed by filtering and the other points are joined to get the chin contour by curve approximation. Classification by analyzing the features of contours are likely to improve the recognition rate and speed on huge face database. There also exist other approaches based on *snake model* involving *geometric active contour model* [Huang and Su (2002)], *active contour model* [Sun *et al.* (2002)], etc.

In fingerprint analysis, the salient features are ridges and minutiae (termination and bifurcation) [ANSI (1986)]. A ridge in a fingerprint is a digital curve, and a minutia is the point of termination of a ridge or the point from where a ridge bifurcates. There exist several applications related with fingerprint image analysis, which require both the minutiae and points sampled from ridge lines constituting a fingerprint topography [Bazen and Gerez (2003), Bhowmick and Bhattacharya (2008), Bhowmick *et al.* (2005a), Ceguerra and Koprinska (2002), Jain *et al.* (1997, 2001), Maltoni *et al.* (2003)]. Large legacy databases are in use today, which require huge space for storage and retrieval, and hence can be stored in a compact and efficient way using feature points and ridge points sampled in an appropriate manner.

6.2.2.3 Deformation analysis

Deformation or skewness of shape data arise in many practical applications such as medical image analysis. With a conventional statistical approach (e.g., Gaussian), the formulation of a shape model, therefore, cannot produce the desired result [Baloch and Krim (2007)]. Hence, several new techniques have come up in recent times to accommodate a departure of the shape data from a familiar distribution [Baloch and Krim (2007), Younes (1999), Zhu (1999)]. Region-based features such as *linelets* and *rib lengths*, whose histograms are computed and averaged to yield a Gibbs-like distribution for shape sampling, have been used in [Zhu (1999)]. A point correspondence-based object recognition, which requires additional transformations to compute shape distance metrics, has been proposed [Belongie *et al.* (2002)]. Sampling a shape by learning an *angle distribution*, with an aim to capture landmarks, present in a non-Gaussian shape data, has been proposed [Baloch and Krim (2007)]. This approach takes care of shape deformation caused by a patient's motion and improper alignment in medical imaging (e.g., X-ray or MRI). A similar model is *diffeomorphisms* of the unit circle via *conformal mappings* [Sharon and Mumford (2004)].

The model of *active contours*, or *snakes* [Kass *et al.* (1998)], has been used as a computational bridge between the low-level image data and the high-level shape information in the domain of cartoon face recognition [Hsu and Jain (2003)]. Human faces are represented semantically via facial components, such as eyes, mouth, face outline, and the hair outline, in the form of a *semantic face graph.* The facial components are encoded by closed (or open) snakes in parametric form, which interact among themselves to align the general facial topology onto the sensed face images. For mimicking facial expressions, the vertices of the mesh model are considered as sample points, which are hierarchically decomposed into three levels: 1) boundaries of facial components, 2) interiors of facial components, and 3) facial skin regions. Facial caricatures are generated based on an individual's facial distinctiveness from the average facial topology of the training set, and an exaggeration coefficient.

6.2.2.4 Shape-based segmentation

One of the most important image processing tools is segmentation, which has potential applications in medical visualization and diagnostics. Image noise, inhomogeneities, and lack of strong edges are some of the hardships that make this process a challenging one. Hence, in today's segmentation, shape knowledge and structural properties are used. Starting from an initial position inside the structure/region of interest, the model evolves to hit the boundaries of the object [Chesnaud *et al.* (1999), Munim and Farag (2007), Pardo *et al.* (2001), Zhu and Yuille (1996)]. In many cases, an energy formulation is given to describe the problem, requiring a minimization through the front evolution. Deformable models have been proposed to evolve a contour (front), minimizing an energy function that characterizes the evolution depending on the intrinsic properties of the contour, sampled appropriately [Kass *et al.* (1998)].

6.3 Proposed Work

6.3.1 Problem Formulation

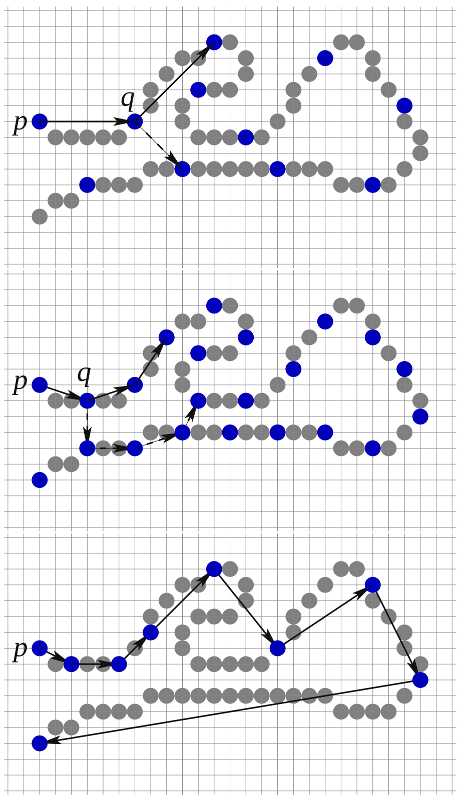
A digital curve $\mathcal{C}$ is represented either by a sequence of chain codes (Sec. 6.1) in a list or by a pattern of '1's (curve points) and '0's (background points) in a 2D array. It is easy to observe that some dropped/missing curve points ('1's) (which are replaced by '0's, thereof) in the 2D array might entirely disorder or upset the actual curve, $\mathcal{C}$. However, if we drop some curves points keeping the invariance of some underlying criterion, then the corresponding curve can be restored/reconstructed without any appreciable change w.r.t. the original one. Thus, selecting an optimal number of points to represent $\mathcal{C}$, is the concerned problem. Further, if we take into account the human/robot visionary mechanism, we should have the provision to increase the selected points so that the actual curve becomes clearer and easier to visualize. Thus, we have the following two perspectives to define the pointillist ensemble of a set of digital curves, C, that correspond to the digital object(s) embedded in a binary image:

Algorithmic Perspective: Given a set of planar digital curves, C, the (sub)optimal set of points, namely $\widehat{C}$, is called the *minimum pointillist ensemble* of C, provided the original set C can be reconstructed from $\widehat{C}$ using an appropriate algorithm.

Visual Perspective: The (*general*) *pointillist ensemble* $\widehat{C_\phi}$ of C should be specified by the *pointillist factor*, $\phi(\geqslant 1)$, such that a viewer can easily perceive/recognize the actual object from the ensemble $\widehat{C_\phi}$ (and also, C can be reconstructed from $\widehat{C_\phi}$ using an appropriate algorithm).

Evidently, $\phi = 1$ corresponds to the minimum pointillist ensemble of C. The small instances in Fig. 6.1 illustrate the above two perspectives of the problem (from the ensemble with $\phi = 2$, the actual object is more prominent than that with $\phi = 1$).

Difficulty of the Problem: Finding the pointillist ensemble directly from a (digital) curve $\mathcal{C}$ (or a set of curves) poses serious reconstruction problems. During the reconstruction (or visually guessing the actual curve), for a particular point p in the ensemble $\widehat{\mathcal{C}}$ corresponding to $\mathcal{C}$, if the point $q \in \widehat{\mathcal{C}}$ lies nearest to p, then q seems to lie nearer to p "along the curve" than any other point $q' \in \widehat{\mathcal{C}}$. That is, after traversing p, q will be traversed (along the reconstructed curve) before traversing any other point $q' \in \widehat{\mathcal{C}}$. However, since the occurrence of q in the original curve is unknown (from the ensemble), it might lead to a wrong traversal and a faulty output, thereof. Few examples illustrated in Fig. 6.3



(a) Uniform distribution of ensemble points (blue) over the curve (gray) is liable to disfigure the output after the reconstruction. From the ensemble point p, the nearest ensemble point q correctly aids the reconstruction; however, the next nearest point (dotted direction) incorrectly directs the reconstruction (direction in solid line), thereby producing a wrong output.

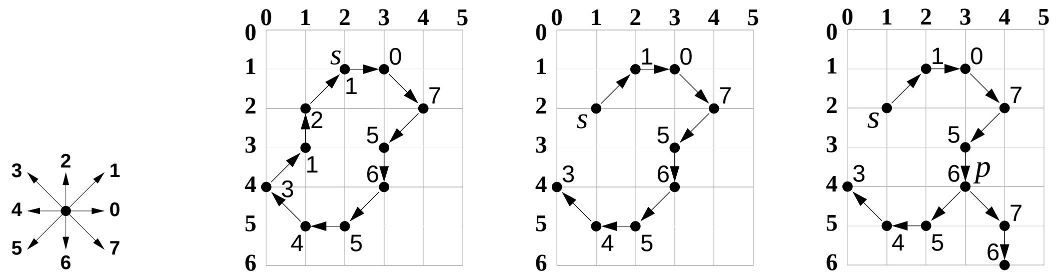
(b) Increasing the (uniform) distribution of ensemble points usually does not produce the required result (due to interference with an undesirable sample), as the curve may possess arbitrary flow pattern.

(c) Even if an algorithm produces an ensemble in which the points are sensibly selected from the curve points (i.e., during the reconstruction, the ensemble points are traversed in order of their occurrences along the actual curve), the resultant output (in the form of a digital polygon) may deviate quite alarmingly from the actual curve.

Figure 6.3: Some instances of pointillist ensembles of a digital curve, showing the difficulty of solving the problem.

explain the possible eventualities that are very likely to produce entirely wrong output (Fig. 6.3(a, b)) or a largely deviated output (Fig. 6.3(c)).

In order to maintain the original topological structure of the digital curve, therefore, we first decompose the set of digital curves into a set of shape-preserving digital polygons. The edges of these polygons are used, in turn, to induce the pseudo-vertices for constituting the pointillist ensemble that can be reconstructed to get back the original curve with a certain precision.



(a) Chain codes in 8-N.

(b) With $s := (2,1)$ as the start point, the chain code is $(2,1)\texttt{07}\ldots\texttt{21}$.

(c) With $s = (1,2)$ as the start point, the chain code is $(1,2)\texttt{10}\ldots\texttt{43}$.

(d) There are two open curves with chain codes $(1,2)\texttt{10}\ldots\texttt{43}$ and $(3,4)\texttt{76}$.

Figure 6.4: Chain codes for defining various digital curves.

6.3.2 Polygonal Decomposition of a Digital Object

Decomposition of a digital curve $\mathcal{C}$ into a digital polygon is performed by analyzing the chain code [Freeman (1961a,b)] of $\mathcal{C}$. In the 8-neighborhood (8N) connectivity[1], two grid points $(i,j) \in \mathcal{C}$ and $(i',j') \in \mathcal{C}$ are neighbors of each other, provided $\max(|i-i'|,|j-j'|) = 1$. Thus, the possible chain codes defining a digital curve connected in 8N are in $\{\texttt{0},\texttt{1},\texttt{2},\ldots,\texttt{7}\}$, as shown in Fig. 6.4(a). The chain code enumeration of a digital curve $\mathcal{C}$, therefore, describes an ordered sequence of grid points such that each point in $\mathcal{C}$ is a neighbor of its predecessor in the sequence. To be precise, if each point in $\mathcal{C}$ has exactly two neighbor points in $\mathcal{C}$, then $\mathcal{C}$ is said to be a *closed curve* (Fig. 6.4(b)). Otherwise, $\mathcal{C}$ has two points with one neighbor each, and the remaining points with two neighbors each, whence $\mathcal{C}$ becomes an *open curve* (Fig. 6.4(c)). Self-intersecting curves, of course, would have points with more than two neighbors; in such a curve, we split the curves into a set of open or/and closed curves only (Fig. 6.4(d)).

The polygonal decomposition adopted by us is based on extraction of approximate straight line segments from a digital curve using the recently published digital-geometric algorithm [Bhowmick and Bhattacharya (2007)]. Certain chain code properties of digital straightness [Klette and Rosenfeld (2004a), Rosenfeld and Klette (2001)] have been used in this algorithm with some relaxations to achieve the desired result. Shown in Fig. 6.5 are results on polygonal decomposition on a set of digital curves representing the contour of a real-world logo image, which show how the polygonal decomposition of a digital curve

[1]The definitions and discussions in this paper are with respect to 8-neighbor connectivity [Klette and Rosenfeld (2004a)] of the object, and are valid as well in 4-neighborhood with appropriate modifications.

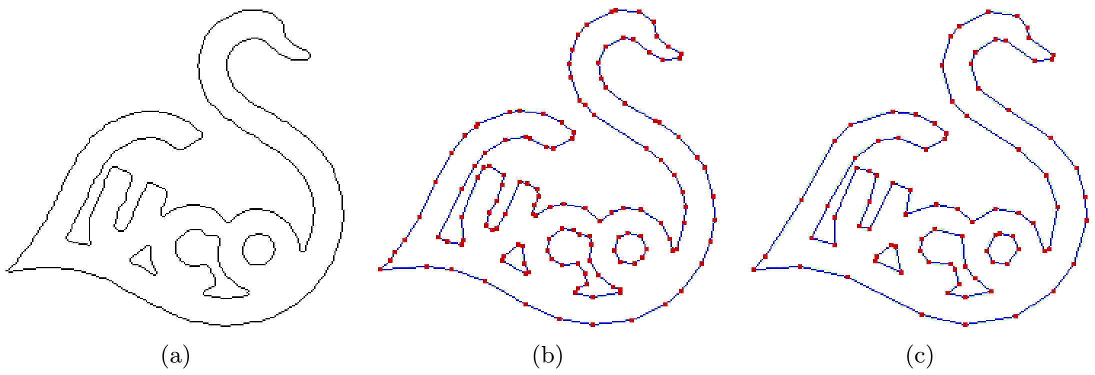


Figure 6.5: An example of polygonal decomposition of a digital curve set (polygon vertices in red): (a) original curve; (b) a tight polygonal decomposition; (c) a relatively slacker decomposition.

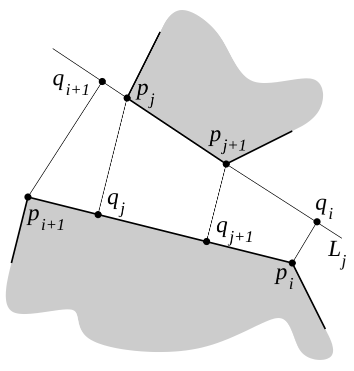
preserves its geometric orderliness. Although a slacked decomposition might digress from the original curve (Fig. 6.5(c)), a tighter decomposition almost traces the actual curve (Fig. 6.5(b)) with a higher complexity (i.e., a larger number of vertices) of the output polygon.

6.3.3 Pointillistic Object Representation

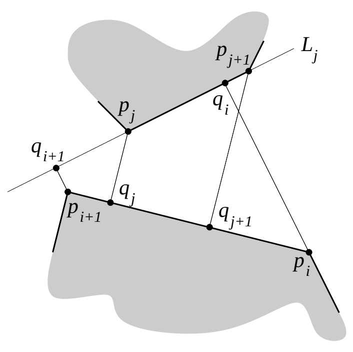
Let $C := \{\mathcal{C}^{(k)} : k = 1, 2, \ldots, K\}$ denote the set of K digital curves constituting a digital object. Let the digital polygon/poly-chain corresponding to the kth (closed/open) digital curve $\mathcal{C}^{(k)}$, be specified by the ordered set of vertices, namely $\mathcal{P}^{(k)} := \langle p_i^{(k)} : i = 1, 2, \ldots, n_k \rangle$, so that corresponding to the set C, we have the set of digital polygons, denoted by $P := \{\mathcal{P}^{(k)} : 1, 2, \ldots, K\}$.

An edge of the digital polygon $\mathcal{P}^{(k)}$ is, therefore, given by $e_i^{(k)} = (p_i^{(k)}, p_{i+1}^{(k)})$, where $i = 1, 2, \ldots, n_k - 1$, whether $\mathcal{C}^{(k)}$ is open or closed; and $e_{n_k}^{(k)} = (p_{n_k}^{(k)}, p_1^{(k)})$ in addition if $\mathcal{C}^{(k)}$ is closed. In order to obtain the pseudo-vertices as a pointillist ensemble of the digital polygons in P (and of C, thereof), we find the edge(s) (over all the polygons in P) having the minimum (Euclidean) distance from each edge of each polygon in P. To be precise, for each edge $e_i^{(k)}$ of each polygon $\mathcal{P}^{(k)} \in P$, we find the nearest edge point on each other edge of all the polygons in P (including $P^{(k)}$), as explained next.

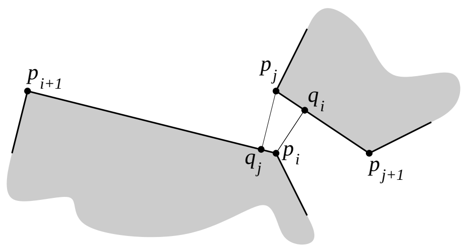
Let $e_i := (p_i, p_{i+1})$ and $e_j := (p_j, p_{j+1})$ be two edges of two (same or different) polygons (Fig. 6.6). Let the real straight lines containing the edges e_i and e_j be L_i and L_j, respectively. Let the projection of e_i on L_j be $e_i' := (q_i, q_{i+1})$, and that of e_j on L_i be



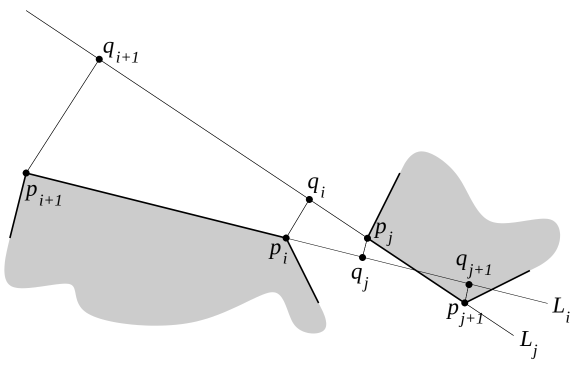
(a) $e'_j \subseteq e_i$ & $e_j \subseteq e'_i$.



(b) $e'_j \subseteq e_i$ & $e_j \cap e'_i \neq \emptyset$.



(c) $e'_j \cap e_i \neq \emptyset$ & $e_j \cap e'_i \neq \emptyset$.



(d) $e'_j \cap e_i = \emptyset$ & $e_j \cap e'_i = \emptyset$.

Figure 6.6: Different cases for finding the minimum distance between two edges of (same or different) polygon(s). The concerned edges are $e_i := (p_i, p_{i+1})$ and $e_j := (p_j, p_{j+1})$, and their respective projections on the lines L_j (containing e_j) and L_i (containing e_i) are e'_i and e'_j.

$e'_j := (q_j, q_{j+1})$, where q_i denotes the foot of the perpendicular from e_i to L_j, and so forth. Then, depending on the *containment relation* of e'_j in e_i and of e'_i in e_j, the nearest point of e_j from e_i is obtained, in accordance with the following possibilities:

Case 1: e'_j is contained in e_i, and e_j is contained in e'_i. Then, the minimum distance between e_i and e_j is given by $d_{ij} = \min\{p_j q_j, p_{j+1} q_{j+1}\}$ (Fig. 6.6(a)). [*Note:* e_i and e_j contain each other's projections, i.e., $e'_j = e_i$ and $e'_i = e_j$, if and only if e_i and e_j are equal in both length and direction.]

Case 2: e'_j is contained in e_i, and let, w.l.o.g., q_i is contained in e_j, whereas q_{i+1} lies outside e_j. Hence, $d_{ij} = \min\{p_j q_j, p_{j+1} q_{j+1}, p_i q_i\}$ (Fig. 6.6(b)).

Case 3: e'_j has partial overlap with e_i, and if, w.l.o.g., q_i and q_j are contained in e_j and

e_i respectively, then $d_{ij} = \min\{p_jq_j, p_iq_i\}$ (Fig. 6.6(c)).

Case 4: Neither e'_j has partial overlap with e_i, nor e'_i with e_j. This implies $d_{ij} = \min\{p_ip_j, p_ip_{j+1}, p_{i+1}p_j, p_{i+1}p_{j+1}\}$ (Fig. 6.6(c)).

In order to induce the pseudo-vertices in an edge $e_i^{(k)}$ of the polygon $\mathcal{P}^{(k)}$, we consider each edge in the set $E_i^{(k)} := \{e_j^{(k')} : e_j^{(k')} \in \mathcal{P}^{(k')} \wedge e_j^{(k')} \neq e_i^{(k)}\}_{k'=1}^{K}$. Using the relation (one out of the four relations on projective containment as mentioned above) of the edge $e_j^{(k')}$ with $e_i^{(k)}$, we compute the distance of $e_j^{(k')}$ from $e_i^{(k)}$. Based on the distances of all the edges of $E_i^{(k)}$, we find the edge, say, $e_j^{(k')}$, that has the minimum distance, say $d_{ij}^{(kk')}$, from $e_i^{(k)}$.

Using the nearest edge $e_j^{(k')}$ from the edge $e_i^{(k)}$ and the corresponding minimum distance ($d_{ij}^{(kk')}$), we induce the first pseudo-vertex, namely $p'^{(k)}_{i(+1)}$, on $e_i^{(k)}$, from the end of its start vertex ($p_i^{(k)}$). The pseudo-vertex $p'^{(k)}_{i(+1)}$ is induced on $e_i^{(k)}$ such that the distance of $p'^{(k)}_{i(+1)}$ from $p_i^{(k)}$ satisfies the following equation.

$$d(p'^{(k)}_{i(+1)}, p_i^{(k)}) = \max_{p \in e_i^{(k)}} \left\{ d(p, p_i^{(k)}) : \phi \cdot d_{ij}^{(kk')} > d(p, p_i^{(k)}) \right\} \tag{6.1}$$

where, $\phi(\geqslant 1)$ is called the *pointillist factor*. The minimum possible value of ϕ is unity, which ensures the generation of a suboptimal set of pseudo-vertices in order that the original polygon can be reconstructed from the pointillist ensemble (given by all polygon vertices in union with the pseudo-vertices). Increasing the value of ϕ induces a larger number of pseudo-vertices, thereby increasing the size of the ensemble and creating a better impression of the actual object.

Once the pseudo-vertex $p'^{(k)}_{i(+1)}$ is inserted on the edge $e_i^{(k)}$, the vertex $p_i^{(k)}$ and $p'^{(k)}_{i(+1)}$ become the nearest pair of vertices (whether of the original polygon or pseudo), which implies that the distance of no other vertex or point on any other edge of $E_i^{(k)}$ from $p_i^{(k)}$ or $p'^{(k)}_{i(+1)}$ is less than the distance between $p_i^{(k)}$ and $p'^{(k)}_{i(+1)}$. Thus, during reconstruction from the pointillist ensemble, $p_i^{(k)}$ and $p'^{(k)}_{i(+1)}$ would be nearest to each other, and so one will be always visited after the other, thereby preserving their order and recreating the edge $e_i^{(k)}$ as required.

Insertion of $p_{i(+1)}^{(k)}$ may change the status of the nearest edge (in $E_i^{(k)}$) of the part of $e_i^{(k)}$ from $p'^{(k)}_{i(+1)}$ to $p_{i+1}^{(k)}$, denoted by the *sub-edge* $e_{i(+1)}^{(k)} := (p'^{(k)}_{i(+1)}, p_{i+1}^{(k)})$. Hence, we recompute the distances of the edges in $E_i^{(k)}$ from $e_{i(+1)}^{(k)}$ and depending on the four possible cases, we find the nearest edge of $e_{i(+1)}^{(k)}$. Based on the distance (say $d_{i(+1)}$) of the nearest edge of

$e^{(k)}_{i(+1)}$ from $e^{(k)}_{i(+1)}$, and the specified value of ϕ, the next pseudo-vertex $p^{(k)}_{i(+2)}$ is inserted in $e^{(k)}_{i(+1)}$ from the end of $p^{(k)}_{i(+1)}$, so that the distance $d(p^{(k)}_{i(+1)}, p^{(k)}_{i(+2)})$ does not exceed $\phi \cdot d_{i(+1)}$ (as in the case of inducing the first pseudo-vertex: Eqn. 6.1).

The above process is continued until the distance of the last induced vertex on $e^{(k)}_i$, namely $p^{(k)}_{i(+m)}$, from $p^{(k)}_{i+1}$ is less than ϕ times the distance of the nearest edge from the corresponding sub-edge, $e^{(k)}_{i(+m)} := (p^{(k)}_{i(+m)}, p^{(k)}_{i+1})$. For each edge of all the K polygons, this is repeated to derive the complete pointillist ensemble corresponding to the given object and the specified value of the pointillist factor.

A Note on Minimality: The process of inducing pseudo-vertices starts from the first edge $e^{(1)}_1$ of the first polygon $\mathcal{P}^{(1)}$ in P, and covers all the edges of all the K polygons in P in succession. The first pseudo-vertex $p'^{(1)}_{1(+1)}$ inserted on $e^{(1)}_1$ — in accordance with Eqn. 6.1 — is obviously at the maximum possible distance from the start vertex $p^{(1)}_1$. For, if we consider $p'^{(1)}_{1(+1)}$ to be the next possible point on $e^{(1)}_1$, then the distance $d(p'^{(1)}_{1(+1)}, p^{(1)}_1)$ exceeds the distance of the nearest vertex from $p^{(1)}_1$, whereby $p'^{(1)}_{1(+1)}$ fails to be the nearest point of $p^{(1)}_1$ and giving rise to wrong results during reconstruction, there of. Hence, the iterative process of inserting pseudo-vertices produces an ensemble of minimal size for $\phi = 1$ subject to the choice of start vertex. With a different start vertex, the resultant ensemble may, however, reduce in size, but finding a start vertex for which the ensemble is of minimum size irrespective of the underlying digital curve is really a very hard and complex problem, which may be investigated in the future for a theoretical solution.

6.3.4 Curve Reconstruction from its Ensemble

As the first step of reconstructing the digital curve $\mathcal{C}$ from its ensemble $\widehat{\mathcal{C}}$, the points constituting $\widehat{\mathcal{C}}$ are lexicographically sorted with their x-coordinates (primary key) and y-coordinates (secondary key) in a 2-dimensional link list. If the ensemble $\widehat{\mathcal{C}}$ consists of m points, then the sorted link list is given by

$$\widehat{\mathcal{C}}_{xy} = \left\langle \widehat{p}_i := (\widehat{x}_i, \widehat{y}_i) \in \mathbb{Z}^2 \mid (\widehat{x}_i < \widehat{x}_{i+1}) \text{ or } (\widehat{x}_i = \widehat{x}_{i+1} \text{ and } \widehat{y}_i < \widehat{y}_{i+1}) \right\rangle_{i=1}^{m}. \quad (6.2)$$

Next, for the reconstruction, we use two important properties related with the *Delaunay triangulation* $DT(S)$ of *any* point set S [Berg *et al.* (2000)], which are as follows:

P1. The closest pair of points in S are neighbors in $DT(S)$.

P2. For the *Euclidean graph*[1] $EG(S)$ of S, the *minimum spanning tree* $MST(EG(S))$ is a subgraph of $DT(S)$.

Hence, we compute the Delaunay triangulation $DT(\widehat{\mathcal{C}})$ of $\widehat{\mathcal{C}}$ by applying Fortune's algorithm [Berg *et al.* (2000)] on $\widehat{\mathcal{C}}_{xy}$, since there is an edge from each point $\widehat{p}_i \in \widehat{\mathcal{C}}$ to its nearest point in $DT(\widehat{\mathcal{C}})$ [Property P1]. However, there are other edges also from $\widehat{p}_i$ to other (second nearest, third nearest, and so forth) points in $DT(\widehat{\mathcal{C}})$, which depends on the *Voronoi diagram* $Vor(\widehat{\mathcal{C}})$ of $\widehat{\mathcal{C}}$. In order to find the nearest point of $\widehat{p}_i$ from $DT(\widehat{\mathcal{C}})$, we apply Kruskal's algorithm of minimum spanning tree [Cormen *et al.* (2000)] on $DT(\widehat{\mathcal{C}})$ to obtain $MST(DT(\widehat{\mathcal{C}}))$ [Property P2]. The reconstructed digital curve is, therefore, given by joining the ordered sequence of vertices of $MST(DT(\widehat{\mathcal{C}}))$.

Reconstruction Time: To compute the Delaunay triangulation of m points constituting the ensemble $\widehat{\mathcal{C}}$, we need $T_{DT} = O(m \log m)$ time [Berg *et al.* (2000)]. Now, the number of edges in $Vor(\widehat{\mathcal{C}})$ being at most $3m - 6 = O(m)$ [Berg *et al.* (2000)] and the Delaunay triangulation being a dual structure of the corresponding Voronoi diagram, the number of edges in $DT(\widehat{\mathcal{C}})$ is $O(m)$. Kruskal's algorithm takes $O(e \log v)$ time for a weighted graph with v vertices and e edges. Hence, the time-complexity for construction of $MST(DT(\widehat{\mathcal{C}}))$ from $DT(\widehat{\mathcal{C}})$ is $T_{MST} = O(m \log m)$, wherefore the total time-complexity of reconstructing the digital curve from its ensemble is given by $T_{DT} + T_{MST} = O(m \log m)$.

6.3.5 Reconstruction Error

During reconstruction, the nearest point-pairs of the ensemble $\widehat{\mathcal{C}}$ are connected by digital straight line segments (DSS) [Klette and Rosenfeld (2004a)]. As a result, there may occur erroneous points. An erroneous point or error point is a digital point $p \in \mathbb{Z}^2$ in a reconstructed DSS for which the nearest point of the original curve $\mathcal{C}$ does not coincide with p.

Hence, a measure of error incurred during reconstruction from the pointillist ensemble, is given by how much a particular point $(x, y) \in \mathcal{C}_k$ has deviated in the corresponding reconstructed polygon, namely $\widehat{\mathcal{P}}_k$. If $\widetilde{p} := (\widetilde{x}, \widetilde{y})$ be the point in $\widehat{\mathcal{P}}_k$ corresponding to $p := (x, y)$ in $\mathcal{C}_k$, then for all points in the curve set $C := \{\mathcal{C}_k : k = 1, 2, \ldots, K\}$, this measure is captured by the variation of the number of error points with their deviations,

[1]If S consists of m points, then the vertices of $EG(S)$ are the points in S and the edges are all $\binom{n}{2}$ undirected pairs of distinct points, the weight of each edge being given by the Euclidean distance between the corresponding points.

where the deviation from p to $\widetilde{p}$ can be measured by

$$dev_\infty(p \to \widetilde{p}) = \min\{|x - \widetilde{x}|, |y - \widetilde{y}|\}. \tag{6.3}$$

Clearly, $dev_\infty(p \to \widetilde{p})$ depends on how tight was the polygonal decomposition of the curves in C (Sec. 6.3.2). Hence, the fraction of the number of points in C with a given deviation d_∞ varies with the precision of the polygonal decomposition. If we adopt the tightest decomposition where no point of a decomposed polygon deviates at all from the corresponding curve point, then the polygons consist of a large number of vertices, thereby reporting a large pointillist ensemble. On the contrary, a slackened decomposition produces polygons with lesser number of vertices, and reducing the ensemble in size, which, of course, introduces appreciable reconstruction errors. A trade-off with the polygonal decomposition is, therefore, necessary to achieve the desired results on pointillist ensemble of C and the resultant reconstruction. A study on the nature of the error distribution for a sufficiently large population of arbitrary digital curves may be, therefore, a promising area of theoretical analysis on pointillist representation of digital curves.

6.3.6 Sampling Quality

The quality of sampling/approximation is quantified, in general, by the amount of discrepancy between the reconstructed set of curves and the original set. There are several measures to assess the approximation of a curve $\mathcal{C}_k$, such as

(i) compression ratio CR $= N_k/M_k$, where N_k is the number of points in $\mathcal{C}_k$ and M_k is the number of vertices in the approximate polygon $\mathcal{P}_k$;
(ii) the integral square error (ISE) between $\mathcal{C}_k$ and $\mathcal{P}_k$.

Since there is always a trade-off between CR and ISE, other measures may also be used [Held *et al.* (1994), Rosin and West (1995), Sarkar (1993)]. These measures, however, may not always be suitable for some intricate approximation criterion. For example, the figure of merit [Sarkar (1993)], given by FOM = CR/ISE, may not be suitable for comparing approximations for some common cases, as shown by [Rosin (1997)]. In a work [Ventura and Chen (1992)], the percentage relative difference, given by $((E_{approx} - E_{opt})/E_{opt}) \times 100$, has been used, where E_{approx} is the error incurred by a suboptimal algorithm under consideration, and E_{opt} the error incurred by the optimal algorithm, under the assumption that same number of vertices are produced by both the algorithms. Similarly, one may use two components, namely fidelity and efficiency, given by $(E_{opt}/E_{approx}) \times 100$ and $(M_{opt}/M_{approx}) \times 100$ respectively, where M_{approx} is the number of vertices in the approximating polygon produced by the suboptimal algorithm and M_{opt} is the same produced by the optimal algorithm subject to same E_{approx} as the suboptimal one [Rosin (1997)].

The algorithm proposed here is not constrained by the number of vertices M_k of the output polygon $\mathcal{P}_k$, and therefore, the measures of approximation where M_k acts as an invariant, are not applicable. Instead, we have considered the error of approximation, namely τ, as the sole parameter in our algorithm, depending on which the number of vertices M_k corresponding to $\mathcal{P}_k$ will change. A high value of τ indicates a loose or slacked approximation, whence the number of vertices M_k decreases automatically, whereas a low value of τ implies a tight approximation, thereby increasing the number of vertices in the approximate polygon. Hence, in accordance to the usage of τ in both of our proposed methods, one based on criterion $\mathrm{C}_{\sum}$ and the other on $\mathrm{C}_{\max}$, the total number of vertices $M := M_1 + M_2 + \ldots + M_K$ in set of approximate polygons $\{\mathcal{P}_k\}_{k=1}^{K}$ corresponding to the input set of DC, namely $\mathcal{I} := \{\mathcal{C}_k\}_{k=1}^{K}$, versus τ, provides the necessary quality of approximation. Since the total number of points lying on all the points in $\mathcal{I}$ characterizes (to some extent) the complexity of $\mathcal{I}$, we consider the compression ratio (CR) as a possible measure of approximation.

Another measure of approximation is given by how much a particular point $(x, y) \in \mathcal{C}_k \in \mathcal{I}$ has deviated in the corresponding polygon $\mathcal{P}_k$. If $\widetilde{p} := (\widetilde{x}, \widetilde{y})$ be the point in $\mathcal{P}_k$ corresponding to $p := (x, y)$ in $\mathcal{I}$, then for all points in $\mathcal{I}$, this measure is captured by the variation of the number of points with isothetic deviation $d_\perp$ w.r.t. $d_\perp$, where the *(isothetic) deviation from p to $\widetilde{p}$* is given by

$$dev_\perp(p \to \widetilde{p}) = \min\{|x - \widetilde{x}|, |y - \widetilde{y}|\}. \tag{6.4}$$

Further, since $dev_\perp(p \to \widetilde{p})$ depends on the chosen value of τ in our algorithm, the fraction of the number of points in $\mathcal{I}$ with deviation $d_\perp$ varies plausibly with τ. So, the *isothetic error frequency* (IEF) (or, simply *error frequency*), given by

$$f(\tau, d_\perp) = \frac{1}{N} \left|\{p \in \mathcal{I} : dev_\perp(p \to \widetilde{p}) = d_\perp\}\right|, \tag{6.5}$$

versus τ and $d_\perp$, acts as the second measure that provides the error distribution for the polygonal approximation of $\mathcal{I}$.

It may be observed that, the error frequency distribution in Eqn. 6.5 is equivalent to the probability density function, and satisfies the criterion

$$\sum_{d_\perp} f(\tau, d_\perp) = 1, \text{ for } \tau = 0, 1, 2, \ldots.$$

In fact, the error frequency distribution function in our measure is a bivariate distribution of (finite-size) samples of size N, depending on the two variables, namely τ and $d_\perp$. A study on the nature of the error frequency distribution for a sufficiently large population of arbitrary digital curves may be, therefore, a promising area of theoretical analysis of polygonal approximation of digital curves.

6.4 Experimental Results

We have implemented the algorithm in C (in SunOS Release 5.7 Generic of Sun_Ultra 5_10, Sparc, 233 MHz). The algorithm takes a set of digital curves, $C := \{\mathcal{C}_k : k = 1, 2, \ldots, K\}$ (Sec. 6.3.2), as input, and using the specified value of the pointillist factor ϕ, derives the optimal/sub-optimal pointillist ensemble (i.e., the unordered set of sample points: Sec. 6.3.3). In order to verify the quality and efficiency (Sec. 6.3.6) of the algorithm, we have also implemented the procedure to reconstruct a curve set from the corresponding ensemble (Sec. 6.3.4).

We have tested the algorithm on several classified image databases of diverse nature and of varied interests pertaining to pattern recognition and computer vision, which are as follows:

(i) Natural image database: 120 images [Martin *et al.* (2001)];

(ii) Logo image database: 1034 images (received on request, from Prof. Anil K. Jain and Aditya Vailya of Michigan State Univ., USA);

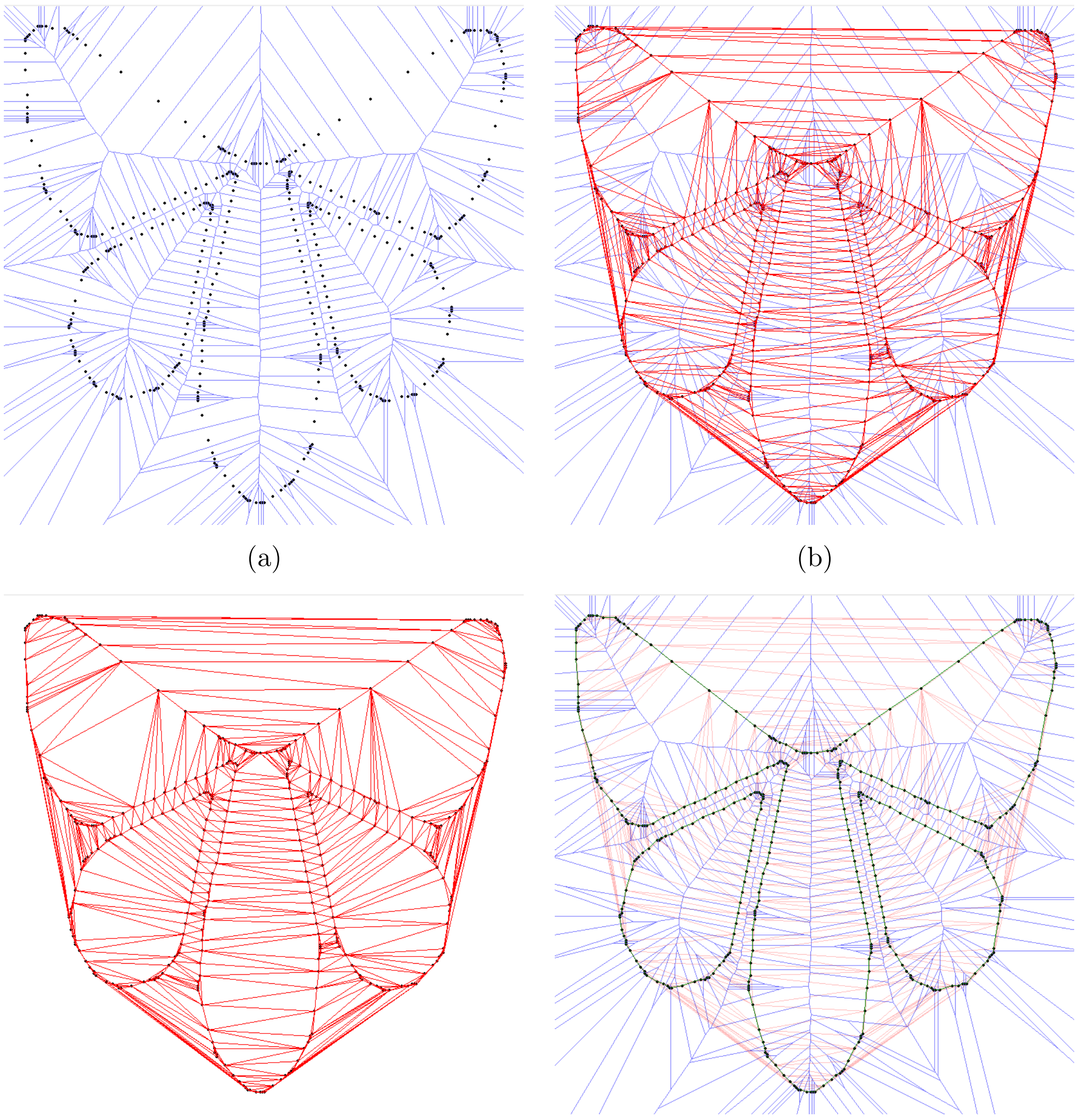
(iii) Selected database of optical and handwritten characters: 550 images [Hull (1994), ISI (2002)];

(iv) Fingerprint databases FVC2000 db1 and db2: 160 images [FVC 2000 (2000)];

(v) Sports video sequences on "pitcher-1" and "pitcher-2": 320 frames;

(vi) Test curve database: 100 irreducible, random, and closed digital curves.

Sets (i), (ii), (iii), and (v) are 8-bit gray-scale images, which have been first passed through an edge-detection stage, followed by thinning [Gonzalez and Woods (1993)]. The procedure of reconstructing a curve from its ensemble has been demonstrated in Fig. 6.7 on a small logo image. Shown in Fig. 6.8 are results on "drummers" (Set (i)) when the pointillist ensemble is minimally defined (i.e., with the pointillist factor $\phi = 1$). The input binary image is first processed to get its polygonal decomposition, from which its pointillist



(a) (b) (c) (d)

Figure 6.7: Step-by-step demonstration of the reconstruction from the pointillist ensemble $\widehat{\mathcal{C}}$ (original curve “logo_411” shown in Fig. 6.11). (a) Voronoi diagram, $Vor(\widehat{\mathcal{C}})$. (b) Delaunay triangulation (in red), $DT(\widehat{\mathcal{C}})$, as obtained from $Vor(\widehat{\mathcal{C}})$. (c) $DT(\widehat{\mathcal{C}})$ as a subgraph of the Euclidean graph $EG(\widehat{\mathcal{C}})$. (d) The reconstructed curve (in green) given by $MST(DT(\widehat{\mathcal{C}}))$. Note that coordinates of the points in the ensemble have been scaled here by a factor of 3 for a better visibility of the underlying geometric data structures.

Table 6.1: Results of the proposed algorithm ($\phi = 1$) on different images

Image name	Image size	$\vert\mathcal{C}\vert$	$\vert\mathcal{P}\vert$	$\vert\widehat{\mathcal{P}}\vert$	Point-illism time	Recons-truction time
"drummers"	340×244	4084	207	796	0.547	0.720
"logo355"	256×256	3513	193	1598	0.482	1.285
"logo416"	256×256	2054	127	628	0.436	0.539
"logo417"	256×256	3138	273	982	0.843	1.100
"test001"	256×256	1624	142	425	0.552	0.726

Time to form the ensemble and reconstruction time shown in seconds
$|\mathcal{C}|$ = number of curve points in the image
$|\mathcal{P}|$ = number of vertices after polygonal decomposition
$|\widehat{\mathcal{P}}|$ = number of points in the pointillist ensemble

ensemble is derived. To demonstrate the efficiency and robustness of the method, we have also shown a part of the reconstruction and the original curve set superimposed on the ensemble. Denser ensembles with $\phi = 2$ and $\phi = 4$, shown in Fig. 6.9, demonstrate the strength of the method in capturing the actual shape in a manner that imitates our psycho-visual sensory reflex.

The ensembles corresponding to the digital contours of some logo images (Set (ii)) for $\phi = 1$ and $\phi = 2$ are shown in Fig. 6.10 and Fig. 6.11. In Table 6.1, we have furnished some other significant parameters for a few images to show the results of the algorithm in finding the pointillist ensembles of the corresponding set of curves.

As mentioned in Sec. 6.2.2.2, points sampled from fingerprint ridge lines constituting a fingerprint topography are used in various biometric applications related with fingerprint image analysis [Bazen and Gerez (2003), Bhowmick and Bhattacharya (2008), Bhowmick *et al.* (2005a), Ceguerra and Koprinska (2002), Jain *et al.* (1997, 2001), Maltoni *et al.* (2003)]. Hence, to demonstrate the results of the algorithm on fingerprint images, we have shown the pointillist ensembles of a fingerprint image (Set (iii)) corresponding to a few values of ϕ in Fig. 6.12[1]. Similar results on three other fingerprint images — selected in a way so that they possess appreciable differences among themselves apropos their ridge topographies — for $\phi = 1$, 2, and 8, are shown in Fig. 6.13. It may be noticed from

[1]The procedures of obtaining the thinned ridge topography (input to the proposed algorithm) from a gray-scale fingerprint image have been discussed in [Bhowmick *et al.* (2005a)].

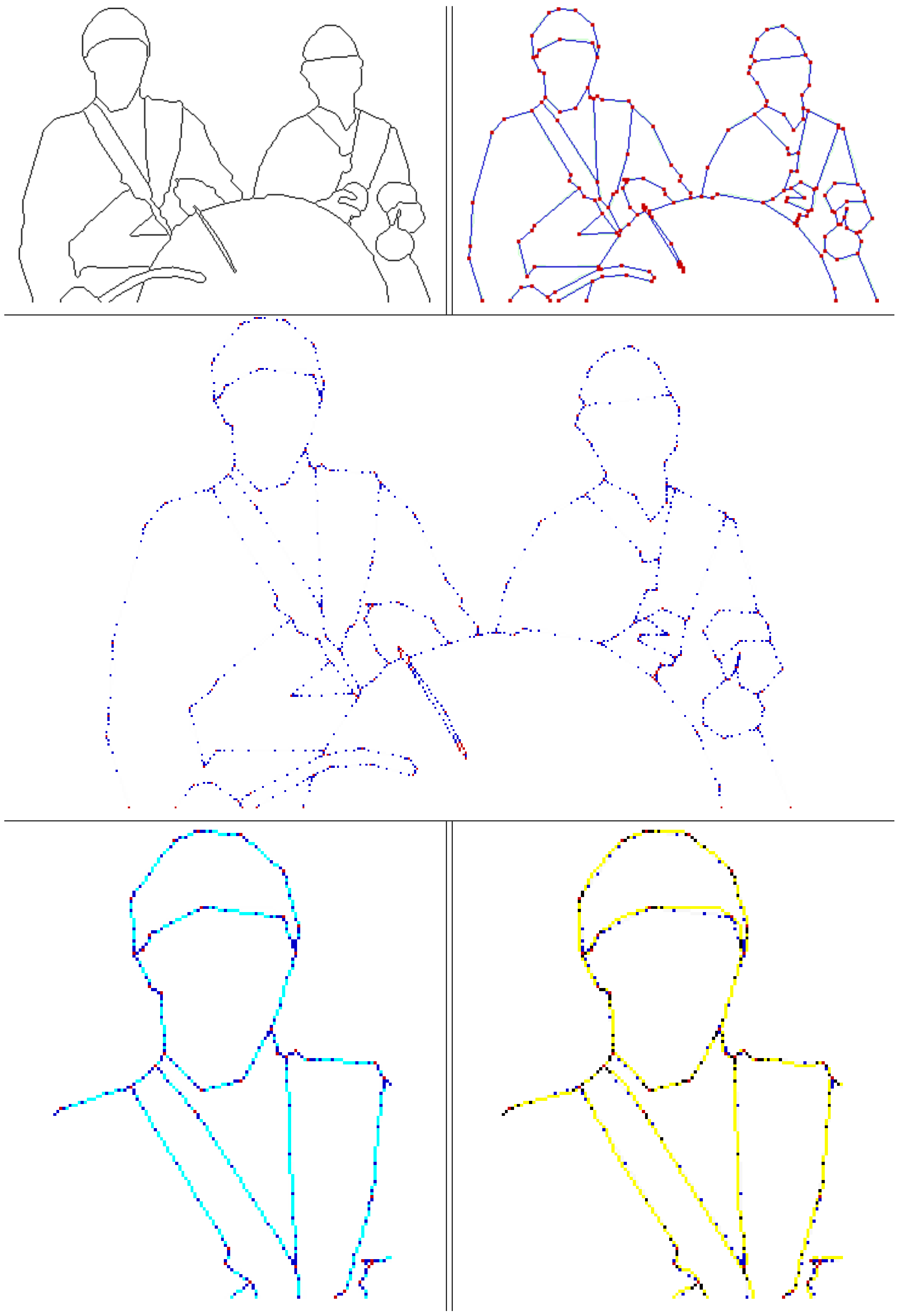


Figure 6.8: Results on “drummers” for minimum pointillism ($\phi = 1$): top left shows the input set; top right shows the polygonal description; middle row shows the pointillist ensemble; bottom left shows a part of the reconstructed set from the pointillist ensemble; bottom right shows the original object superimposed on the ensemble.

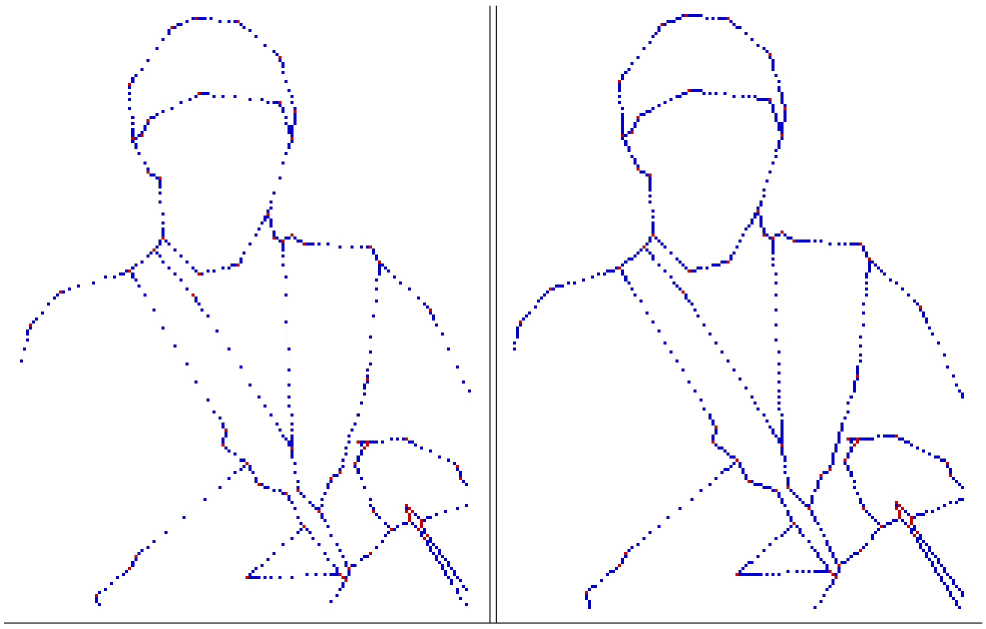


Figure 6.9: Pointillist ensembles of “drummers” for higher pointillist factor: left with $\phi = 2$; right with $\phi = 4$.

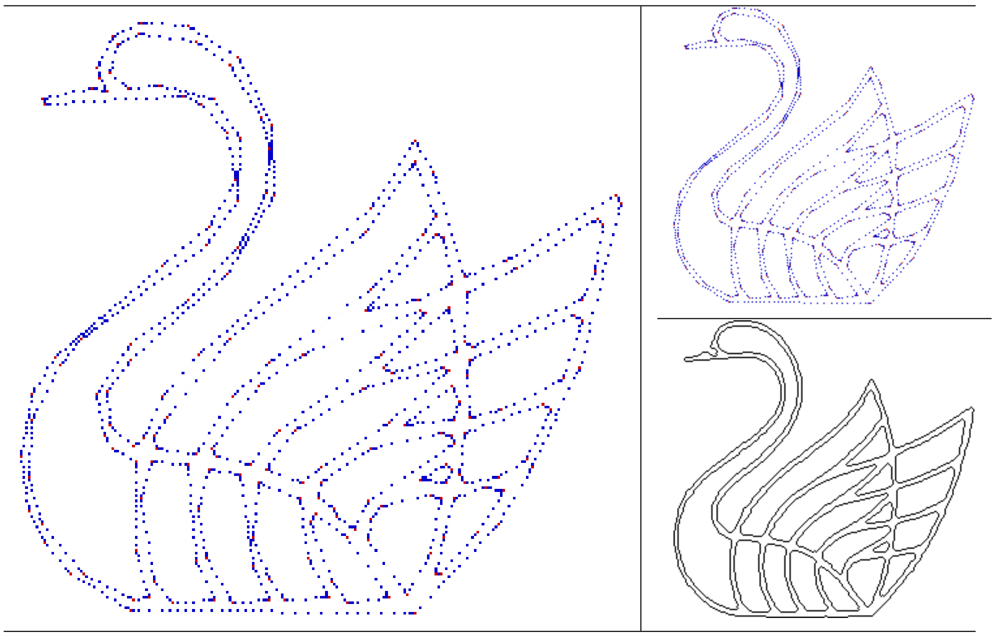


Figure 6.10: Results on “logo355”: left image shows the ensemble for $\phi = 1$; right top is for $\phi = 2$; right bottom shows the original object.

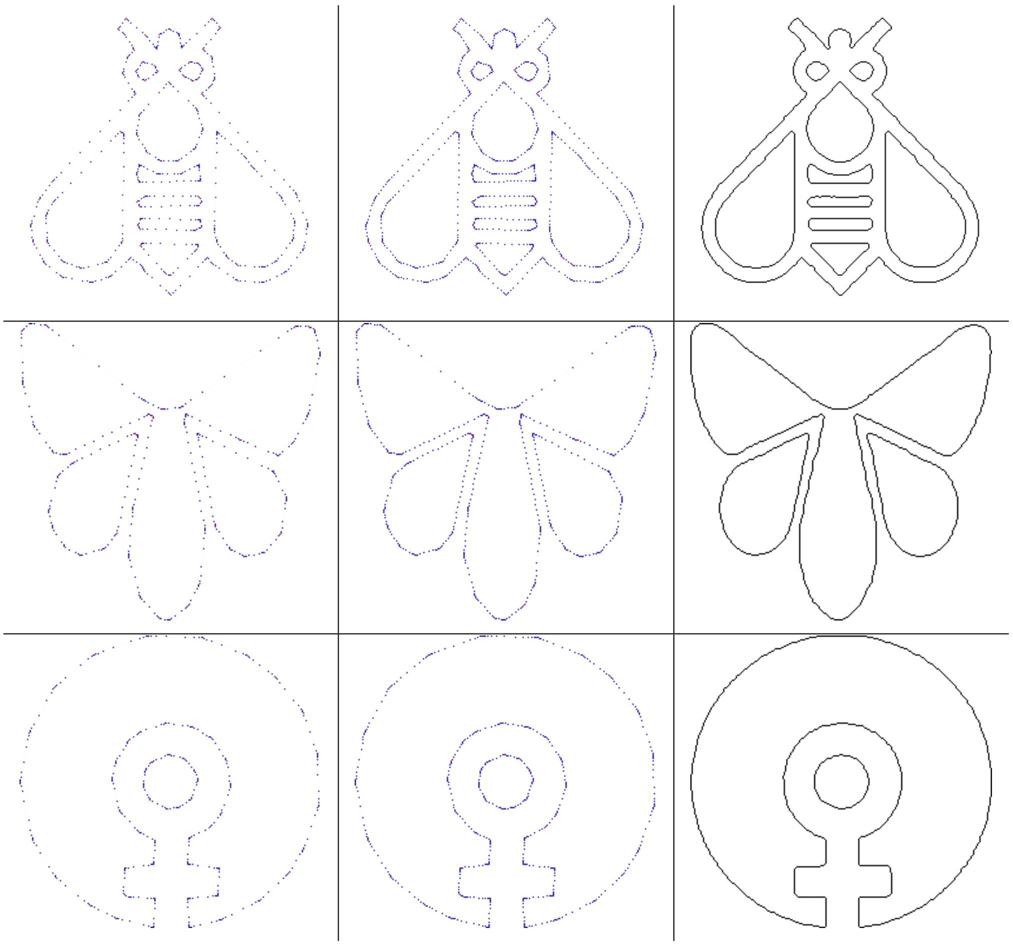


Figure 6.11: Results on some other polygons corresponding to logo images: left image shows the ensemble for $\phi = 1$; middle one is for $\phi = 2$; right one shows the original object.

their pointillist ensembles that density of points is high in and around a region where the ridges possess bifurcations or crossovers. Further, as the pointillist factor ϕ is gradually increased, the pointillist ensemble tends to the original/input ridge set, which justifies the convergence and robustness of the algorithm.

As another potential application where the algorithm may be used, we have also presented here a portion of the results on a small video sequence in Figs. 6.14 and Fig. 6.15. Since vertex- or point-based representation is usually followed for shape coding in the video object plane (Sec. 6.2.2.1), a (optimal or suboptimal) pointillist ensemble may be used in an appropriate way. Such an ensemble, without any ordering (unlike the ordered set of vertices of a polygon), captures the shape with a nearly-lossless reconstructive quality.

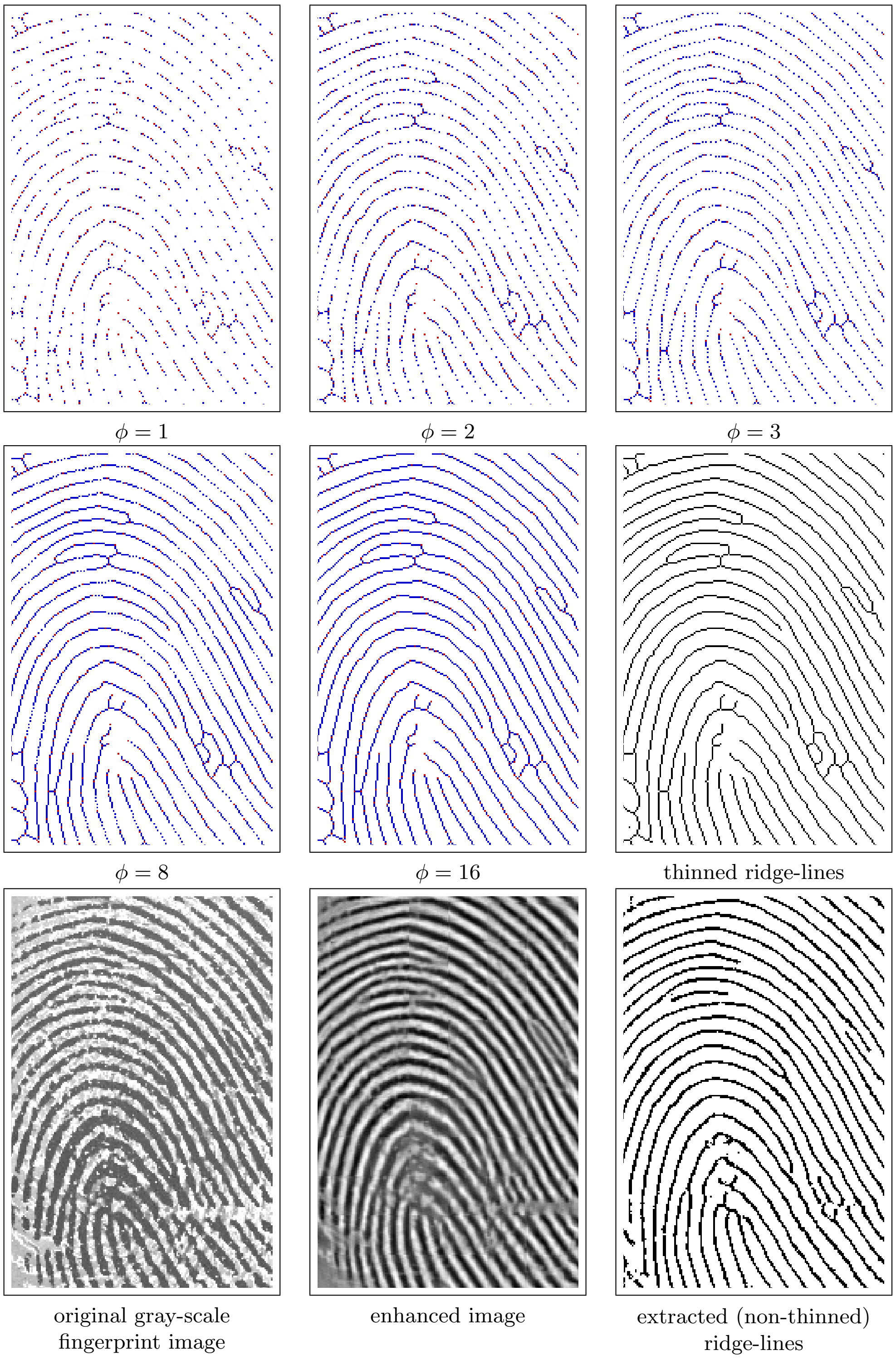


Figure 6.12: Results of the proposed algorithm on a fingerprint image.

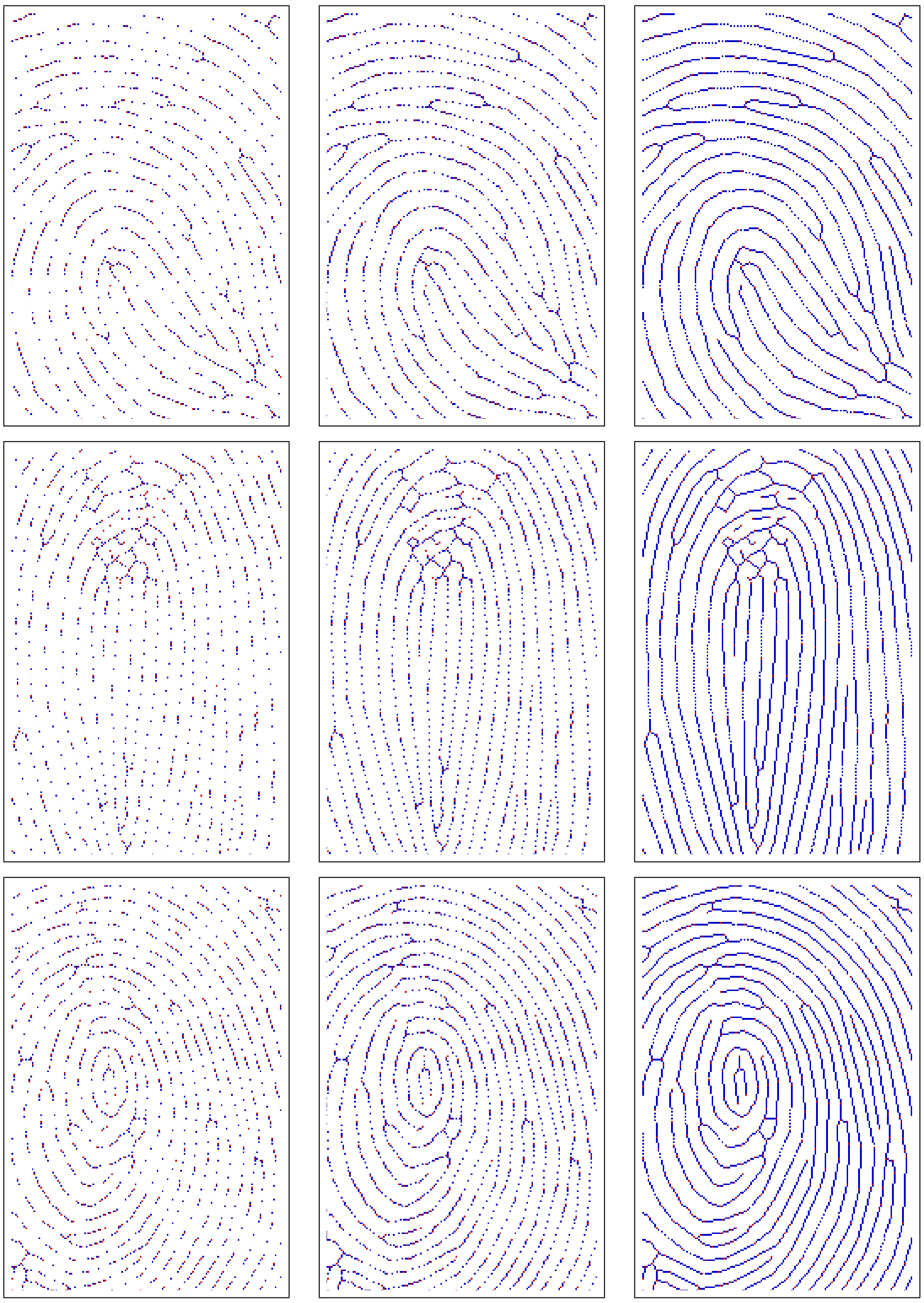


Figure 6.13: Results on three other fingerprint images for $\phi = 1$ (left column), $\phi = 2$ (middle column) and $\phi = 8$ (right column).

The pointillist ensembles for $\phi = 1$ (Fig. 6.14) corresponding to 16 frames, which represent a part of the "pitcher" video sequence, depict how the ensembles gradually change over these frames, though without suffering any appreciable loss (when subject to reconstruction). Visually, the underlying objects are, of course, more pronounced when the pointillist factor ϕ is increased, say, to $\phi = 2$ (Fig. 6.15), which is a suboptimal solution.

6.5 Conclusion

We have shown how a set of digital curves, after being decomposed into a set of shape-preserving digital polygons, can be represented by an appropriate pointillist ensemble. The size of the ensemble can be controlled by the pointillist factor, ϕ, which, when set to unity, produces a minimally defined ensemble, and when gradually increased, produces larger ensembles depicting clearer impression about the underlying object set. The proposed algorithm used to find the ensemble imitates the human visual mechanism, and the subsequent reconstruction using the nearest neighbor rule reproduces a digital curve that almost preserves its original shape, provided the polygonal decomposition of the original curve is reasonably good. Thus, the pointillist ensemble is likely to serve a useful and efficient representation of an object for an application dealing with computer or robot vision system.

The concept of pointillism ensures a freedom from ordering of the points in the resultant ensemble. The school of art on pointillism [Gage (1987), Seurat (1966)] initiated the Neo-impressionist style of painting by point-dabbing in the 19th century, an idea, which is deployed here to obtain an elegant representation of a digital object in the form of a reduced order-free point set. As mentioned in Sec. 6.3.3, a challenging problem is to find the minimum pointillist ensemble corresponding to any digital curve for $\phi = 1$ so that a reconstruction of the ensemble based on the nearest-neighbor rule is able to recreate the original curve without fail. Further possibilities involving pointillist ensembles lie in (i) shape analysis of digital objects using their pointillist ensembles; (ii) color image representation and subsequent analysis using ensembles in the color spectrum; and (iii) other image processing and computer vision applications.

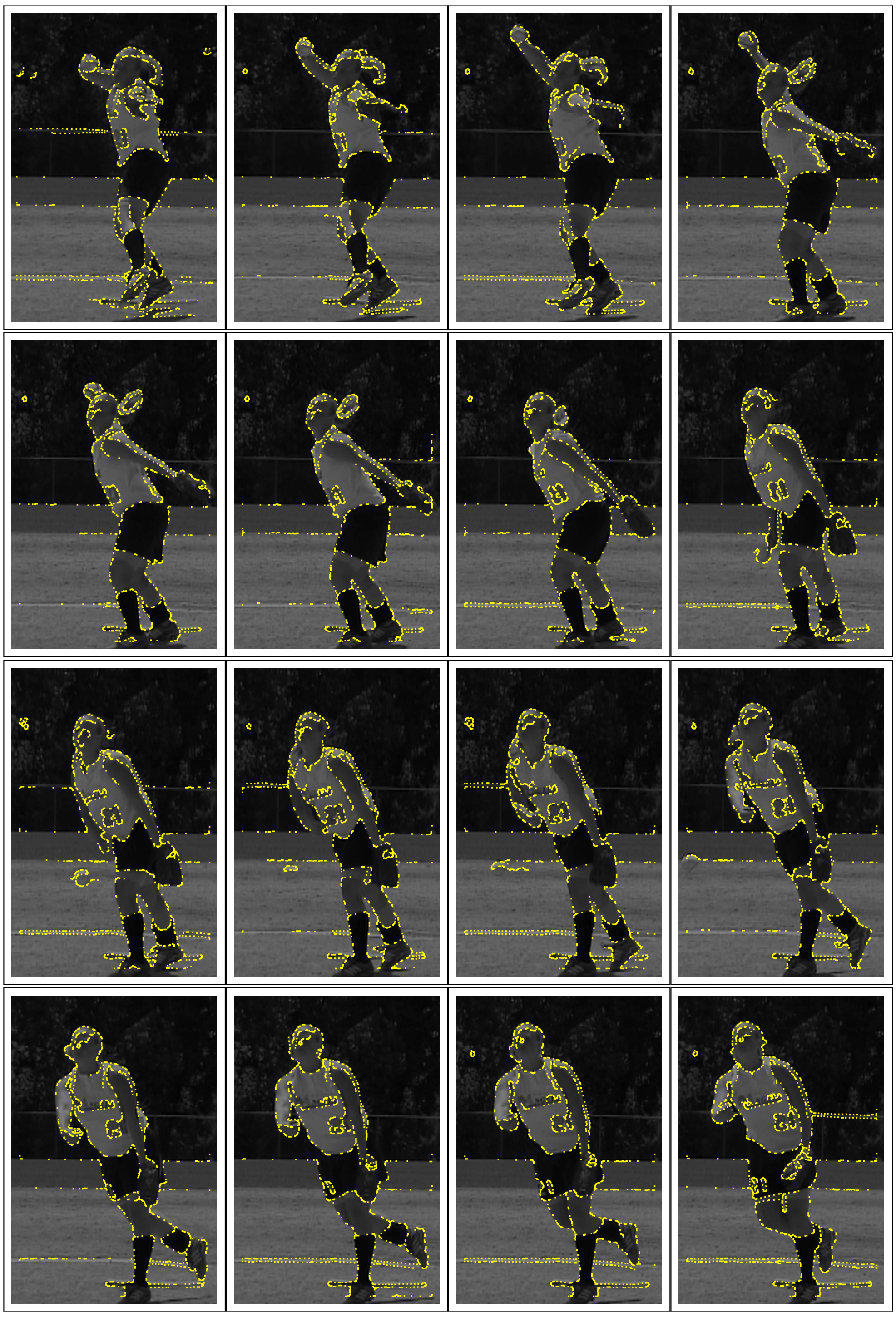


Figure 6.14: Results of the algorithm on the "pitcher" video sequence (in row-major order) for $\phi = 1$. Note that points (shown in yellow) in each ensemble have been enlarged nine time just for sake of clarity.

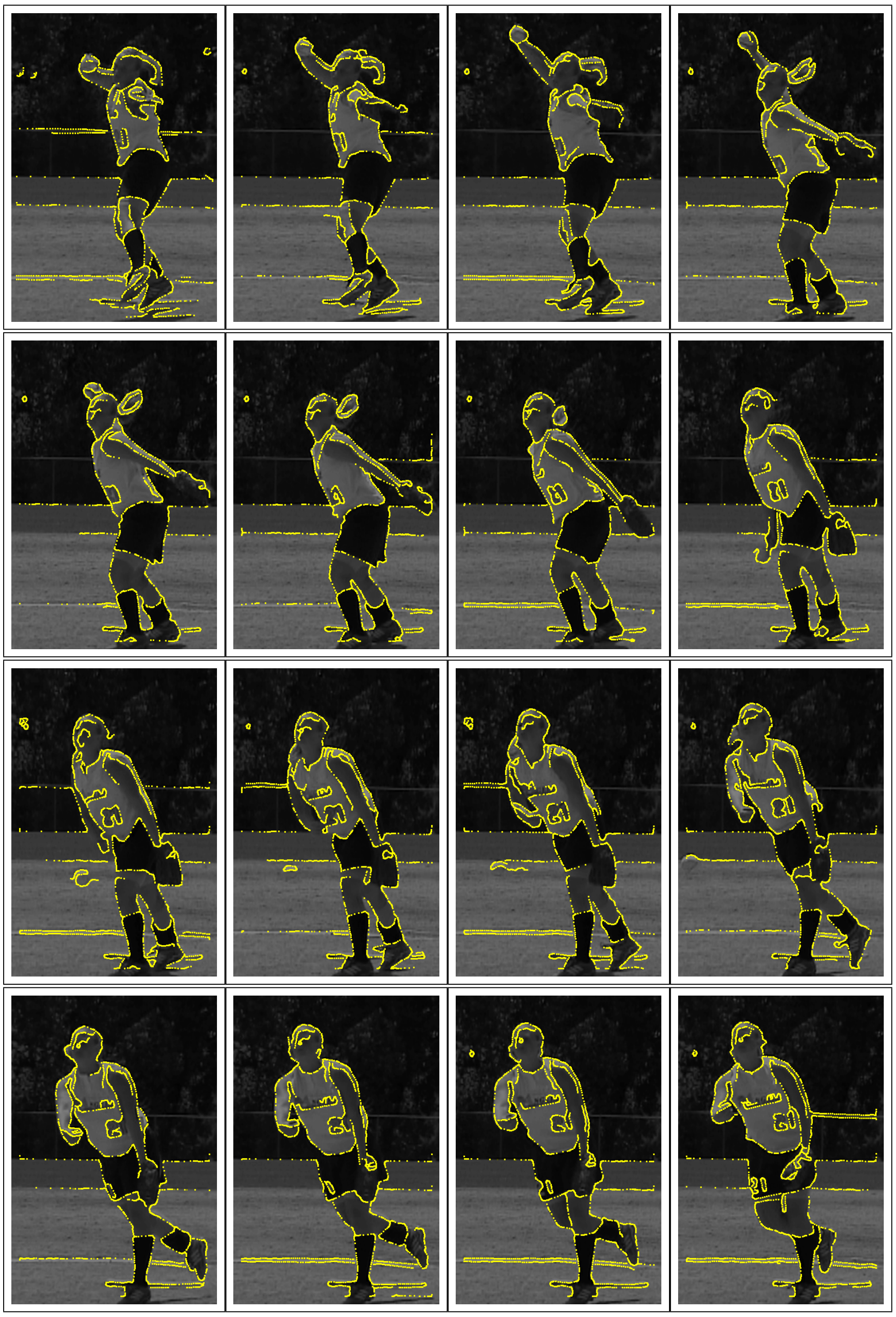
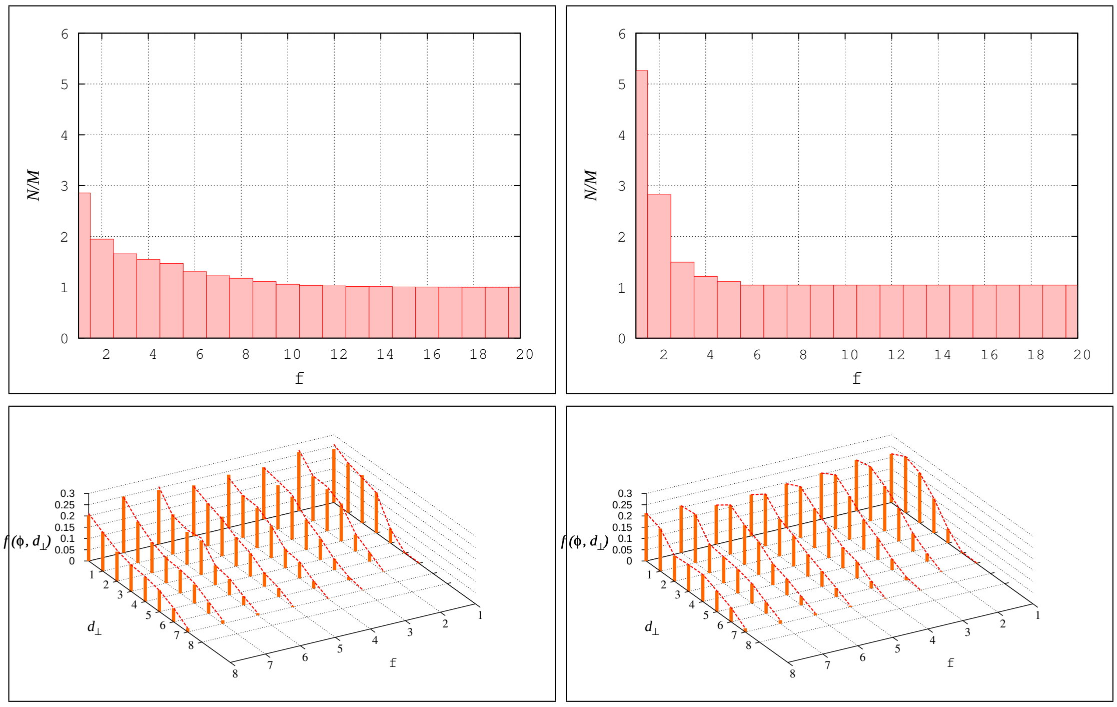


Figure 6.15: Results (enlarged points, as in Fig. 6.14) on the “pitcher” video sequence (in row-major order) for $\phi = 2$.



fvc2000:db2a pitcher-2

Figure 6.16: Plots showing the quality of sampling on the fingerprint database fvc2000:db2a (four images shown in Figs. 6.12 and 6.13) and on the video sequence "pitcher-2" (Figs. 6.14 and 6.15). The top row shows the variation of compression ratio CR $= N/M$ with the pointillist factor ϕ, and the bottom row shows the error frequency for $\phi = 1 - 8$.

Chapter 7

Archival Image Indexing with Connectivity Features using Randomized Masks

7.1 Introduction

With the advent of emerging digital libraries and multimedia, construction, maintenance, and efficient usage of several archives consisting of very large image databases, pose a great challenge in information technology. These databases are frequently searched for locating references, or for serving various kinds of indexing or retrieval queries. In archival image indexing, the emphasis is specially given on deciding whether a particular image exists in the database, rather than on finding the similarity matches of the given image. In most of the existing image indexing and retrieval systems, determination of feature vectors requires computationally intensive floating point operations. In this work, we have designed a novel fuzzy technique for indexing a large archive of binary images based on simple connectivity features such as the number of connected components and the number of holes. To ensure a unique index for each image, an iterative technique is employed on the bit-plane of the image using a pseudo-random spatial masking scheme. A data structure called *discrimination tree*, consisting of a hierarchical arrangement of *kd*-trees [Berg *et al.* (2000)] is also proposed for efficient indexing of images.

There are two principal approaches to image indexing: (i) those based on the features extracted from raw image data [Chang and Lee (1991), Faloutsos *et al.* (1994), Hirata and Kato (1992), Idris and Panchanathan (1995), Smith and Chang (1996), Swain and Ballard (1991)], and (ii) those based on the coefficients in the compressed or transformed domain [Jacobs *et al.* (1995), Liang and Kuo (1997), Ma and Manjunath (1994), Pentland *et al.* (1994), Wang *et al.* (1997)]. A majority of the techniques proposed in the image indexing literature extract robust features, meant for high similarity matches. There also exist some fuzzy-neural techniques [Verma and Kulkarni (2004)], techniques based on fuzzy sets [Gesù (1999), Gesù and Maccarone (1998)] and soft image distances [Gesù and Roy (2000, 2002)], and those based on genetic algorithms [Gesù and Bosco (2005), Steji *et al.* (2003)]. However, inexplicit organization of the feature space often raises various scalability issues. In particular, the query response time increases significantly with the database size. In this work, only the topological properties of an image are used, based on a new idea of randomized masking, to address the issue of feature space organization.

Topological properties of a digital image [Brimkov and Barneva (2004), Klette and Rosenfeld (2004a)] typically represent the underlying geometric shape of the image. These properties are resistant to the common spatial changes made to the image, such as stretching, rotation, scaling, translation, and other rubber-sheet transformations. One such important topological feature of a binary image is its Euler Number, which is defined as the number of connected components minus the number of holes present in the image [Gonzalez and Woods (1993), Pratt (1978)]. It is also used as one of the major features for image classification or indexing.

Euler number has numerous applications in various domains, e.g., medical diagnosis [Chen and Yan (1988), Pogue *et al.* (2000)], optical character recognition [Matas and Zimmermann (2005)], shadow detection [Rosin and West (1995)], document image processing [Srihari (1986)], and for constructing feature vectors [Stavrianopoulou and Anastassopoulos (2000)]. The strength and elegance of Euler number lie in its simplicity, ability to capture the overall structural property of a binary image, and in easy implementability. Furthermore, the availability of low-cost on-chip computation of Euler number [Bishnu *et al.* (2001)] makes it a very convenient feature for image indexing. Recently, several attempts have been made to exploit the strength and usefulness of Euler number for handling a gray-scale image with the help of "Euler vector" [Bishnu *et al.* (2005, 2006)], which is constructed from the Euler numbers corresponding to the four most significant bit-planes of a gray-scale image.

Although Euler number has certain discriminating ability, it alone cannot serve the purpose of uniquely identifying or indexing all the images in even a small or moderately-sized database. Hence, a few additional features, if judiciously selected along with the Euler number, may perform the desired task of discriminating all the images in a database.

In this work, we use Euler number, as well as the two basic spatial features from which it is derived, namely, the number of connected components, C, and the number of holes, H, along with a masking technique for characterizing an image uniquely. The striking feature of this simple, yet novel approach, lies in extracting these dual characteristics (i.e., C and

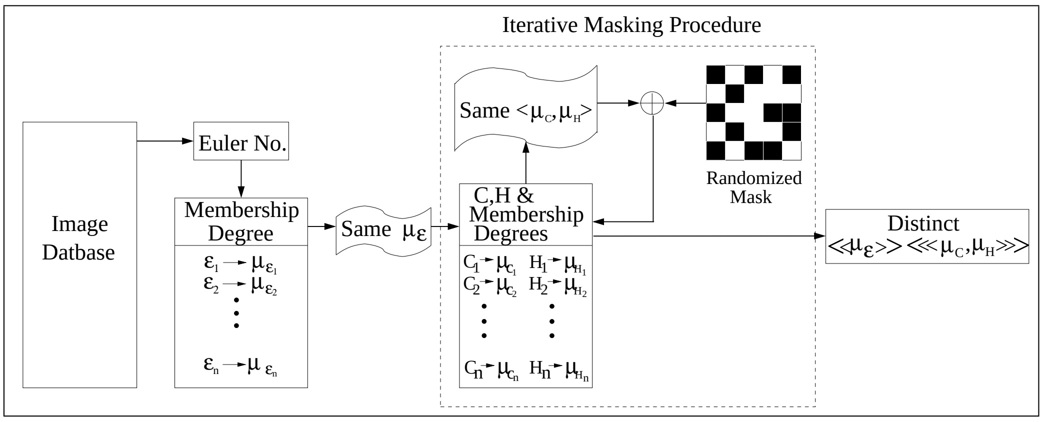

Iterative Masking Procedure
Image Datbase
Euler No.
Membership Degree
$\varepsilon_1 \rightarrow \mu_{\varepsilon_1}$
$\varepsilon_2 \rightarrow \mu_{\varepsilon_2}$
$\varepsilon_n \rightarrow \mu_{\varepsilon_n}$
Same μ_ε
C,H & Membership Degrees
$C_1 \rightarrow \mu_{C_1}$ $H_1 \rightarrow \mu_{H_1}$
$C_2 \rightarrow \mu_{C_2}$ $H_2 \rightarrow \mu_{H_2}$
$C_n \rightarrow \mu_{C_n}$ $H_n \rightarrow \mu_{H_n}$
Same $<\mu_C, \mu_H>$
Randomized Mask
Distinct $\ll\mu_\varepsilon\gg \lll\mu_C,\mu_H\ggg$

Figure 7.1: The scheme of the proposed method.

H) iteratively from a few derived images obtained by Boolean *xor*-ing the given image with a few synthetic pseudo-random masks. In each iteration, when C and H are obtained, a membership value for each image is assigned using a fuzzy membership function that captures the number of connected components and the number of holes of a given image. The fuzzy membership function is designed in such a way that the membership value of an image represents its topological characteristics with respect to the given database. The ultimate objective of the procedure is to generate a unique index (feature vector) for each binary image in the database. The overall scheme of the proposed method is shown in Fig. 7.1.

The proposed algorithm converges within a small number of iterations, and is easy to implement. The algorithm has been tested on three different image databases, and the results are found to be very encouraging. For example, when applied on a logo database of 1034 binary images, the algorithm converges within 3 iterations, thereby requiring only 3 randomized masks to uniquely characterize all of them. Only the seeds for the randomized masks are required to be saved as they are produced by a pseudo-random number generator. For this database, the length of the feature vector is at most 9, and on the average, it turns out to be 5. We have proposed an extended scheme to ensure rotational invariance, but it requires computation of the principal axes of an image.

7.2 Proposed Work

In a database consisting of N binary images, there should be N distinct feature vectors so that there is a one-to-one correspondence between each image and its representative vector.

Here, we demonstrate that simply the number of connected components and the number of holes of the original image, along with those computed iteratively for a few derived masked images, are sufficient to generate the desired feature vector of the original image. The membership values of the Euler number, the connected components and the holes, obtained over all the iterations, constitute the feature vector. It should be mentioned here that the images of the database are median-filtered before running the proposed indexing algorithm. The median filtering is required to eliminate the noisy components in a binary image, if any.

7.2.1 Euler Number and its Primary Features

As mentioned earlier, the image features that are used in our algorithm are connected components, holes, and the Euler number [Gonzalez and Woods (1993)], which are defined as follows:

Connected component: A connected component of a set is a subset of maximal size such that any two of its points can be joined by a connected curve (in 8-neighborhood) lying entirely in the subset.

Hole: A hole in a set is a region of the background, which is a connected component (in 4-neighborhood) and is completely enclosed by the object.

Euler number: The Euler number (genus) of an image is defined as the number of connected components minus the number of holes in the image.

Thus, if C and H denote the number of connected components and the number of holes in a digital image respectively, then its Euler number E is given by:

$$E = C - H \tag{7.1}$$

An efficient and easily implementable algorithm for computing the Euler number of a binary image is based on the exhaustive study of the local patterns [Pratt (1978)], which form the following three sets of 2×2 bit patterns, called *bit quads*:

$$Q_1 = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \right\}$$
$$Q_2 = \left\{ \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \right\}$$
$$Q_3 = \left\{ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right\}$$

If q_1, q_2, q_3 represent the respective number of patterns from the sets Q_1, Q_2, Q_3 in a binary image $\mathcal{I}$, then its Euler number in 8-neighborhood, is given by:

$$E(\mathcal{I}) = \frac{1}{4}(q_1 - q_2 - 2q_3) \tag{7.2}$$

It may be noted that, all the 10 bit quads in the sets Q_1, Q_2, and Q_3 are distinct. Hence, each of the 10 bit quads, when enumerated in row major order, yields a unique 4-bit number, thereby enabling the formation of a Look Up Table (LUT) containing the 10 unique elements of the following modified sets:

$Q_1 = \{1, 2, 4, 8\}$

$Q_2 = \{14, 13, 11, 7\}$

$Q_3 = \{6, 9\}$

There exist several efficient algorithms for computation of the Euler number [Bishnu *et al.* (2001), Chen and Yan (1988), Pratt (1978)]. We have implemented the algorithm [Pratt (1978)] for computing the Euler number in 8-neighborhood, using the above LUT.

7.2.2 The Fuzzy Model and Membership Function

A fuzzy-based technique is proposed here in accordance to the fuzzy model for the objects of the universe. The topological features of an image, namely the Euler number (E), the number of connected components (C), and the number of holes (H), are the objects of the universe taken individually. In this model, it is assumed that the set of values of each of the three topological features is a fuzzy set. The membership function $\mu_{X_i} : \mathcal{U} \rightarrow [0, 1]$, where X assumes E, C, or H; X_i, the feature value of the ith image, indicates the degree of resemblance of the ith image to the overall trend of the database. In other words, if the majority of the images in a database $\mathcal{D}$ possess low (resp. high) values of X, then an image with a low (resp. high) value of X in $\mathcal{D}$ will have a high membership value, whereas, an image with a high (resp. low) value of X in the same database $\mathcal{D}$ will have a low membership value. In general, the membership function corresponding to the ith image for the topological feature X, is defined as:

$$\mu_{X_i} = \frac{f_{X_{Mo}}}{F_{X_i}} \tag{7.3}$$

where, X can assume E, C, or H, $f_{X_{Mo}}$ is the modal (highest) frequency of the frequency distribution of X, and F_{X_i} denotes the cumulative frequency corresponding to X_i derived from the distribution of X sorted in the non-increasing order of frequency. As the modal frequency [Vertan and Boujemma (2000)] embodies the central tendency of the database, and as the cumulative frequency, F_{X_i}, is derived from the sorted frequency distribution, their ratio signifies the degree of membership of the ith image (having the feature value X_i) to the database. It may be noted that μ_X maps the set X into the interval $[0,1]$, and thus, X is a fuzzy set [Bezdek (1993), Zadeh (1965)].

Given a binary image database $\mathcal{D} = \{\mathcal{I}_k\}_{k=1}^{N}$, there may exist several images having identical values of Euler number. Let there be η distinct Euler numbers, $\epsilon_1, \epsilon_2, \ldots, \epsilon_\eta$, sorted in the non-increasing order of their frequencies, for all the images in $\mathcal{D}$. Let $f_{\epsilon_{Mo}}$ be the modal frequency, and let the cumulative frequency distribution be $F_{\epsilon_1}, F_{\epsilon_2}, \ldots, F_{\epsilon_\eta}$. Now, by replacing X with ϵ in Eqn. 7.3, the membership value of an image, having Euler number ϵ_i and cumulative frequency F_{ϵ_i}, is derived as:

$$\mu_{\epsilon_i} = \frac{f_{\epsilon_{Mo}}}{F_{\epsilon_i}} \tag{7.4}$$

Let $\mathcal{E}_t$ be the subset of $\mathcal{D}$, which contains the images with the same membership value. Since the membership values decided by the Euler numbers of different images are not enough to classify the images further, the number of connected components, C, and the number of holes, H, are chosen as the features to derive the corresponding membership values, which will further (partially or completely) segregate the images in each subset $\mathcal{E}_t$ of $\mathcal{D}$. We have adopted the algorithm given in [Gonzalez and Woods (1993)] for finding the number of connected components, $C_{\mathcal{I}_k}$, of each image $\mathcal{I}_k$. The membership values, μ_{C_k} and μ_{H_k}, corresponding to C_k and H_k of each image $\mathcal{I}_k$ are calculated in a similar way as done for computation of the Euler number by using Eqn. 7.3.

The tuple $\langle \mu_{C_k}, \mu_{H_k} \rangle$ is used to classify $\mathcal{D}$ further. Using this tuple, we have devised a scheme that can be employed to index all the N images distinctly in the database $\mathcal{D}$. The membership degree of C for one of the databases we have used, is shown in Fig. 7.2. It is evident from this figure that the images having their values of C close to the modal value possess a higher degree of membership, whereas those with their C values away from the modal value possess a low degree of membership.

7.2.3 Generation of Randomized Mask

In order to differentiate two or more binary images with identical membership values, $\langle \mu_\epsilon \rangle$ and $\langle \mu_C, \mu_H \rangle$, an artificial binary mask is generated pseudo-randomly. When this mask is Boolean *xor*-ed, pixel by pixel, with an image $\mathcal{I}_k$ having the membership tuple $\langle \mu_{C_{\mathcal{I}_k}}, \mu_{H_{\mathcal{I}_k}} \rangle$, a new image $\mathfrak{J}_k$ will be produced with $C_{\mathfrak{J}_k}$ and $H_{\mathfrak{J}_k}$, as the number of connected components and the number of holes respectively. The corresponding membership

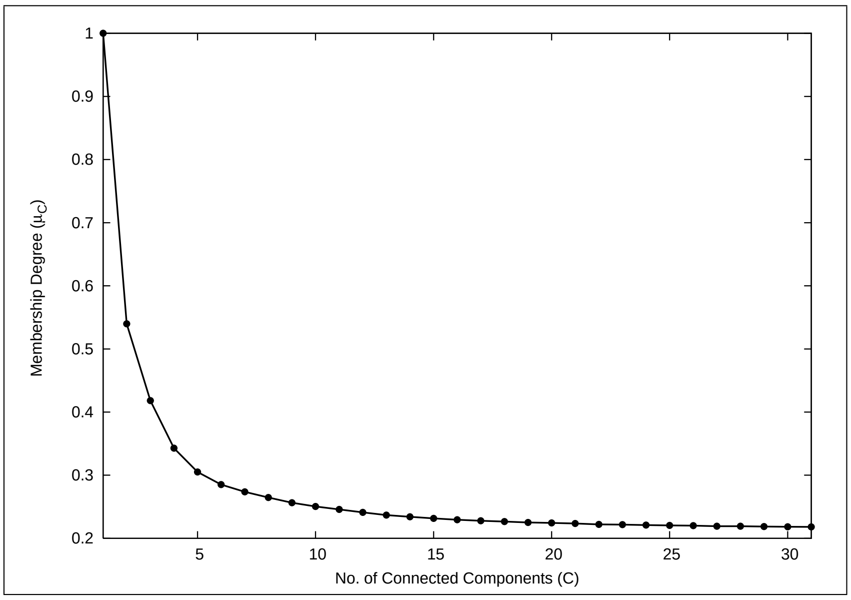


Figure 7.2: Membership degree for C corresponding to a given database.

tuple would be $\langle \mu_{C_{\mathcal{J}_k}}, \mu_{H_{\mathcal{J}_k}} \rangle$, which is possibly different from $\langle \mu_{C_{\mathcal{I}_k}}, \mu_{H_{\mathcal{I}_k}} \rangle$. In spite of its random characterization, a particular mask, however, may not be able to segregate all images in a particular set $\{\mathcal{I}_k\}$ having sufficiently different membership values, and therefore, more randomized masks may be required to resolve the residual ambiguity. Hence, the randomized masks are generated a few more times iteratively, so that after the final iteration, each of the N images in the database $\mathcal{D}$ is distinctly indexed.

It may be observed that the images in the database $\mathcal{D}$ may have nonuniform sizes. Hence, in order to have the compatibility of *xor*-operation between an image of $\mathcal{D}$ and a mask of predefined size, all the images in $\mathcal{D}$ are normalized to the size of the mask. Further, at each iteration, a single mask is produced for all images in $\mathcal{D}$, and the masks over all iterations have identical predefined size, $\mu \times \mu$ pixels. In our experiments, we have considered $\mu = 256$.

Let $\mathcal{A}_\alpha$ be a square matrix of size $\alpha \times \alpha$ (such that α divides μ), where, $\mathcal{A}_\alpha(x, y) \in \{0, 1\}$ for $1 \leq x, y \leq \alpha$. The matrix $\mathcal{A}_\alpha$ is constructed based on an eight-bit positive integer r, which is generated randomly each time for deciding each element of $\mathcal{A}_\alpha$. Let $r(x, y)$ be the random number that decides the entry $\mathcal{A}_\alpha(x, y)$. Then the decision for $\mathcal{A}_\alpha(x, y)$ is taken as follows:

$$\mathcal{A}_\alpha(x,y) = \begin{cases} 1 & \text{if } \; r(x,y) > \rho \\ 0 & \text{otherwise} \end{cases} \tag{7.5}$$

In Eqn. 7.5, the parameter ρ, which is an integer, plays a crucial role in determining the ratio of the number of 0s to that of 1s in $\mathcal{A}_\alpha$. Below it is shown that, assuming the uniform distribution of r in the integer range $[0, 255]$, the probability that the number of 0s is the same (or, differs by at most unity, if α is odd) as that of 1s in $\mathcal{A}_\alpha$, would be maximum for $\rho = 127$ [Roberts (1984)].

Let p be the probability that $\mathcal{A}_\alpha(x,y)$ is 1. Then the probability that $\mathcal{A}_\alpha$ contains i $1s$ and $(n-i)$ $0s$, denoted by $P(1^i)$, where $n = \alpha^2$, is given by

$$P(1^i) = \binom{n}{i} p^i (1-p)^{n-i} \tag{7.6}$$

Hence, from Eqn. 7.6, the probability of having equal number of $0s$ and $1s$ in $\mathcal{A}_\alpha$, i.e., $i = \frac{n}{2}$, n being assumed to be even, is given by

$$P(1^{\frac{n}{2}}) = \binom{n}{\frac{n}{2}} p^{\frac{n}{2}} (1-p)^{\frac{n}{2}},$$

and this probability is maximized when $p = \frac{1}{2}$. The same can also be proved if n is odd.

Now, in Eqn. 7.5, since ρ varies from 0 to 255, we have $p = \frac{255-\rho}{256}$. Thus, if p $= \frac{1}{2}$, then $\rho = 127$. Hence, the probability of having equal number of $0s$ and $1s$ in $\mathcal{A}_\alpha$, and in the mask generated thereof, is maximum for $\rho = 127$.

We assume that no *a priori* information is available about the images in the database. It is, therefore, important to set the parameter ρ to 127 in order to have equal number of $1s$ and $0s$ in the mask. In our experiments also, we have found that the algorithm requires minimum number of iterations for $\rho = 127$.

Further, a departure in the value of ρ from 127 will reduce the probability that the number of 0s is same as that of 1s in $\mathcal{A}_\alpha$. The generation of the synthetic mask depends on the generation of the corresponding $\mathcal{A}_\alpha$ in the concerned iteration, and therefore, changing the value of ρ will play a significant role in designing the masking scheme, if desired. In Fig. 7.3, different synthetic masks are shown, as obtained for different values of ρ.

Let $\mathcal{A}_\alpha^{(i)}$ be the matrix generated at the ith iteration, such that, the Hamming distance $L\left(\mathcal{A}_\alpha^{(i)}, \mathcal{A}_\alpha^{(i-1)}\right)$ between $\mathcal{A}_\alpha^{(i)}$ and $\mathcal{A}_\alpha^{(i-1)}$, $i \geq 1$, $\mathcal{A}_\alpha^{(0)} = \mathbf{0}$, satisfies the criterion stated in Eqn. 7.7.

$$L\left(\mathcal{A}_\alpha^{(i)}, \mathcal{A}_\alpha^{(i-1)}\right) = \sum_{x=1}^{\alpha} \sum_{y=1}^{\alpha} \mathcal{A}_\alpha^{(i)}(x,y) \sim \mathcal{A}_\alpha^{(i-1)}(x,y) \geq \tau \tag{7.7}$$

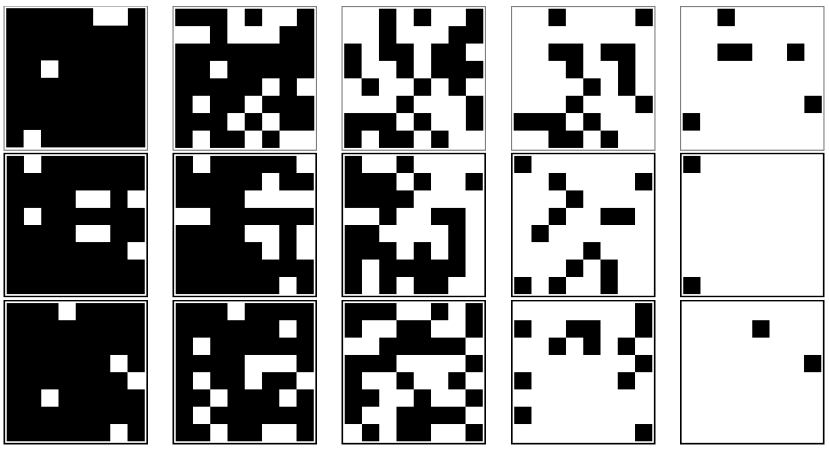


Figure 7.3: Instances of randomized masks for different values of ρ, given $\alpha = 8$.

$$\text{where, } \mathcal{A}_\alpha^{(i)}(x,y) \sim \mathcal{A}_\alpha^{(i-1)}(x,y) = 0, \text{ if } \mathcal{A}_\alpha^{(i)}(x,y) = \mathcal{A}_\alpha^{(i-1)}(x,y); \\ = 1, \text{ otherwise.}$$

Let $\mathcal{M}_\alpha^{(i)}$ be the randomized binary mask of size $\mu \times \mu$ generated in the ith iteration from $\mathcal{A}_\alpha^{(i)}$ as follows. Let $\beta (= \mu/\alpha)$ be the length of the square block that acts as the "building block" of $\mathcal{M}_\alpha^{(i)}$. That is, $\mathcal{M}_\alpha^{(i)}$ consists of α^2 building blocks, where, each building block is made of β^2 pixels (all of which are either 0 or 1 for a particular building block). Then $\mathcal{M}_\alpha^{(i)}$ is constructed as shown in the following equation:

$$\mathcal{M}_\alpha^{(i)}(x,y) = \mathcal{A}_\alpha^{(i)}(p,q) \tag{7.8}$$

where, $p = \left\lceil \frac{x}{\beta} \right\rceil, q = \left\lceil \frac{y}{\beta} \right\rceil, \forall x, \forall y, 1 \le x \le \mu, 1 \le y \le \mu,$ such that, the mask $\mathcal{M}_\alpha^{(i)}$ differs from the mask $\mathcal{M}_\alpha^{(i-1)}$ in the preceding iteration by at least $\tau\beta^2$ pixels, in conformity with Eqn. 7.7.

From Eqn. 7.7, it is evident that the size of the square blocks constituting the random synthetic mask is decided by α. This fact is illustrated in Fig. 7.4. It may be interesting to note that along with the value of ρ, the value of α may be also be tuned to obtain better resolving power of the synthetic mask. It is true that a smaller size of the blocks gives the mask more resolving power, but it may be less reliable from the point of view of robustness. We have found that $\alpha = 8$ gives the optimal solution for the logo and the stamp images used in our experiments.

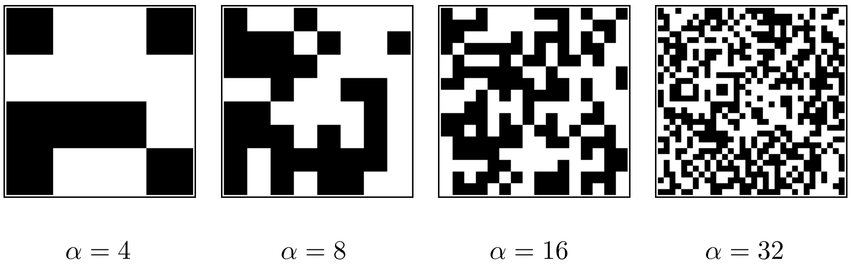


Figure 7.4: Different masks produced by changing the values of α for $\rho = 127$.

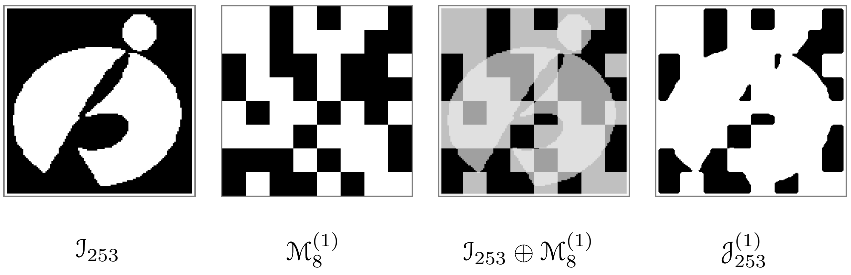

$\mathcal{I}_{253}$
$\mathcal{M}_8^{(1)}$
$\mathcal{I}_{253} \oplus \mathcal{M}_8^{(1)}$
$\mathcal{J}_{253}^{(1)}$

Figure 7.5: A randomized mask ($\mathcal{M}_8^{(1)}$) overlaid on a sample image $\mathcal{I}_{253}$ to obtain the masked image $\mathcal{J}_{253}^{(1)}$.

7.2.4 Iterative XOR-ing with Randomized Masks

Let $\mathcal{J}_k^{(i)}$ be the image obtained when *xor*-operation (represented by '$\oplus$') is performed between the image $\mathcal{I}_k$ and the mask $\mathcal{M}_\alpha^{(i)}$ in the ith iteration. In other words, $\mathcal{J}_k^{(i)} = \mathcal{I}_k \oplus \mathcal{M}_\alpha^{(i)}$. An example of *xor*-operation between a sample image with a randomized mask is shown in Fig. 7.5.

Let $\{\mathcal{I}_k\}$ be a subset of images in a database $\mathcal{D}$ with all of its *xor*-ed images $\{\mathcal{J}_k^{(i-1)}\}$ having identical $\langle \mu_C, \mu_H \rangle$ in the $(i-1)$th iteration. When each image $\mathcal{I}_k$ in this subset is *xor*-ed with $\mathcal{M}_\alpha^{(i)}$ in ith iteration to get a set of images, namely, $\{\mathcal{J}_k^{(i)}\} = \{\mathcal{I}_k \oplus \mathcal{M}_\alpha^{(i)}\}$, there would be possibly different membership degree tuples, $\langle \mu_C, \mu_H \rangle$, in the set $\{\mathcal{J}_k^{(i)}\}$, thereby requiring further masking of the images in the $(i+1)$th iteration of the algorithm. The iterations are carried on until all images with the same $\langle \mu_E \rangle$ have distinct ordered set of tuples, $\big\langle \langle \mu_C, \mu_H \rangle \big\rangle$, considered over all the iterations. In effect, all N images in the database $\mathcal{D}$ have N distinct feature vectors (of possibly unequal sizes), each of which is of the form $\Big\langle \langle \mu_{E_k} \rangle \big\langle \langle \mu_{C_k}^{(i)}, \mu_{H_k}^{(i)} \rangle \big\rangle_{i=0}^{i \le j} \Big\rangle$ corresponding to a unique image $\mathcal{I}_k \in \mathcal{D}$, where, j is the total number of iterations required for $\mathcal{D}$, and $\langle \mu_{C_k}^{(0)}, \mu_{H_k}^{(0)} \rangle$ is derived from $\langle C, H \rangle$ values of $\{\mathcal{I}_k\}$ without any *xor*-ing. A schematic representation of the entire algorithm has been

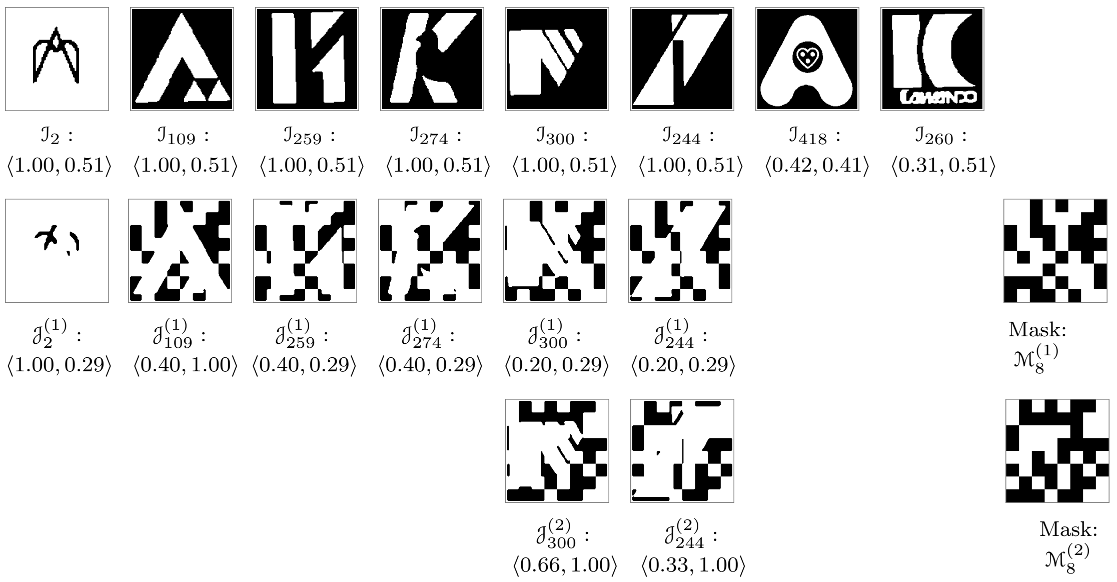

$\mathcal{I}_2$: ⟨1.00, 0.51⟩
$\mathcal{I}_{109}$: ⟨1.00, 0.51⟩
$\mathcal{I}_{259}$: ⟨1.00, 0.51⟩
$\mathcal{I}_{274}$: ⟨1.00, 0.51⟩
$\mathcal{I}_{300}$: ⟨1.00, 0.51⟩
$\mathcal{I}_{244}$: ⟨1.00, 0.51⟩
$\mathcal{I}_{418}$: ⟨0.42, 0.41⟩
$\mathcal{I}_{260}$: ⟨0.31, 0.51⟩
$\mathcal{J}_2^{(1)}$: ⟨1.00, 0.29⟩
$\mathcal{J}_{109}^{(1)}$: ⟨0.40, 1.00⟩
$\mathcal{J}_{259}^{(1)}$: ⟨0.40, 0.29⟩
$\mathcal{J}_{274}^{(1)}$: ⟨0.40, 0.29⟩
$\mathcal{J}_{300}^{(1)}$: ⟨0.20, 0.29⟩
$\mathcal{J}_{244}^{(1)}$: ⟨0.20, 0.29⟩
Mask: $\mathcal{M}_8^{(1)}$
$\mathcal{J}_{300}^{(2)}$: ⟨0.66, 1.00⟩
$\mathcal{J}_{244}^{(2)}$: ⟨0.33, 1.00⟩
Mask: $\mathcal{M}_8^{(2)}$

Figure 7.6: Demonstration of the proposed algorithm on a representative set of 8 images for generation of feature vectors of varying sizes by iterative *xor*-ing with randomized masks.

shown earlier in Fig. 7.1.

Fig. 7.6 demonstrates the iterative *xor*-ing procedure and generation of requisite feature vectors on a small representative set of 8 images. The *xor*-ing procedure converges just in 2 iterations; this indicates the speed and efficiency of the proposed method. In the 1st row, the membership tuple, $\langle \mu_C^{(0)}, \mu_H^{(0)} \rangle$ for all eight images are shown. It is evident that the images $\mathcal{I}_{418}$ and $\mathcal{I}_{260}$ are segregated at this level and the other six images are still having the similar membership tuple. Hence, these six images with $\langle \mu_C^{(0)}, \mu_H^{(0)} \rangle = \langle 1.00, 0.51 \rangle$ are selected for *xor*-ing in iteration (1). The 2nd row shows the *xor*-ed images in iteration (1) and the resultant tuples, $\langle \mu_C^{(1)}, \mu_H^{(1)} \rangle$. In the 2nd row, excepting the two images, $\mathcal{J}_{244}^{(1)}$ and $\mathcal{J}_{300}^{(1)}$, all other images have distinct $\langle \mu_C^{(1)}, \mu_H^{(1)} \rangle$, and the formation of their feature vectors, therefore, ends here. Since the images $\mathcal{J}_{244}^{(1)}$ and $\mathcal{J}_{300}^{(1)}$ have identical $\langle \mu_C^{(1)}, \mu_H^{(1)} \rangle = \langle 0.20, 0.29 \rangle$, their parent images, $\mathcal{I}_{244}$ and $\mathcal{I}_{300}$, are considered in the 2nd iteration, where they are *xor*-ed with the corresponding randomized mask, and their $\langle \mu_C^{(2)}, \mu_H^{(2)} \rangle$ values become different, as shown in the 3rd row. Thus, the process terminates after iteration (2). The two randomized masks, $\mathcal{M}_8^{(1)}$ and $\mathcal{M}_8^{(2)}$, which are used for *xor*-ing in the above two iterations, are shown in the respective rows.

Thus, out of the 8 images comprising the exemplary set in Fig. 7.6, the length of the feature vector for image $\mathcal{I}_{260}$ (for unique $\mu_\epsilon = 0.1842$) is just 1, that for $\mathcal{I}_{418}$ (no *xor*-ing required) is $1 + 2 = 3$, whereas, the same for images $\mathcal{I}_{244}$ and $\mathcal{I}_{300}$ (*xor*-ed up to 2nd iteration) is $1 + 2 \times 3 = 7$ each, and for the remaining 4 images (*xor*-ed in 1st iteration only), the length of the feature vector for each, becomes $1 + 2 \times 2 = 5$. Thus, the length of the feature vector in the proposed algorithm may vary from image to image. Similar images are more likely to have longer feature vectors, since they cannot be distinguished among themselves with a small number of features, whereas, an image, possessing lesser similarity with the rest of the images in the database, is likely to have a shorter feature vector.

7.2.5 Variable Feature Vector

As evident from Sec. 7.2.4, the feature vectors obtained for various images are characterized by variable feature length, depending on their structural behavior apropos the iterative *xor*-ing with the randomized masks used in our algorithm. In most of the conventional procedures, the length of a feature vector (sometimes the feature vector itself) is predefined. In contrast, an adaptive construction of feature vectors, depending on the need and the topological properties of an image, is done here based on the output of each iteration of the randomized masking procedure. This greedy algorithm provides an effective and near-optimal choice of feature vectors. Furthermore, the dimensionality of the feature vector that has maximal length in the concerned space, would increase automatically with the execution of the algorithm on a database of larger size and diversity. The number of iterations in the randomized masking procedure increases only with an appreciable increase in the size and diversity of the image database.

7.2.6 Algorithm for Computing the Feature Vector

```
COMPUTE_FEATURE_VECTOR

Input : Image database $\mathcal{D}$ consisting of $N$ images
Output : Feature vectors of each image

Steps:
1.  set parameters $\alpha, \rho, \mu, \tau$ (default: 8, 127, 256, 8)
2.  for $k = 1$ to $N$
3.      normalize the image $\mathcal{I}_k$ and find its Euler number $E_k$
```

```
4.  find the frequency distribution of the Euler nos. obtained in steps (2)-(3)
    to get the distinct Euler nos., say $\eta$ in number; hence obtain $\eta$ (disjoint)
    subsets of $\mathcal{D}$, namely $\mathcal{E}_t$, such that $\bigcup_{t=1}^{n} \mathcal{E}_t = \mathcal{D}$
5.  for $t = 1$ to $\eta$
6.      for each image $\mathcal{I}_k \in \mathcal{E}_t$
7.          find $\langle C_k^{(0)}$ and $H_k^{(0)}\rangle$, and calculate
            their membership values to get $\langle\langle\mu_{C_k}^{(0)}, \mu_{H_k}^{(0)}\rangle\rangle$
8.  for $t = 1$ to $\eta$
9.      construct the subsets of $\mathcal{E}_t$, s.t. each $\mathcal{S}_{t,u}$ ($u$th sub-subset of $\mathcal{E}_t$)
        contains the images having identical $\langle\mu_C, \mu_H\rangle$
10.     if $\mathcal{S}_{t,u}$ has only one image
11.         then add this to the final solution of feature vectors [step 22]
12. initialize $i = 1$
13. generate a randomized mask $\mathcal{M}_\alpha^{(i)}$ conforming to Eqns. 7.7 and 7.8 and
    put the corresponding $\mathcal{A}_\alpha^{(i)}$ in the final solution [step 22]
14. for all $t$
15.     for all $u$
16.         MASKED_VECTOR($\mathcal{S}_{t,u}$, $\mathcal{M}_\alpha^{(i)}$, $i$)
17. remove all subsets $\mathcal{S}_{t,u}$ used in steps 14-16 and (re)name the new subsets as $\mathcal{S}_{t,u}$
18. if at least one $\mathcal{S}_{t,u}$ contains at least two images
19.     then $i = i + 1$ and go to step 13
20. else
21.     assign $n = i$, and go to step 22
22. return the final solution:
        (i) the number of masks (iterations), $n$
        (ii) $\langle\mathcal{A}_\alpha^{(i)}\rangle_{i=1}^{i=n}$
        (iii) $\langle\mu_{E_k}\rangle\langle\langle\mu_{C_k}^{(i)}, \mu_{H_k}^{(i)}\rangle\rangle_{i=0}^{i\leq j}$, $1 \leq k \leq N$
```

```
Procedure MASKED_VECTOR($\mathcal{S}_{t,u}$, $\mathcal{M}_\alpha^{(i)}$, $i$)

1.  for each image $\mathcal{I}_k \in \mathcal{S}_{t,u}$
2.      evaluate $\mathcal{J}_k^{(i)} = \mathcal{I}_k \oplus \mathcal{M}_\alpha^{(i)}$
3.      find $C_k^{(i)}$ and $H_k^{(i)}$ of $\mathcal{J}_k$
4.  compute the frequency distributions of $C$ and $H$
5.  determine the mode frequencies of $C$ and $H$
```

```
6. for each image $\mathcal{I}_k \in \mathcal{S}_{t,u}$
7.         calculate the membership tuple $\langle \mu_{C_k}, \mu_{H_k} \rangle$
8.         append the tuple $\langle \mu^{(i)}_{C_k}, \mu^{(i)}_{H_k} \rangle$ to $\langle \langle \mu^{(j)}_{C_k}, \mu^{(j)}_{H_k} \rangle \rangle^{i-1}_{j=0}$ to get $\langle \langle \mu^{(j)}_{C_k}, \mu^{(j)}_{H_k} \rangle \rangle^{i}_{j=0}$
9. return the subsets of $\mathcal{S}_{t,u}$, with identical $\langle \mu^{(i)}_{C_k}, \mu^{(i)}_{H_k} \rangle$
```

7.2.7 Data Structure for Storing Image Features

In order to store and retrieve the images using the proposed algorithm, a data structure called *discrimination tree*, consisting of a sequence of nested *kd*-trees [Berg *et al.* (2000)], is used here. It may be noted that the quad-tree, which is commonly used in many of the existing image indexing methodologies [Ahmad and Grosky (1997)], is not suitable for storing the feature vectors (the images, thereof) derived in this algorithm, since the quadtree would become very unbalanced because of the uneven distribution of the membership value tuples $\langle \mu_C, \mu_H \rangle$. A *kd*-tree, on the other hand, would be always balanced and would ensure searching of an image in logarithmic time, since it splits a 2-dimensional plane alternately about the median w.r.t. the x-coordinate and about the median w.r.t. the y-coordinate, such that at each level of the *kd*-tree, the points stored at any two nodes differ in number at most by unity.

Let there be $n^{(i)}$ distinct tuples of $\langle \mu^{(i)}_C, \mu^{(i)}_H \rangle$ produced in the ith ($i = 0, 1, 2, \ldots, n$) iteration of the algorithm for a database $\mathcal{D}$ containing N images. At the ith iteration, the corresponding feature tuples, $\langle \mu^{(i)}_C, \mu^{(i)}_H \rangle$, on a 2-dimensional plane, namely the μ_{CH} plane, constitute a planar set of points, which may be stored in a 2-dimensional *kd*-tree. Let $\mathcal{T}^{(i)}$ represent the *kd*-tree that is constructed at the ith iteration. Then $\mathcal{T}^{(i)}$ contains $n^{(i)}$ leaf nodes and has $\mathcal{O}(\log n^{(i)})$ height, where, each leaf node contains a unique $\langle \mu_C, \mu_H \rangle$ tuple that may be the $\langle \mu_C, \mu_H \rangle$ tuple for more than one image (after *xor*-ing with mask $\mathcal{M}^{(i)}_\alpha$, if $i \geq 1$, and original, if $i = 0$) in $\mathcal{D}$, as shown in Fig. 7.7.

It is quite evident that, if at least one of the leaf nodes of $\mathcal{T}^{(i)}$ has more than one image of $\mathcal{D}$ associated with it, only then the algorithm proceeds for the $(i+1)$th iteration. For each such leaf node ν, having more than one associated image in $\mathcal{T}^{(i)}$, another *kd*-tree $\mathcal{T}^{(i+1)}_\nu$ would be created in the$(i+1)$th iteration. The *kd*-tree $\mathcal{T}^{(i+1)}_\nu$ contains all distinct $\langle \mu^{(i+1)}_C, \mu^{(i+1)}_H \rangle$ tuples of only those images associated with the same node ν of $\mathcal{T}^{(i)}$. The process is repeated until each leaf node in the *kd*-tree has exactly one image associated with it, as stated in the algorithm of Sec. 7.2.6. The tree, thus created with a number

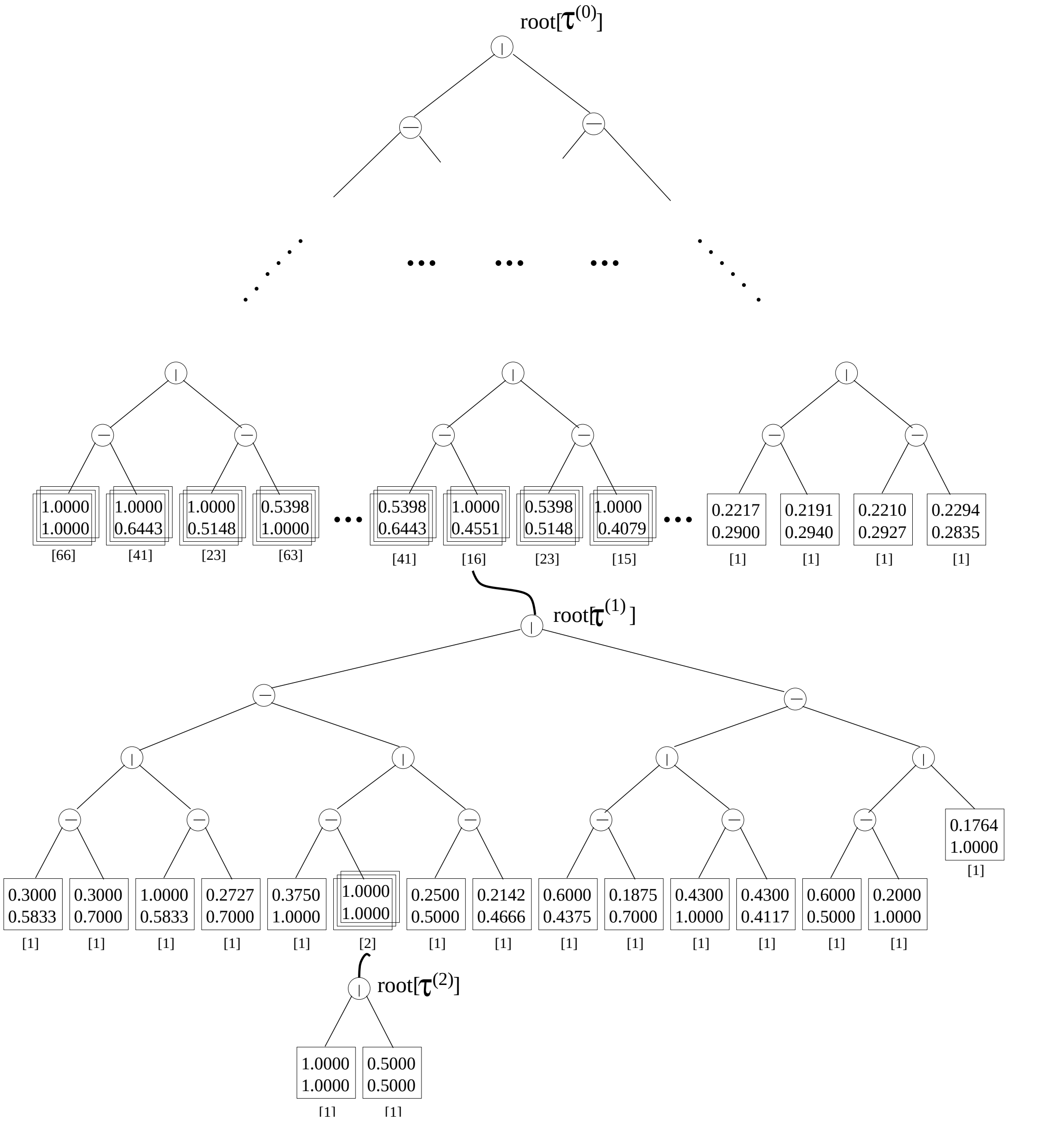


Figure 7.7: Discrimination tree for database D1 (refer Sec. 7.2.7). The $\langle \mu_C, \mu_H \rangle$ tuples are mentioned in the leaf nodes indicated by the square boxes. The numbers written inside the square braces ([ ]) below the leaf nodes show the numbers of images having the identical $\langle \mu_C, \mu_H \rangle$ tuple. This figure indicates that, at level one, there are 247 distinct $\langle \mu_C, \mu_H \rangle$ tuples, and corresponding to $\langle 1.0000, 0.4551 \rangle$, 16 images are associated. The next level *kd*-tree, only one shown here for $\langle \mu_C, \mu_H \rangle = \langle 1.0000, 0.4551 \rangle$, has 15 distinct leaf nodes. Finally, at the third level, the two images with $\langle \mu_C, \mu_H \rangle = \langle 1.0000, 1.0000 \rangle$ are resolved.

of *kd*-trees organized in a hierarchical manner, is termed as the *discrimination tree*, as it discriminates the images having the same membership degree, $\langle \mu_C, \mu_H \rangle$. The root of each *kd*-tree at each level contains the iteration number, the seed of the randomized mask used for that iteration, the modal frequencies, and the frequency distributions of C and H. The root of the discrimination tree, being the root of a *kd*-tree at level zero, does not store any seed for the randomized mask, since no *xor*-ing is done at this iteration. It may be noted that the maximum length of the feature vector is bounded by the depth of the discrimination tree.

7.2.8 Image Retrieval

The query image, $\mathcal{I}_q$, is median-filtered and normalized. The C and H values of this preprocessed image are then computed. Now, the membership tuple $\langle \mu_{C_q}, \mu_{H_q} \rangle$ of $\mathcal{I}_q$, corresponding to the given database $\mathcal{D}$, is determined using the mode values, Mo_C and Mo_H stored at the root of the discrimination tree. Note that (as stated in Sec. 7.2.7) the root of each *kd*-tree will contain the modes of C and H of the images associated with it. Searching in the discrimination tree for this $\langle \mu_{C_q}, \mu_{H_q} \rangle$ tuple begins at the root ($\mathcal{T}^{(0)}$), as shown in Fig. 7.7. When the search results in a leaf node having more than one image associated with it, it proceeds to the root of the next level *kd*-tree ($\mathcal{T}^{(1)}$). The root contains the seed of the mask at this level, as well as the Mo_C and the Mo_H of $\mathcal{T}(1)$. The query image is *xor*-ed with this mask, its membership tuple $\langle \mu_{C_q}, \mu_{H_q} \rangle$ is calculated, and $\mathcal{T}^{(1)}$ is searched with this tuple. This process is repeated until a leaf node with a single image is found. If a leaf node is not found at any level of the search, then it is concluded that the query image does not exist in the database.

The major steps of the retrieval algorithm is as follows.

```
1.  preprocess the query image to $\mathcal{I}_q$
2.  set $root = root(\mathcal{T}^{(0)})$ and $i = 0$
3.  **while** (TRUE)
4.          **if** $i = 0$
5.                  **then** set $\mathcal{J}_q^{(i)} = \mathcal{I}_q$
6.          **else**
7.                  compute $\mathcal{J}_q^{(i)} = \mathcal{I}_q \oplus \mathcal{M}_\alpha^{(i)}$
8.          find $C_q^{(i)}$ and $H_q^{(i)}$ of $\mathcal{J}_q^{(i)}$
9.          compute $\langle \mu_{C_q}^{(i)}, \mu_{H_q}^{(i)} \rangle$ using the modal frequencies stored
                  at the *root* of the *kd*-tree $\mathcal{T}^{(i)}$
```

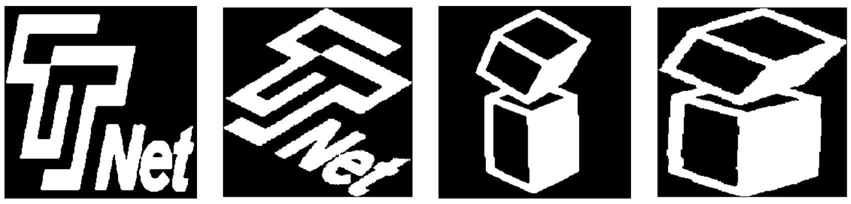


Figure 7.8: Two images and the corresponding rotated images

```
10.   search the kd-tree $\mathcal{T}^{(i)}$ with $\langle \mu^{(i)}_{C_q}, \mu^{(i)}_{H_q} \rangle$
11.   if search returns NULL
12.       then declare that the image is not in the database and return
13.   else
14.       if the resultant leaf node points to the next level kd-tree
15.           then set $i = i + 1$ and $root = root(\mathcal{T}^{(i)})$
16.       else
17.           report the details of the image
```

7.2.9 Translation, Scaling, and Rotation Invariance

The rotation invariance is achieved by rotating the image to make its principal axis vertical using principal component analysis. The *discrimination tree* is constructed by *xor*-ing these rotationally invariant images with the randomized mask(s). In order to achieve translation and scaling invariance, the foreground is extracted from the background and normalized to the size $\mu \times \mu$. The scheme is shown for two example images in Fig. 7.8.

Following are the preprocessing steps in brief, applied on each image of the database before indexing, to make it invariant to any kind of affine transformation:

(i) compute the principal axis;
(ii) rotate the image to make the principal axis vertical;
(iii) extract the foreground from the background;
(iv) median-filter and normalize the image to size $\mu \times \mu$.

7.2.9.1 Insertion and Deletion of an Image

A new image can be indexed easily by inserting the image at an appropriate place in the discrimination tree. At first, C and H values of the new image are computed, and the frequency distributions of C and H are updated to (re)compute Mo_C and Mo_H. Then, the membership tuple $\langle \mu_C, \mu_H \rangle$ is determined. Next, the $\langle \mu_C, \mu_H \rangle$ tuple is dynamically inserted [Overmars (1983)] into the *kd*-tree $\mathcal{T}^{(0)}$. If this tuple is a distinct (new) $\langle \mu_C, \mu_H \rangle$ tuple, then it is added as a new node in this *kd*-tree. Otherwise, it has a *kd*-tree at the next level associated with it. In such a case, the image is *xor*-ed with the corresponding mask stored at the root of the *kd*-tree at the next level, and the $C_\oplus$ and $H_\oplus$ values are calculated for the *xor*-ed image. The modes of this *kd*-tree are iteratively modified if required, depending on the previous distribution. The membership tuple $\langle \mu_{C\oplus}, \mu_{H\oplus} \rangle$ of the new *xor*-ed image is calculated, and $\langle \mu_{C\oplus}, \mu_{H\oplus} \rangle$ is inserted into the next level *kd*-tree.

Thus, the procedure for inserting a new node is followed for the *kd*-tree at the root ($\mathcal{T}^{(i)}$) iteratively until this new node is inserted as a single node. However, a threshold, for the number of images associated with a leaf node, may be imposed (say, a maximum of 5), below which the next-level *kd*-tree is not constructed. Rather, the images are inserted in a lateral linked-list and discriminated by the $\langle C, H \rangle$ tuple of the corresponding *xor*-ed images with a local randomized mask. In such a case, the leaf node stores the seed for the local randomized mask.

The deletion of an image from the discrimination tree is performed in a similar way, since a deleted image implies deletion or modification of the associated leaf node. However, the mode values stored at the roots of the *kd*-trees have to be reassessed and have to be replaced with the new values, if changed. Further, if the leaf node indicates a next iteration, then the number of images associated with this node is decremented, $\langle \mu_C, \mu_H \rangle$ of the *xor*-ed image is computed for the next iteration, and the corresponding leaf node is associated with the next level *kd*-tree. The process is repeated until the final leaf node representing the image is located and deleted. The mode frequencies stored at the roots of the *kd*-trees are updated accordingly.

7.3 Experimental Results

We have implemented the proposed method in C on a Sun_Ultra 5_10, Sparc, 233 MHz, the OS being the SunOS Release 5.7 Generic. A tool called CONFERM (Connectivity Features with Randomized Masks) [Biswas *et al.* (2004)] has been developed for this purpose. The method has been tested on 3 sets of binary images for our experiments: (i) database D1 of 1034 logo images; (ii) database D2 of 106 logo images; and (iii) database D3 of 2598 stamp images. The average execution time and average query time for each of these three sets of images are given in Table 7.1 and in Table 7.2 respectively. The summary of results

Table 7.1: Average CPU time in seconds per image for indexing databases D1, D2, and D3

	D1	D2	D3
$\mu_E, \langle \mu_C^{(0)}, \mu_H^{(0)} \rangle$	0.476	0.508	0.510
$\langle \mu_C^{(i)}, \mu_H^{(i)} \rangle_{i=1}^{i=j}$	0.422	0.120	0.080
Total	0.898	0.628	0.588

obtained for databases D1, D2 and D3, is presented in Table 7.3. The results vary with the databases and also with the different values of α. This observation reflects the randomized and adaptive nature of the proposed algorithm.

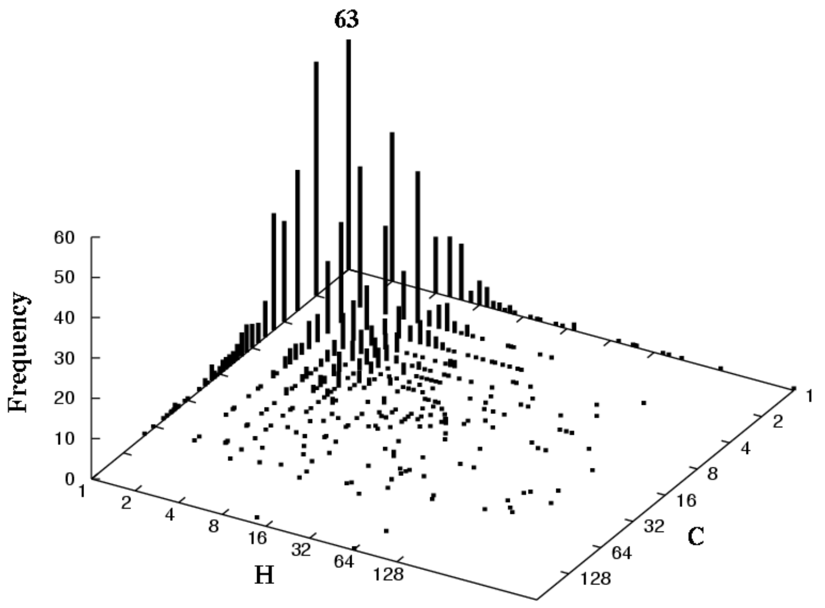
In Table 7.3, $\overline{\lambda}$ denotes the average length of feature vector for the database $\mathcal{D}$, which is given in Eqn. 7.9, where, λ_k denotes the feature vector length of the image $\mathcal{I}_k \in \mathcal{D}$.

$$\overline{\lambda} = \frac{1}{N} \sum_{k=1}^{k=N} \lambda_k \tag{7.9}$$

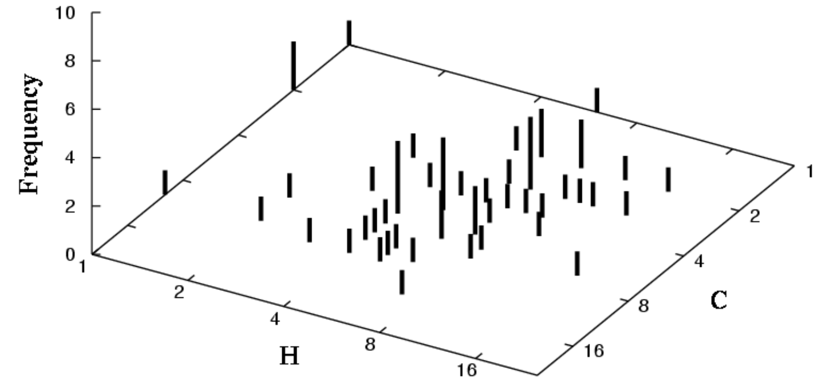
Table 7.2: Average query time in seconds per image for databases D1, D2 and D3

	D1	D2	D3
No. of Images	1034	106	2598
Avg. Query Time	0.0584	0.0595	0.0635

Fig. 7.9 shows the reduction in frequency of occurrences of identical $\langle \mu_C, \mu_H \rangle$ tuples with the progress of iterations in the algorithm, and demonstrates the segregation power of the indexing scheme. To cite a few examples, four sample images from D1 and another four from D2 have been shown in Fig. 7.10 and Fig. 7.11 respectively. These images have been selected so as to represent the possible output vectors of the proposed algorithm. Similarly, a representative set of images from the stamp image database D3 are shown in Fig. 7.12. For instance, in Fig. 7.10, $\mathcal{I}_{761}$ is the only image in D1 with distinct $\mu_E = 0.1842$ and therefore, no other feature is required to uniquely index this image. The next image shown, on the other hand, needs the tuple $\langle \mu_{C^{(0)}}, \mu_{H^{(0)}} \rangle = \langle 0.3642, 0.3131 \rangle$ along with its $\mu_E = 0.2235$ since there are some more images in D1 having $\mu_E = 0.2235$; there is, however, no other image in D1 with $\mu_E = 0.2235$ and $\langle \mu_{C^{(0)}}, \mu_{H^{(0)}} \rangle = \langle 0.3642, 0.3131 \rangle$. Hence, the



Frequency distribution of $\langle C^{(0)}, H^{(0)} \rangle$



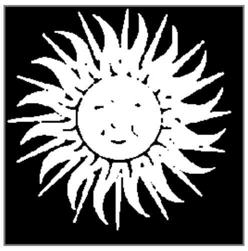
Frequency distribution of $\langle C^{(1)}, H^{(1)} \rangle$ for 63 images in D1 with same membership degree, $\langle \mu_C^{(0)}, \mu_H^{(0)} \rangle$

Figure 7.9: Distribution of $\langle C, H \rangle$ for Database D1.

Table 7.3: Results for databases D1, D2, and D3

	D1			D2			D3		
α	8	16	32	8	16	32	8	16	32
$\langle\mu_E\rangle$	62	62	62	53	53	53	162	162	162
$\#\langle\mu_C,\mu_H\rangle^{(0)}$	247	247	247	79	79	79	1281	1281	1281
$\#\langle\mu_C,\mu_H\rangle^{(1)}$	710	948	1007	106	106	106	1742	1887	2035
$\#\langle\mu_C,\mu_H\rangle^{(2)}$	967	1034	1034	–	–	–	2334	2402	2476
$\#\langle\mu_C,\mu_H\rangle^{(3)}$	1034	–	–	–	–	–	2598	2598	2598
$\overline{\lambda}$	5.16	4.57	4.45	2.51	2.51	2.51	4.63	4.47	4.46

$\#\langle\mu_C,\mu_H\rangle^{(i)}$= No. of distinct vectors after ith iteration.
Each feature vector after ith iteration is given by:
$\langle\mu_E\rangle\langle\mu_C^{(0)},\mu_H^{(0)}\rangle\langle\mu_C^{(1)},\mu_H^{(1)}\rangle\cdots\langle\mu_C^{(i)},\mu_H^{(i)}\rangle$.


$\mathfrak{I}_{761}$

$\mathfrak{I}_{174}$

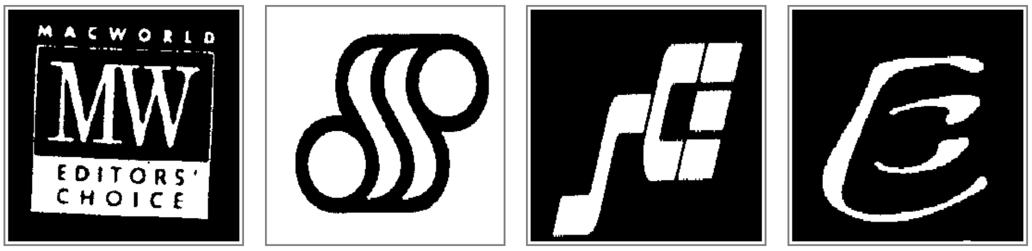
$\mathfrak{I}_{162}$

$\mathfrak{I}_{261}$

$\mathfrak{I}_{761}$: $\langle 0.1842\rangle$
$\mathfrak{I}_{174}$: $\langle 0.2235\rangle\big\langle\langle 0.3642, 0.3131\rangle\big\rangle$
$\mathfrak{I}_{162}$: $\langle 0.1786\rangle\big\langle\langle 0.3642, 0.3100\rangle, \langle 0.8641, 1.0000\rangle\big\rangle$
$\mathfrak{I}_{261}$: $\langle 0.2111\rangle\big\langle\langle 0.3642, 0.4220\rangle, \langle 1.0000, 0.7325\rangle, \langle 1.0000, 0.5000\rangle\big\rangle$

Figure 7.10: Four sample images of database D1 with their $\langle\mu_E\rangle\big\langle\langle\mu_C,\mu_H\rangle\big\rangle$.

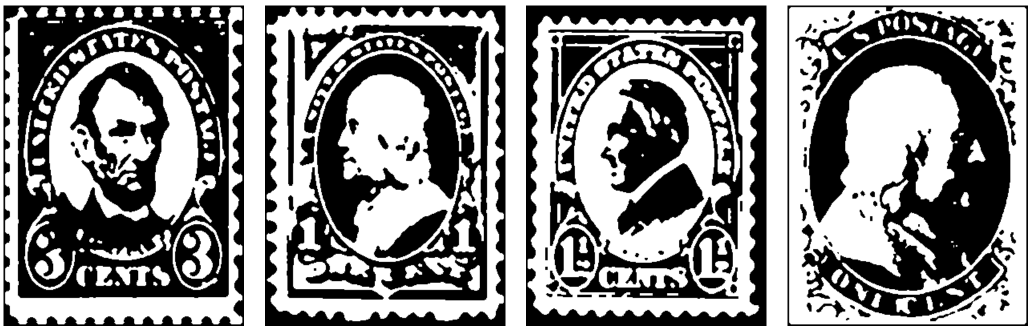
vector $\langle 0.2235\rangle\big\langle\langle 0.3642, 0.3131\rangle\big\rangle$ is a distinct vector with one-to-one correspondence with image $\mathfrak{I}_{174}$, in the feature space of D1. Similar justifications hold for the vectors shown corresponding to the other two images in Fig. 7.10. The four images and their respective vectors, shown in Fig. 7.11, also obey the same one-to-one correspondence in the feature space of D2. It may be mentioned here that the height of the discrimination tree is at most three for all the three databases, the root being at height zero.



$\mathfrak{I}_{69}$ $\mathfrak{I}_{53}$ $\mathfrak{I}_{26}$ $\mathfrak{I}_{63}$

$\mathfrak{I}_{69}: \langle 0.1388 \rangle$
$\mathfrak{I}_{53}: \langle 0.2128 \rangle \big\langle \langle 1.0000, 0.3830 \rangle \big\rangle$
$\mathfrak{I}_{26}: \langle 0.1785 \rangle \big\langle \langle 1.0000, 0.3272 \rangle, \langle 1.0000, 1.0000 \rangle \big\rangle$
$\mathfrak{I}_{63}: \langle 0.5263 \rangle \big\langle \langle 1.0000, 1.0000 \rangle, \langle 0.6781, 0.8000 \rangle, \langle 1.0000, 0.5850 \rangle \big\rangle$

Figure 7.11: Four sample images of database D2 with their $\langle \mu_E \rangle \big\langle \langle \mu_C, \mu_H \rangle \big\rangle$.



$\mathfrak{I}_{aun}$ $\mathfrak{I}_{afk}$ $\mathfrak{I}_{asn}$ $\mathfrak{I}_{aaf}$

$\mathfrak{I}_{aun}: \langle 0.1185 \rangle$
$\mathfrak{I}_{afk}: \langle 0.1760 \rangle \big\langle \langle 0.2344, 0.1313 \rangle \big\rangle$
$\mathfrak{I}_{asn}: \langle 0.4534 \rangle \big\langle \langle 0.2238, 0.2216 \rangle, \langle 0.4328, 0.6822 \rangle \big\rangle$
$\mathfrak{I}_{aaf}: \langle 0.7604 \rangle \big\langle \langle 0.1320, 0.1465 \rangle, \langle 0.6128, 0.7422 \rangle, \langle 0.8129, 1.0000 \rangle \big\rangle$

Figure 7.12: Four sample images of database D3 with their $\langle \mu_E \rangle \big\langle \langle \mu_C, \mu_H \rangle \big\rangle$.

7.4 Conclusion

This work proposes a novel indexing technique for archival binary images by employing a connectivity-based algorithm that extracts certain topological features of each binary image and their membership degrees with respect to the database. The algorithm uses a set of randomized spatial masks to change an image until the feature vector becomes unique for each image in the database. The method automatically adapts to the database size and diversity. A simple iterative *xor*-ing procedure on the bit-plane of an image is used, and this adds speed, elegance and strength to the proposed algorithm.

The algorithm offers a very effective indexing scheme for archival and object-type of images, as observed in our experiments. For other general types of databases consisting of human face images, natural scenes, or those with diverse texture, the performance of this algorithm may not be satisfactory, unless additional features are considered and augmented appropriately. For indexing a gray scale image, proper adaptation of this technique, such as usage of Euler Vector [Bishnu *et al.* (2002)] may be explored in the light of similar randomized spatial masking.

Acknowledgement: The logo database D1 has been received on request, from Prof. Anil K. Jain and Mr. Aditya Vailya of the Michigan State University, USA. The databases D2 and D3 have been collected from the websites
http://documents.cfar.umd.edu/resources/database/UMDlogo.html, and
http://www.mcgees.org/stampalbums.html, respectively.

Chapter 8

Summary and Open Problems

In this thesis, we have developed several new algorithms based on digital geometry for representations and characterizations of digital contours and objects, and exploited their applications to shape description and image indexing. The proposed algorithms and their performance on various problems have been analyzed in the perspective of other existing approaches. The techniques have been tested rigorously on several databases and experimental experience reveals encouraging results.

In Chapter 2, we have shown how the minimum-(maximum-)-area outer (inner) isothetic cover of a digital object can be constructed against a background grid. The algorithm, based on a combinatorial technique, outputs the isothetic cover in terms of a sequence of vertices (grid points). The proof of correctness of the algorithm has also been provided. As the algorithm does not involve any floating point computation, and requires only integer addition and comparison, it terminates very fast. Several open problems may arise in the context of determining an isothetic cover, for example, that of finding the specific object-grid registration such that the complexity of the cover is optimum in terms of the number of vertices, perimeter, or area. Also, it would be a challenging task to reconstruct the digital object if a collection of isothetic covers at different positions and orientations are given. The algorithm for the non-uniform grid can be extended to higher dimensions for possible computation of outer or inner approximation in rough sets.

In Chapter 3, we have described how the orthogonal hull of a digital object can be computed while traversing the object contour orthogonally (following the principle mentioned in Chapter 2). The algorithm uses a similar combinatorial technique based on the chain code of the isothetic cover, and modifies the boundary of the outer isothetic cover to produce the orthogonal hull. This is a single pass algorithm with worst-case time complexity being linear in the length of the contour of the object. As a future work, it may be useful in devising a procedure to represent the regions that give rise to non-convexity in terms of convex deficiency tree for shape analysis. Also, the usage of orthogonal hull in digital tomography may be explored.

Chapter 4 elaborates three applications of isothetic covers with different flavors. The multigrid shape code exploits the fact that the accuracy of the object description, captured by the isothetic cover, can be controlled by varying the grid size. An efficient retrieval scheme has been designed and implemented to demonstrate the power and versatility of such shape codes. The shape complexity measure based on isothetic covers provides useful information about the complexity of a digital object without using any Euclidean measure. A few open problems emerge in this context, for example, enumeration of distinct isothetic polygons that can be drawn on a given grid and formulating their generation procedure. In this chapter, we have also shown how a handwritten character prototype can be ranked using the isothetic chord lengths. This has potential applications to automatic handwritten character recognition. The document image analysis is another potential area for exploring applications, where isothetic covers can be used to segment document images. Some preliminary results in this regard have been shown in Chapter 2.

In Chapter 5, we have presented a novel concept of approximating a digital curve (DC) by a cellular envelope. The salient features of the algorithm are (i) non-dependence on thickness variation of the input DC, (ii) use of a combinatorial approach for constructing the optimum cellular envelope for the given DC, (iii) use of straightness properties inherited from digital geometry, and (iv) implementation without requiring any floating point operation, which collectively make it robust, fast, and efficient.

In Chapter 6, we have shown how a set of digital curves, after being decomposed into a set of shape preserving digital straight edges, can be represented by an appropriate pointillist ensemble. The size of the ensemble can be controlled by the pointillist factor. The proposed algorithm for finding the ensemble imitates the human visual mechanism, and the reconstruction method using the nearest neighbor rule, reproduces the digital curve that almost preserves its original shape, provided the polygonal decomposition of the original curve is reasonably good. Further possibilities involving pointillist ensembles include (i) shape analysis of digital objects using their pointillist ensembles, (ii) color image representation and subsequent analysis using ensembles in the color spectrum, and (iii) exploring other image processing and computer vision related applications.

Chapter 7 describes a novel indexing technique for archival binary images by employing a greedy algorithm defined over certain connectivity features of binary images and their fuzzy membership degrees to the database. An efficient randomization technique is used in the algorithm to expedite the procedure and to make it adaptive to the database size and diversity. The iterative *xor*-ing procedure has an inherent property of capturing the characteristic topological features of an image within a few iterations, thereby adding elegance and strength to the algorithm.

The chapters described in the thesis, present various algorithms targeting to derive the geometric characteristics of a digital object for potential applications. All the algorithms reported in this thesis involve only integer-domain arithmetic operations, (except the computation of fuzzy membership value required in Chapter 7), and hence, are very efficient computationally. We have also demonstrated the usage of these characteristics and representations of digital objects in several applications. These representations can further be extended to several diverse areas, some of them being digital tomography, rough sets, and document image analysis.

Bibliography[*]

AHMAD, I. AND GROSKY, W.I. (1997). Spatial similarity based retrievals and image indexing by hierarchical decomposition. In *Proc. Intl. Database Engg. and Applns. Symposium (IDEAS '97)*, 269–278.

AKEN, J.R.V. AND NOVAK, M. (1985). Curve-drawing algorithms for raster display. *ACM Trans. Graphics*, **4**, 147–169.

AKINDELE, O.T. AND BELAID, A. (1993). Page segmentation by segment tracing. In *Proc. Intl. Conf. Document Analysis and Recognition (ICDAR),* IEEE CS Press, USA, 341–344.

ALMÁADEED, S., ELLIMAN, D. AND HIGGINS, C.A. (2002). A database for Arabic handwritten text recognition research. In *Proc. 8th Intl. Workshop Frontiers in Handwriting Recognition*, 485–489.

ALTHAUS, E. AND MEHLHORN, K. (2002). Traveling salesman-based curve reconstruction in polynomial time. *SIAM J. Comput.*, **31**, 27–66.

ALTHAUS, E., MEHLHORN, K., NÄHER, S. AND SCHIRRA, S. (2000). Experiments on curve reconstruction. In *Proc. 2nd Workshop Algorithm Eng. Exper.*, 103–114.

AMENTA, N., BERN, M. AND EPPSTEIN, D. (1998). The crust and the β-skeleton: Combinatorial curve reconstruction. *Graphical Models and Image Processing*, **60**, 125–135.

ANDERSON, I.M. AND BEZDEK, J.C. (1984). Curvature and tangential deflection of discrete arcs: A theory based on the commutator of scatter matrix pairs and its application to vertex detection in planar shape data. *IEEE Trans. PAMI*, **6**, 27–40.

ANSI (1986). *Fingerprint Identification — Data Format for Information Interchange.* American National Standards Institute, New York.

ANTOINE, J.P., BARACHE, D., CESAR, R.M.J. AND DA F. COSTA, L. (1996). Multiscale shape analysis using the continuous wavelet transform. In *Proc. Intl. Conf. Image Processing (ICIP),* IEEE CS Press, USA, 291–294.

[*]**Reference Style:** AUTHORS (year). Title. *Journal/Proceedings name*, volume, pages.

Antonacopoulos, A. and Meng, H. (2002). A ground-truthing tool for layout analysis performance evaluation. In *Proc. Intl. Workshop Document Analysis Systems*, vol. 2423 of *LNCS*, 236–244.

Asano, T. and Kawamura, Y. (2000). Algorithmic considerations on the computational complexities of digital line extraction problem. *Systems and Computers in Japan*, **31**, 29–37.

Asano, T., Kawamura, Y., Klette, R. and Obokkata, K. (2003). Digital curve approximation with length evaluation. *IEICE Trans. Fundamentals of Electronics, Communication and Computer Sciences*, **E86-A**, 987–994.

Attali, D. (1997). α-regular shape reconstruction from unorganized points. In *Proc. 13th Ann. Sympos. Comput. Geom.*, ACM Press, 248–253.

Attneave, F. (1954). Some informational aspects of visual perception. *Psychological Review*, **61**, 183–193.

Audet, C., Hansen, P. and Messine, F. (2007). Extremal problems for convex polygons. *J. of Global Optimization*, **38**, 163–179.

Avis, D. and Bremner, D. (1995). How good are convex hull algorithms? In *Proc. Annual Symposium on Computational Geometry*, ACM Press, 20–28.

Avis, D. and Horton, J. (1985). Remarks on the sphere of influence graph. In *Proc. Conf. Discr. Geom. Convexity*, vol. 440, Ann. New York Accad. Sci., 323–327.

Balazs, P. (2008). On the number of hv-convex discrete sets. In V. Brimkov, R. Barneva and H. Hauptman, eds., *Intl. Workshop on Combinatorial Image Analysis (IWCIA)*, vol. 4958 of *LNCS*, Springer-Verlag, Berlin, Heidelberg, 112–123.

Baloch, S.H. and Krim, H. (2007). Flexible skew-symmetric shape model for shape representation, classification, and sampling. *IEEE Trans. Image Processing*, **16**, 317–328.

Barber, B.B., Dobkin, D.P. and Huhdanpaa, H. (1993). The quickhull algorithm for convex hull. The Geometry Center, University of Minnesota, Technical Report GCG53.

Barutcuoglu, Z. and DeCoro, C. (2006). Hierarchical shape classification using bayesian aggregation. In *Proc. IEEE SMI'06*, 44–48.

Bazen, A.M. and Gerez, S.H. (2003). Fingerprint matching by thin-plate spline modeling of elastic deformations. *Pattern Recognition*, **36**, 1859–1867.

Belongie, S., Malik, J. and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Trans. PAMI*, **24**, 509–522.

Bemporad, A., Filippi, C. and Torrisi, F.D. (2004). Inner and outer approximations of polytopes using boxes. *Computational Geometry – Theory and Applications*, **27**, 151–178.

Berg, M.D., Kreveld, M.V., Overmars, M. and Schwarzkopf, O. (2000). *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin.

Bernardini, F. and Bajaj, C.L. (1997). Sampling and reconstructing manifolds using α-shapes. In *Proc. 9th Canadian Conf. Comput. Geom.*, 193–198.

Bertrand, G., Imiya, A. and Klette, R., eds. (2001). *Digital and Image Geometry: Advanced Lectures*, vol. 2243 of *LNCS*. Springer, Berlin.

Bezdek, J.C. (1993). Fuzzy models - what are they and why? *IEEE Trans. on Fuzzy Systems*, **1**, 1–6.

Bezdek, J.C. and Anderson, I.M. (1985). An application of the c-varieties clustering algorithms to polygonal curve fitting. *IEEE Trans. Sys., Man, and Cybern.*, **15**, 637–641.

Bhattacharya, P. and Rosenfeld, A. (1990). Contour codes of isothetic polygons. *Computer Vision, Graphics, and Image Processing (CVGIP)*, **50**, 353–363.

Bhattacharya, U., Parui, S.K., Sridhar, M. and Kimura, F. (2005). Two-stage recognition of handwritten Bangla alphanumeric characters using neural classifiers. In *Proc. 2nd Indian Intl. Conf. on Artificial Intelligence*, IICAI, 1357–1376.

Bhowmick, P. and Bhattacharya, B.B. (2007). Fast polygonal approximation of digital curves using relaxed straightness properties. *IEEE Trans. PAMI*, **29**, 1590–1602.

Bhowmick, P. and Bhattacharya, B.B. (2008). Removal of digital aberrations in fingerprint ridgelines using B-splines. *Pattern Recognition* (in press).

Bhowmick, P., Bishnu, A., Bhattacharya, B.B., Kundu, M.K., Murthy, C.A. and Acharya, T. (2005a). Determination of minutiae scores for fingerprint image applications. *Intl. J. Image and Graphics*, **5**, 1–35.

Bhowmick, P., Biswas, A. and Bhattacharya, B.B. (2005b). Isothetic polygons of a 2D object on generalized grid. In *Proc. 1st Intl. Conf. Pattern Recognition and Machine Intelligence (PReMI)*, vol. 3776 of *LNCS*, Springer, Berlin, 407–412.

Bhowmick, P., Biswas, A. and Bhattacharya, B.B. (2006). PACE: Polygonal Approximation of thick digital curves using Cellular Envelope. In *Proc. 5th Indian Conf. Computer Vision, Graphics and Image Processing (ICVGIP)*, vol. 4338 of *LNCS*, Springer, Berlin, 299–310.

Bhowmick, P., Biswas, A. and Bhattacharya, B.B. (2007a). DRILL: Detection and Representation of Isothetic Loosely connected components without Labeling. In *Proc. 6th Intl. Conf. Advances in Pattern Recognition (ICAPR)*, World Scientific, Singapore, 343–348.

Bhowmick, P., Biswas, A. and Bhattacharya, B.B. (2007b). Ranking of optical character prototypes using cellular lengths. In *Proc. Intl. Conf. Computing: Theory and Applications (ICCTA)*, IEEE CS Press, USA, 422–426.

Bishnu, A., Bhattacharya, B., Kundu, M., Murthy, C. and Acharya, T. (2001). A pipeline architecture for computing the Euler number of a binary image. *Journal of Systems Architecture*, **51**, 470–487.

Bishnu, A., Bhowmick, P., Dey, S., Bhattacharya, B.B., Kundu, M.K., Murthy, C.A. and Acharya, T. (2002). Combinatorial classification of pixels for ridge extraction in a gray-scale fingerprint image. In *Proc. Indian Conf. Computer Vision, Graphics and Image Processing (ICVGIP)*, Allied Publishers Pvt. Ltd., New Delhi, 451–456.

Bishnu, A., Bhattacharya, B., Kundu, M., Murthy, C. and Acharya, T. (2005). Euler vector for search and retrieval of gray-tone images. *IEEE Transactions on Systems, Man, and Cybernetics*, **35**, 801–811.

Bishnu, A., Das, S., Nandy, S.C. and Bhattacharya, B.B. (2006). Simple algorithms for partial point set pattern matching under rigid motion. *Pattern Recognition*, **39**, 1662–1671.

Biswas, A., Bhowmick, P. and Bhattacharya, B.B. (2004). **CONFERM**: **Con**nectivity **Fe**atures with **R**andomized **M**asks and their applications to image indexing. In *Proc. Indian Conf. Computer Vision, Graphics and Image Processing (ICVGIP)*, Allied Publishers Pvt. Ltd., New Delhi, 556–562.

Biswas, A., Bhowmick, P. and Bhattacharya, B.B. (2005a). **MuSC**: **Mu**ltigrid **S**hape **C**odes and their applications to image retrieval. In *Proc. Intl. Conf. Computational Intelligence and Security*, vol. 3801 of *LNCS*, Springer, Berlin, 1057–1063.

Biswas, A., Bhowmick, P. and Bhattacharya, B.B. (2005b). **TIPS**: On finding a **T**ight **I**sothetic **P**olygonal **S**hape covering a 2D object. In *Proc. 14th Scandinavian Conf. Image Analysis (SCIA)*, vol. 3540 of *LNCS*, Springer, Berlin, 930–939.

Biswas, A., Bhowmick, P. and Bhattacharya, B.B. (2007a). Characterization of isothetic polygons for image indexing and retrieval. In *Proc. Intl. Conf. Computing: Theory and Applications (ICCTA)*, IEEE CS Press, USA, 590–594.

Biswas, A., Bhowmick, P. and Bhattacharya, B.B. (2007b). SCOPE: Shape Complexity of Objects using isothetic Polygonal Envelope. In *Proc. 6th Intl. Conf. Advances in Pattern Recognition (ICAPR)*, World Scientific, Singapore, 356–360.

Biswas, A., Bhowmick, P. and Bhattacharya, B.B. (2008). Multiresolution shape codes with applications to image retrieval. *Electronic Letters on Computer Vision and Image Analysis (ELCVIA)*, (accepted).

Blum, H. (1967). A transformation for extracting new descriptors of shape. In W.W. Dunn, ed., *Models for the Perception of Speech and Visual Form*, MIT Press, Cambridge, 362–380.

Bookstein, F. (1991). *Morphometric Tools for Landmark Data: Geometry and Biology*. Cambridge Univ. Press.

Bousquet-Mélou, M. (1996). A method for the enumeration of various classes of column-convex polygons. *Discrete Mathematics*, **154**, 1–25.

Bousquet-Mélou, M. and Fédou, J. (1995). The generating function of convex polyominoes: The resolution of a q-differential system. *Discrete Mathematics*, **137**, 53–75.

Boxer, L. (1993). Computing deviations from convexity in polygons. *Pattern Recognition Letters*, **14**, 163–167.

Brimkov, V.E. (2009). Digitization scheme that assures faithful reconstruction of plane figures. *Pattern Recognition*, **42**, 1637–1649.

Brimkov, V.E. and Barneva, R.P. (2004). Connectivity of discrete planes. *Theor. Comput. Sci.*, **319**, 203–227.

Brimkov, V.E. and Barneva, R.P. (2005). Plane digitization and related combinatorial problems. *Discrete Appl. Math.*, **147**, 169–186.

Brimkov, V.E. and Klette, R. (2008). Border and surface tracing - theoretical foundations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **30**, 577–590.

Brimkov, V.E., Coeurjolly, D. and Klette, R. (2007). Digital planarity-a review. *Discrete Appl. Math.*, **155**, 468–495.

Brinkhoff, T., Kriegel, H.P. and Schneider, R. (1995). Measuring the complexity of polygonal objects. In *Proc. Third ACM International Workshop on Advances in Geographical Information Systems,* ACM Press, 109–117.

Buvaneswari, A. and Naidu, P.S. (1998). Estimation of shape of binary polygonal object from scattered field. *IEEE Trans. Image Processing*, **7**, 253–257.

CCITT (1994). Facsimile coding schemes and coding functions for group 4 facsimile apparatus. CCITT Recommendation T.6.

Ceguerra, A. and Koprinska, I. (2002). Integrating local and global features in automatic fingerprint verification. In *Proc. 16th Intl. Conf. Pattern Recognition (ICPR),* IEEE CS Press, USA, 347–350.

Cesar, R.M. and Costa, L. (1997). Application and assessment of multiscale bending energy for morphometric characterization of neural cells. *Review of Scientific Instruments*, **68**, 2177–2186.

Chang, C. and Lee, S. (1991). Retrieval of similar pictures on pictorial databases. *Pattern Recognition*, **24(7)**, 675–680.

Chaudhuri, B.B. and Rosenfeld, A. (1998). On the computation of digital convex hull and circular hull of a digital region. *Pattern Recognition*, **31**, 2007–2016.

Chazelle, B. (1993). An optimal convex hull algorithm in any fixed dimension. *Discrete Comput. Geom.*, **10**, 377–409.

Chazelle, B. and Incerpi, J. (1984). Triangulation and shape complexity. *ACM Transactions on Graphics*, **3**, 135–152.

Chen, M.H. and Yan, P.F. (1988). A fast algorithm to calculate Euler number for binary image. *Pattern Recognition Letters*, **8 (5)**, 295–297.

Chen, T.C. and Chung, K.L. (2001). A new randomized algorithm for detecting lines. *Real Time Imaging*, **7**, 473–481.

Chen, Y. and Sundaram, H. (2005). Estimating the complexity of 2D shapes. In *Proc. Multimedia Signal Processing Workshop*, Shanghai, China, 14–17.

Chesnaud, C., Réfrégier, P. and Boulet, V. (1999). Statistical region snake-based segmentation adapted to different physical noise models. *IEEE Trans. PAMI*, **21**, 1145–1157.

Climer, S. and Bhatia, S.K. (2003). Local lines: A linear time line detector. *Pattern Recognition Letters*, **24**, 2291–2300.

Cormen, T.H., Leiserson, C.E. and Rivest, R.L. (2000). *Introduction to Algorithms*. Prentice Hall of India, New-Delhi.

Costa, L.d.F. and R. M. Cesar, J. (2001). *Shape Analysis and Classification*. CRC Press.

de Figueiredo, L.H. and Gomes, J. (1995). Computational morphology of curves. *Visual Computer*, **11**, 105–112.

Dey, T.K. (2007). *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge University Press.

Dey, T.K. and Kumar, P. (1999). A simple provable curve reconstruction algorithm. In *Proc. 10th Ann. ACM-SIAM Sympos. Discr. Algorithms*, 893–894.

Dey, T.K., Mehlhorn, K. and Ramos, E.A. (1999). Curve reconstruction: Connecting dots with good reason. In *Symposium on Computational Geometry*, ACM Press, 197–206.

Dey, T.K., Mehlhorn, K. and Ramos, E.A. (2000). Curve reconstruction: Connecting dots with good reason. *Comput. Geom. Theory and Appl.*, **15**, 229–244.

Díaz-Baqez, J.M. and Mesa, J.A. (2001). Fitting rectilinear polygonal curves to a set of points in the plane. *European Journal of Operational Research*, **130**, 214–222.

Ducksbury, P.G. and Varga, M.J. (1997). Region based image content descriptors and representation. In *Proc. 6th Intl. Conf. Image Processing and Its Applications*, vol. 2, 561–565.

Dunham, J.G. (1986). Optimum uniform piecewise linear approximation of planar curves. *IEEE Trans. PAMI*, **8**, 67–75.

Dwyer, R.A. (1995). The expected size of the sphere-of-influence graph. *Computational Geometry*, **5**, 155–164.

Edelsbrunner, H. (1992). Weighted alpha shapes. Tech. Rep., University of Illinois at Urbana Champaign, IL, USA.

Edelsbrunner, H. (1998). Shape reconstruction with Delaunay complex. In *Proc. LATIN'98: Theoretical Informatics*, vol. 1380 of *LNCS*, 119–132.

Edelsbrunner, H., Kirkpatrick, D.G. and Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Trans. Information Theory*, **29**, 551–559.

Faloutsos, C., Barber, R., Flickner, M., Hafner, J., Niblack, W., Petkovic, D. and Equitz, W. (1994). Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, **3 (3/4)**, 231–262.

Fam, A. and Sklansky, J. (1977). Cellularly straight images and the Hausdorff metric. In *Proc. Conf. on Pattern Recognition and Image Processing*, 242–247.

Feldman, J. (2000). Bias toward regular form in mental shape spaces. *J. Experimental Psychology: Human Perception*, **26**, 1–44.

Fischler, M.A. and Wolf, H.C. (1994). Locating perceptually salient points on planar curves. *IEEE Trans. PAMI*, **16**, 113–129.

Freeman, H. (1961a). On the encoding of arbitrary geometric configurations. *IRE Trans. Electronic Computers*, **EC-10**, 260–268.

Freeman, H. (1961b). Techniques for the digital computer analysis of chain-encoded arbitrary plane curves. In *Proc. National Electronics Conf.*, vol. 17, 421–432.

Freeman, H. and Davis, L.S. (1977). A corner finding algorithm for chain-coded curves. *IEEE Trans. Computers*, **26**, 297–303.

FVC 2000 (2000). Fingerprint Verification Competition, 2000. http://bias.csr.unibo.it/fvc2000/download.asp.

Gage, J.T. (1987). The Technique of Seurat: A Reappraisal. *Art Bulletin*, **69**.

Garris, M.D. and Wilkinson, R.A. (1992). In *NIST Special Database 3: Binary Images for Handwritten Segmented Characters (HWSC)*, John Wiley and Sons, New York.

Gatos, B. and Mantzaris, S.L. (2000). A novel recursive algorithm for area location using isothetic polygons. In *Proc. Intl. Conf. on Pattern Recognition (ICPR)*, vol. 3, IEEE CS Press, USA, 492–495.

Gatos, B., Mantzaris, S.L., Chandrinos, K.V., Tsigris, A. and Perantonis, S.J. (1999). Integrated algorithms for newspaper page decomposition and article tracking. In *Proc. Intl. Conference on Document Analysis and Recognition (ICDAR)*, 559–562.

Gatrell, L. (1989). CAD-based grasp synthesis utilizing polygons, edges and vertices. In *Proc. IEEE Intl. Conf. Robotics and Automation*, 184–189.

Gdalyahu, Y. and Weinshall, D. (1999). Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes. *IEEE Trans. PAMI*, **21**, 1312–1328.

Geer, P. and McLaughlin, H.W. (2003). Cellular lines: An introduction. *Discrete Mathematics and Theoretical Computer Science*, 167–178.

Geist, J., Wilkinson, R.A., Janet, S., Grother, P.J., Hammond, B., Larsen, N.W., Klear, R.M., Burges, C.J.C., Creecy, R., Hull, J.J., Vogl, T.P. and Wilson, C.L. (1994). The second census optical character recognition systems conf. Natl. Inst. of Standards and Technology, Maryland, Technical Report.

Gerken, P. (1994). Object-based analysis-synthesis coding of image sequences at very low bit rates. *IEEE Trans. Circuits Syst. Video Technol.*, **4**, 228–235.

Gesù, V., Bosco, G.L., Millonzi, F. and Valenti, C. (2008). Discrete tomography reconstruction through a new memetic algorithm. In M. Giacobini, A. Brabazon,

S. Cagnoni, G.D. Caro, R. Drechsler, A. Ekrt, A. Esparcia-Alczar, M. Farooq, A. Fink, J. McCormack, M. O'Neill, J. Romero, F. Rothlauf, G. Squillero, S. Uyar and S. Yang, eds., *EvoWorkshops*, vol. 4974 of *Lecture Notes in Computer Science*, Springer, 347–352.

Gesù, V.D. (1999). Soft computing and image analysis. *Soft Computing for Image Processing*, 247–259.

Gesù, V.D. and Bosco, G.L. (2005). A genetic integrated fuzzy classifier. *Pattern Recognition Letters*, **26**, 411–420.

Gesù, V.D. and Maccarone, M. (1998). Solving image analysis problems with fuzzy-sets. In *Proc. International Conference SOFT-COMP-98*, 156–169.

Gesù, V.D. and Roy, S. (2000). Fuzzy measures for image distance. *Advances in Fuzzy Systems and Intelligent Technoligies*, 156–164.

Gesù, V.D. and Roy, S. (2002). Pictorial indexes and soft image distances. In *AFSS '02: Proceedings of the 2002 AFSS International Conference on Fuzzy Systems. Calcutta*, Springer-Verlag, London, UK, 360–366.

Gesù, V.D. and Valenti, C. (2004). The stability problem and noisy projections in discrete tomography. *J. Vis. Lang. Comput.*, **15**, 361–371.

Giesen, J. (2000). Curve reconstruction, the traveling salesman problem and menger's theorem on length. *Discr. Comput. Geom.*, **24**, 577–603.

Gold, C. and Snoeyink, J. (2001). Crust and anti-crust: A one-step boundary and skeleton extraction algorithm. *Algorithmica*, **30**, 144–163.

Gonzalez, R.C. and Woods, R.E. (1993). *Digital Image Processing*. Addison-Wesley, California, USA.

Goshtasby, A. (1985). Description and discrimination of planar shapes using shape matrices. *IEEE Trans. PAMI*, **7**, 738–743.

Graham, R. (1972). An efficient algorithm for determining the convex hull of a finite point set. *Info. Proc. Letters*, **1**, 132–133.

Gu, Y.H. and Tjahjadi, T. (1999). Corner-based feature extraction for object retrieval. In *Proc. Intl. Conf. Image Processing (ICIP),* IEEE CS Press, USA, 119–123.

Guru, D.S., Shekar, B.H. and Nagabhushan, P. (2004). A simple and robust line detection algorithm based on small eigenvalue analysis. *Pattern Recognition Letters*, **25**, 1–13.

Guyon, I., Schomaker, L. and Plamondon, R. (1994). UNIPEN project of on-line data exchange and recognizer benchmarks. In *Proc. 12th Intl. Conf. Pattern Recognition (ICPR),* IEEE CS Press, USA, 29–33.

Held, A., Abe, K. and Arcelli, C. (1994). Towards a hierarchical contour description via dominant point detection. *IEEE Trans. Sys., Man, and Cybern.*, **24**, 942–949.

Herman, G. (1980). *Image Reconstruction from Projections: The Fundamentals of Computerized Tomography*. Academic Press, New York.

Herman, G. and Kuba, A. (1999). *Discrete Tomography: Foundations, Algorithms and Applications*. Birkhuser, Boston.

Hirata, K. and Kato, T. (1992). Query by visual example - content based image retrieval. In *Proc. 3rd International Conference on Extending Database Technology: Advances in Database Technology*, 56–71.

Hotter, M. (1990). Object-oriented analysis-synthesis coding based on two-dimensional objects. *Signal Processing: Image Commun.*, **2**, 409–428.

Hsu, R.L. and Jain, A.K. (2003). Generating discriminating cartoon faces using interacting snakes. *IEEE Trans. PAMI*, **25**, 1388–1398.

Hu, J. and Yan, H. (1997). Polygonal approximation of digital curves based on the principles of perceptual organization. *Pattern Recognition*, **30**, 701–718.

Hu, M.K. (1962). Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, **8**, 179–187.

Huang, F.Z. and Su, J. (2002). Face contour detection using geometric active contours. In *Proc. 4th World Congress on Intelligent Control and Automation*, IEEE CS Press, USA, 2090–2093.

Hull, J.J. (1994). A database for handwritten text recognition research. *IEEE Trans. PAMI*, **16**, 550–554.

Huttenlocaher, D., Klanderman, G. and Rucklidge, W. (1993). Comapring images using the Hausdorff distance. *IEEE Trans. PAMI*, **15**, 850–863.

Hyde, S.T., Andersson, S., Blum, Z., Lidin, S., Larsson, K., Landh, T. and Ninham, B.W. (1997). *The Language of Shape*. Elsevier.

Idris, F. and Panchanathan, S. (1995). Image indexing using vector quantization. In *Proc. Storage and Retrieval for Image and Video Databases*, 373–380.

Imai, H. and Iri, M. (1986). Computational geometric methods for polygonal approximations of a curve. *Computer Vision, Graphics, and Image Processing*, **36**, 31–41.

Irvin, R. and McKeown, D. (1989). Methods for exploiting the relationship between buildings and their shadows in aerial imagery. *IEEE Trans. Systems, Man, and Cybernetics*, **19**, 1564–1575.

ISI (2002). Handwritten Bangla characters. http://www.isical.ac.in/∼ujjwal/download/-database.html.

ISO (1992). Coded representation of picture and audio information progressive bi-level image compression. ISO Draft 11544.

ISO (1999). Information technology coded representation of picture and audio information lossy/lossless coding of bi-level images. ISO FCD. 14492.

Jacobs, C., Finkelstein, A. and Salesin, D. (1995). Fast multiresolution image querying. In *Proc. SIGGRAPH*, 277–286.

Jain, A.K. and Yu, B. (1998). Document representation and its application to page decomposition. *IEEE Trans. PAMI*, **20**, 294–308.

Jain, A.K., Hong, L. and Bolle, R. (1997). On-line fingerprint verification. *IEEE Trans. PAMI*, **19**, 302–313.

Jain, A.K., Ross, A. and Prabhakar, S. (2001). Fingerprint matching using minutiae and texture features. In *Proc. Intl. Conf. Image Processing (ICIP),* IEEE CS Press, USA, 282–285.

Jaromczyk, J.W. and Kowaluk, M. (1987). A note on relative neighborhood graphs. In *Proc. 6th ACM Sympos. Comput. Geom.*, ACM Press, 233–241.

Jarvis, R.A. (1973). On the identification of the convex hull of a finite set of points in the plane. *Info. Proc. Letters*, **2**, 18–21.

Kamon, Y., Flash, T. and Edelman, S. (1995). Learning to grasp using visual information. In *Proc. IEEE Intl. Conf. Robotics and Automation*, 2470–2476.

Kaneko, T. and Okudaira, M. (1985). Encoding of arbitrary curves based on the chain code representation. *IEEE Trans. Commun.*, **33 (7)**, 697–707.

Kang, K.W. and Kim, J.H. (2004). Utilization of hierarchical, stochastic relationship modeling for Hangul character recognition. *IEEE Trans. PAMI*, **26**, 1185–1196.

Karlsson, R. and Overmars, M. (1988). Scanline algorithms on a grid. *BIT Numerical Mathematics*, **28**, 227–241.

Kartikeyan, B. and Sarkar, A. (1989). Shape description by time series. *IEEE Trans. PAMI*, **11**, 977–984.

Kass, W., Witkin, A. and Terzopoulos, D. (1998). Snakes: Active contour models. *Intl. J. Computer Vision*, **1**, 321–331.

Katsaggelos, A.K., Kondi, L., Meier, F.W., Ostermann, J. and Schuster, G.M. (1998). MPEG-4 and rate distortion based shape coding techniques. In *Proc. IEEE*, 1126–1154.

Kaup, A. and Heuer, J. (2000). Polygonal shape descriptors – an efficient solution for image retrieval and object localization. In *Proc. 34th Asilomar Conference on Signals, Systems, and Computers*, vol. 1, IEEE CS Press, USA, 59–64.

Kim, C.E. (1982). On cellular straight line segments. *Computer Graphics Image Processing*, **18**, 369–391.

Kim, D.H., Hwang, Y.S., Park, S.T., Kim, E.J., Pack, S.H. and Bang, S.Y. (1993). Handwritten korean character image database PE92. In *Proc. Second Intl. Conf. Document Analysis and Recognition (ICDAR)*, 470–473.

Kim, D.I. and Lee, S.W. (1998). Automatic evaluation of handwriting qualities of handwritten Hangul image database, KU-1. In *Proc. Sixth Intl Workshop Frontiers in Handwriting Recognition*, 455–464.

Kirkpatrick, D.G. and Radke, J.D. (1985). A framework for computational morphology. In G.T. Toussaint, ed., *Computational Geometry*, Elsevier, 217–248.

Kirkpatrick, D.G. and Seidel, R. (1986). The ultimate planar convex hull algorithm? *SIAM Jour. Comput.*, **15**, 287–299.

Klette, R. (1982). *Hüllen für endliche Punktmengen.* In R. Klette and J. Mecke, eds., *Proc. Geometric Problems of Image Processing*, Jena University, Germany, 74–83.

Klette, R. (2000). Cell complexes through time. In *Proc. Vision Geometry*, vol. IX of *SPIE 4117*, 134–145.

Klette, R. (2001a). Digital geometry – The birth of a new discipline. In L.S. Davis, ed., *Foundations of Image Understanding*, Kluwer, Boston, Massachusetts, 33–71.

Klette, R. (2001b). Multigrid convergence of geometric features. In G. Bertrand, A. Imiya and R. Klette, eds., *Digital and Image Geometry: Advanced Lectures*, vol. 2243 of *LNCS*, Springer, Berlin, Germany, 314–333.

Klette, R. and Rosenfeld, A. (2004a). *Digital Geometry: Geometric Methods for Digital Picture Analysis*. Morgan Kaufmann Series in Computer Graphics and Geometric Modeling, Morgan Kaufmann, San Francisco.

Klette, R. and Rosenfeld, A. (2004b). Digital straightness: A review. *Discrete Applied Mathematics*, **139**, 197–230.

Kong, T.Y. (2001). Digital topology. In L.S. Davis, ed., *Foundations of Image Understanding*, Kluwer, Boston, Massachusetts, 33–71.

Kong, T.Y. and Rosenfeld, A., eds. (1996). *Topological Algorithms for Digital Image Processing*. Elsevier, Amsterdam, The Netherlands.

Lachaud, J.O., Vialard, A. and de Vieilleville, F. (2007). Fast, accurate and convergent tangent estimation on digital contours. *Image and Vision Computing*, **25**, 1572–1587.

Latecki, L.J. and Rosenfeld, A. (2002). Recovering polygons from noisy data. *Computer Vision and Image Understanding*, **86**, 1–20.

Latecki, L.J., Eckhardt, U. and Rosenfeld, A. (1995). Well-composed sets. *Computer Vision and Image Understanding*, **8**, 61–70.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient based learning applied to document recognition. *Proceedings of the IEEE*, **86**, 2278–2324.

Lee, S., Cho, D., Cho, Y., Son, S., Jang, E., Shin, J. and Seo, Y. (1999). Binary shape coding using baseline-based method. *IEEE Trans. Circuits Syst. Video Technol.*, **9**, 44–58.

Lengyel, J., Reichert, M., Donald, B.R. and Greenberg, D.P. (1990). Real-time robot motion planning using rasterizing computer. *Computer Graphics, ACM*, **24**, 327–335.

Lewis, D. (1992). An evaluation of phrasal and clustered representations on a text categorization task. In *Proc. 15th ACM SIGIR Conf. Research and Development in Information Retrieval*, 37–50.

Li, B. and Holstein, H. (2003). Using k-d trees for robust 3D point pattern matching. In *Proc. 4th Intl. Conf. 3-D Digital Imaging and Modeling*, 95–102.

Liang, K.C. and Kuo, C.C.J. (1997). Wavelet-compressed image retrieval using successive approximation quantization (SAQ) features. In *Proc. SPIE Voice, Video, and Data Communications*, 206–217.

Ling, H. and Jacobs, D.W. (2007). Shape classification using the inner-distance. *IEEE Trans. PAMI*, **29**, 286–299.

Liow, Y.T. and Pavlidis, T. (1990). Use of shadows for extracting buildings in aerial images. *Computer Vision, Graphics, and Image Processing CVGIP*, **49**, 242–277.

Liu, M., He, Y., Hu, H. and Yu, D. (2004). Dimension reduction based on rough set in image mining. In *Proc. Intl. Conf. on Computer and Information Technology (CIT'04)*, 39–44.

Lorentz, G.G. (1949). A problem of plane measure. *American Journal of Mathematics*, 417–426.

Ma, W. and Manjunath, B. (1994). Pictorial queries: Combining feature extraction with database search. Tech. Rep., Santa Barbara, USA.

Maltoni, D., Maio, D., Jain, A.K. and Prabhakar, S. (2003). *Handbook of Fingerprint Recognition*. Springer-Verlag, New York.

Maragos, P.A. and Schafer, R.W. (1986). Morphological skeleton representation and coding of binary images. *IEEE Trans. Acoust., Speech, Signal Processing*, **34**, 1228–1244.

Märgner, V., Pechwitz, M. and ElAbed, H. (2005). ICDAR 2005 Arabic handwriting recognition competition. In *Proc. Intl. Conf. Document Analysis and Recognition (ICDAR)*, 70–74.

Martin, D., Fowlkes, C., Tal, D. and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Intl. Conf. Computer Vision*, vol. 2, 416–423.

Matas, J. and Zimmermann, K. (2005). Unconstrained license plate and text localization and recognition. In *Proc. IEEE Conference on Intelligent Transportation Systems*, 572–577.

Matula, D.W. and Sokal, R.R. (1984). Properties of Gabriel graphs relevant to geographical variation research and the culturing of points in the plane. *Geographical Analysis*, **12**, 205–222.

Mokhtarian, F. and Mackworth, A.K. (1992). A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Trans. PAMI*, **14 (8)**, 789–805.

Mokhtarian, F. and Mohanna, F. (2002). Content-based video database retrieval through robust corner tracking. In *Proc. IEEE Workshop on Multimedia Signal Processing*, 224–228.

Mokhtarian, F., Abbasi, S. and Kittler, J. (1997). Efficient and robust retrieval by shape content through curvature scale space. *Image Databases and Multi-Media Search*, 51–58.

Morales, A., Sanz, P.J. and Á. P. del Pobil (2002). Vision-based computation of three-finger grasps on unknown planar objects. In *Proc. IEEE Intl. Conf. on Intelligent Robots and Systems*, 1711–1716.

Mukhopadhyay, A. and Rao, S.V. (1998). Output-sensitive algorithm for computing the β-skeleton. In *Proc. 10th Canadian Conf. Computational Geometry (CCCG)*, 285–289.

MUNIM, H.E.A.E. AND FARAG, A.A. (2007). Curve/surface representation and evolution using vector level sets with application to the shape-based segmentation problem. *IEEE Trans. PAMI*, **29**, 945–958.

NAKAMURA, A. AND AIZAWA, K. (1984). Digital circles. *Computer Vision, Graphics, and Image Processing*, **26**, 242–255.

NAKAMURA, A. AND ROSENFELD, A. (1997). Digital calculus. *Information Sciences*, **98**, 83–98.

NANDY, S.C. AND BHATTACHARYA, B.B. (2000). Safety zone problem. *Journal of Algorithms*, **37**, 538–569.

NANDY, S.C., MUKHOPADHYAYA, K. AND BHATTACHARYA, B.B. (2008). Recognition of the largest empty ortho-convex polygon in a point set. In *Proc. Canadian Conference on Computational Geometry (CCCG)*, (to appear).

NORONHA, S. AND NEVATIA, R. (2001). Detection and modeling of buildings from multiple aerial images. *IEEE Trans. PAMI*, **23**, 501–518.

O'CONELL, K.J. (1997). Object-adaptive vertex-based shape coding method. *IEEE Trans. Circuits and Systems for Video Technology*, **7(1)**, 251–255.

O'CONNELL, K.J. (1997). Object-adaptive vertex-based shape coding method. *IEEE Trans. Circuits Syst. Video Technol.*, **7**, 251–255.

ODDO, L.A. (1992). Global shape entropy: A mathematically tractable approach to building extraction in aerial imagery,. In *Proc. 20th SPIE AIPR Workshop*, 91–101.

O'ROURKE, J. (1982). Computing the relative neighborhood graph in the L_1 and L_∞ metrics. *Pattern Recognition*, **15**, 189–192.

OVERMARS, M. (1983). *The design of dynamic data structures*. LNCS, Springer-Verlag, Berlin.

PAGE, D.L., KOSCHAN, A., SUKUMAR, S., ABIDI, B. AND ABIDI, M. (2003). Shape analysis algorithm based on information theory. In *Proc. IEEE ICIP'03*, 14–17.

PAL, S.K. AND MITRA, P. (2004a). Case generation using rough sets with fuzzy representation. *IEEE Trans. on Knowledge and Data Engg.*, **16**, 292–300.

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

Pardo, X., Carreira, M., Mosquera, A. and Cabello, D. (2001). A Snake for CT image segmentation integrating region and edge information. *Image and Vision Computing*, **19**, 461–475.

Pavlidis, I., Singh, R. and Papanikolopoulos, N.P. (1997). An on-line handwritten note recognition method using shape metamorphosis. In *Proc. 4th Intl. Conf. Document Analysis and Recognition (ICDAR)*, 914–918.

Pavlidis, T. (1978). A review of algorithms for shape analysis. *Comput. Graph. Image Process.*, **7**, 243–258.

Pavlidis, T. (1980). Algorithms for shape analysis and waveforms. *IEEE Trans. PAMI*, **2**, 301–312.

Pawlak, Z. (1990). Theory of rough sets: A new methodology for knowledge discovery (abstract). In *ICCI*, 11.

Pentland, A., Picard, R. and Sclaroff, S. (1994). Photobook: Tools for content-based manipulation of image databases. In *Proc. SPIE Storage and Retrieval for Image and Video Databases II*, 34–47.

Perez, J.C. and Vidal, E. (1994). Optimum polygonal approximation of digitized curves. *Pattern Recognition Letters*, **15**, 743–750.

Persoon, E. and Fu, K. (1977). Shape discrimination using fourier descriptors. *IEEE Trans. Syst., Man, Cybern.*, **7**, 170–179.

Pitas, I. and Venetsanopoulos, A.N. (1990). Morphological shape decomposition. *IEEE Trans. PAMI*, **12**, 38–45.

Pitty, A.F. (1984). *Geomorphology*. Blackwell.

Plamondon, R. and Srihari, S.N. (2000). On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. PAMI*, **22**, 63–84.

Pogue, B., Mycek, M. and Harper, D. (2000). Image analysis for discrimination of cervical neoplasia. *Journal of Biomedical Optics*, **5 (1)**, 72–82.

Pratt, W. (1978). *Digital Image Processing*. John Wiley and Sons.

Preparata, F.P. and Shamos, M.I. (1985). *Computational Geometry - An Introduction*. Spinger-Verlag, New York.

Prokop, R.J. and Reeves, A.P. (1992). A survey of moment-based techniques for unoccluded object representation and recognition. *CVGIP: Graph. Models Image Process.*, **54**, 438–460.

Roberts, F. (1984). *Applied Combinatorics*. Prentice Hall Inc., USA.

Rosenfeld, A. (1974). Digital straight line segments. *IEEE Trans. Computers*, **23**, 1264–1268.

Rosenfeld, A. (1979). Digital topology. *American Mathematical Monthly*, **86**, 621–630.

Rosenfeld, A. and Kak, A.C. (1982). *Digital Picture Processing, 2nd ed.*. Academic Press, New York.

Rosenfeld, A. and Klette, R. (2001). Digital straightness. *Electronic Notes in Theoretical Computer Sc.*, **46**, http://www.elsevier.nl/locate/entcs/volume46.html.

Rosin, P.L. (1997). Techniques for assessing polygonal approximation of curves. *IEEE Trans. PAMI*, **19**, 659–666.

Rosin, P.L. (1999). Measuring rectangularity. *Machine Vision and Applications*, **11**, 191–196.

Rosin, P.L. (2000). Shape partitioning by convexity. *IEEE Trans. Sys., Man, and Cybern.*, **30**, 202–210.

Rosin, P.L. (2003). Measuring shape: Ellipticity, rectangularity, and triangularity. *Machine Vision and Applications*, **14**, 172–184.

Rosin, P.L. and West, G.A.W. (1995). Non-parametric segmentation of curves into various representations. *IEEE Trans. PAMI*, **17**, 1140–1153.

Sarkar, D. (1993). A simple algorithm for detection of significant vertices for polygonal approximation of chain-coded curves. *Pattern Recognition Letters*, **14**, 959–964.

Schröder, K. and Laurent, P. (1999). Efficient polygon approximations for shape signatures. In *Proc. Intl. Conf. Image Processing (ICIP),* IEEE CS Press, USA, 811–814.

Schuster, G.M. and Katsaggelos, A.K. (1997). *Rate-Distortion Based Video Compression: Optimal Video Frame Compression and Object Boundary Encoding*. Kluwer, Norwell.

Schuster, G.M. and Katsaggelos, A.K. (1998). An optimal polygonal boundary encoding scheme in the rate distortion sense. *IEEE Trans. Circuits and Systems for Video Technology*, **7**, 13–26.

Sclaroff, S. and Pentland, A. (1995). Modal matching for correspondance and recognition. *IEEE Trans. PAMI*, **17**, 545–561.

Serra, J. (1982). *Image Analysis and Mathematical Morphology*. Academic, New York.

Seurat, G.P. (1966). Letter to Maurice Beaubourg (1890). In L. Nochlin, ed., *Impressionism and Post-Impressionism: Sources and Documents (p. 113)*, Englewood Cliffs (NJ).

Sharon, E. and Mumford, D. (2004). 2D-shape analysis using conformal mapping. In *Proc. IEEE CVPR*, vol. 2, 350–357.

Sharvit, D., Chan, J., Tek, H. and Kimia, B. (1998). Symmetry based indexing of image databases. *J. Visual Communication and Image Representation*, **9**, 366–380.

Shuster, G.M. and Katsaggelos, A.K. (1998). An optimal polygonal boundary encoding scheme in the rate distortion sense. *IEEE Trans. Image Processing*, **7(1)**, 13–26.

Siddiqi, K. and Kimia, B.B. (1993). Parts of visual form: Computational aspects. In *Proc. IEEE CVPR*, 75–81.

Sklansky, J. (1972a). Measuring concavity on a rectangular mosaic. *IEEE Trans. Comput.*, **21**, 1355–1364.

Sklansky, J. (1972b). Minimum-perimeter polygons of digitized silhouettes. *IEEE Trans. Comput.*, **21**, 260–268.

Sklansky, J. and Kibler, D.F. (1976). A theory of nonuniformly digitized binary pictures. *IEEE Transactions on Systems, Man, and Cybernetics*, **6**, 637–647.

Sloboda, F. and Zat'ko, B. (1995). On boundary approximation. In *Proc. of the 6th International Conference on Computer Analysis of Images and Patterns (CAIP '95)*, 488–495.

Smith, J. and Chang, S.F. (1996). Visualseek: a fully automated content-based image query system. *ACM Multimedia*, 87–98.

Sonka, M., Hlavac, V. and Boyle, R. (1993). *Image Processing, Analysis, and Machine Vision*. Chapman and Hall.

Srihari, S. (1986). Document image understanding. In *Proc. ACM/IEEE Joint Fall Computer Conference*, 87–96.

Stavrianopoulou, A. and Anastassopoulos, V. (2000). The Euler feature vector. In *Proc. Intl. Conference on Pattern Recognition*, IEEE CS Press, USA, 1022–1024.

Steji, Z., Takama, Y. and Hirota, K. (2003). Genetic algorithms for a family of image similarity models incorporated in the relevance feedback mechanism. *Applied Soft Computing*, **2 (4)**, 306–327.

Stern, H.I. (1989). Polygonal entropy: A convexity measure. *Pattern Recognition Letters*, **10**, 229–235.

Sun, D., Zhou, Z. and Wu, L. (2002). Face boundary extraction by statistical constraint active contour model. In *Proc. IEEE Conf. Sys., Man, and Cybern.*, vol. 6, 3–5.

Sur-Kolay, S. and Bhattacharya, B.B. (1988). Inherent nonslicibility of rectangular duals in VLSI floorplanning. In *Proc. FSTTCS*, vol. 338, LNCS, Springer, Berlin, 88–107.

Swain, M. and Ballard, D. (1991). Color indexing. *International Journal of Computer Vision*, **7 (1)**, 11–32.

Swart, G. (1985). Finding the convex hull facet by facet. *Journal of Algorithms*, 17–48.

Tadrat, J., Boonjing, V. and Pattaraintakorn, P. (2007). A framework for using rough sets and formal concept analysis in case based reasoning. In *Proc. IEEE Intl. Conf. on Information Reuse and Integration*, 227–232.

TANIGAWA, S. AND KATOH, N. (2006). Polygonal curve approximation using grid points with application to a triangular mesh generation with small number of different edge lengths. In *Proc. Intl. Conf. on Algorithmic Aspects in Information and Management, AAIM 2006*, 161–172.

TEH, C.H. AND CHIN, R.T. (1989). On the detection of dominant points on digital curves. *IEEE Trans. PAMI*, **2**, 859–872.

TIAN, G., GLEDHILL, D. AND TAYLOR, D. (2003). Comprehensive interest points based imaging mosaic. *Pattern Recognition Letters*, **24**, 1171–1179.

TOUSSAINT, G. (1991). Efficient triangulation of simple polygons. *Visual Computer*, **7**, 280–295.

TOUSSAINT, G.T. (1988). A graph-theoretical primal sketch. In G.T. Toussaint, ed., *Computational Morphology*, Elsevier.

TSAI, A., WELLS, W.M., WARFIELDA, S.K. AND WILLSKY, A.S. (2005). An EM algorithm for shape classification based on level sets. *Medical Image Analysis*, **9**, 491–502.

VELTKAMP, R.C. (1992). The γ-neighborhood graph. *Comput. Geom. Theory Appl.*, **4**, 227–246.

VELTKAMP, R.C. (1995). Boundaries through scattered points of unknown density. *Graphics Model and Image Proc.*, **57**, 441–452.

VENTURA, J.A. AND CHEN, J.M. (1992). Segmentation of two-dimensional curve contours. *Pattern Recognition*, **25**, 1129–1140.

VERMA, B. AND KULKARNI, S. (2004). A fuzzy-neural approach for interpretation and fusion of colour and texture features for CBIR systems. *Applied Soft Computing*, **5 (1)**, 119–130.

VERTAN, C. AND BOUJEMMA, N. (2000). Embedding fuzzy logic in content based image retrieval. In *Proc. 19th Int'l Meeting North Am. Fuzzy Information Processing Soc.*, 85–89.

WALL, K. AND DANIELSSON, P.E. (1984). A fast sequential method for polygonal approximation of digitized curves. *Computer Vision, Graphics, and Image Processing*, **28**, 220–227.

Wang, H., Schuster, G., Katsaggelos, A. and Pappas, T. (2003). An efficient rate-distortion optimal shape coding approach utilizing a skeleton-based decomposition. *IEEE Trans. Image Processing*, **12**, 1181–1193.

Wang, J., Wiederhold, G., Firschein, O. and Wei, S. (1997). Wavelet-based image indexing techniques with partial sketch retrieval capability. In *Proc. IEEE International forum on Research and Technology Advances in Digital Libraries*, 13–24.

Wang, J.Y. and Su, G.D. (2003). The research of chin contour in fronto-parallel images. In *Proc. International Conference on Machine Learning and Cybernetics*, vol. 5, 2814–2819.

Willats, J. (1997). *Art and Representation: New Principles in the Analysis of Pictures*. Princeton Univ. Press.

Wolf, C., Jolion, J.M., Kropatsch, W. and Bischof, H. (2000). Content based image retrieval using interest points and texture features. In *Proc. 15th Intl. Conf. Pattern Recognition (ICPR),* IEEE CS Press, USA, vol. 4, 234–237.

Wu, J. and Jiang, Z. (2005). On constructing the minimum orthogonal convex polygon for the fault-tolerant routing in 2-D faulty meshes. *IEEE Transactions on Reliability*, **54**, 449–458.

Wu, L.D. (1984). A piecewise linear approximation based on a statistical model. *IEEE Trans. PAMI*, **6**, 41–45.

Wuescher, D.M. and Boyer, K.L. (1991). Robust contour decomposition using a constant curvature criterion. *IEEE Trans. PAMI*, **13**, 41–51.

Xie, Y. and Ji, Q. (2001). Effective line detection with error propagation. In *Proc. Intl. Conf. Image Processing (ICIP),* IEEE CS Press, USA, 181–184.

Yacoub, S., Saxena, V. and Sami, S.N. (2005). Perfectdoc: A ground truthing environment for complex documents. In *Proc. 8th Intl. Conf. Document Analysis and Recognition*, 452–456.

Yin, P.Y. (1998). A new method for polygonal approximation using genetic algorithms. *Pattern Recognition Letters*, **19**, 1017–1026.

Yin, P.Y. (2003). Ant colony search algorithms for optimal polygonal approximation of plane curves. *Pattern Recognition*, **36**, 1783–1797.

Yin, P.Y. (2004). A discrete particle swarm algorithm for optimal polygonal approximation of digital curves. *J. Visual Comm. Image Reprsn.*, **15**, 241–260.

Yip, B. and Klette, R. (2003). Angle counts for isothetic polygons and polyhedra. *Pattern Recognition Letters*, **21**, 1275–1278.

Younes, L. (1999). Optimal matching between shapes via elastic deformations. *Image Vis. Comput.*, **17**, 381–389.

Yu, S. and Thonnat, M. (1992). Using apparent boundary and convex hull for the shape characterization of foramifera images. In *Proc. 11th IAPR Intl. Conf. Image, Speech and Signal Analysis*, 569–572.

Zadeh, L. (1965). Fuzzy sets, information and control. *International Journal of Computer Vision*, **8**, 338–353.

Zahn, C. and Roskies, R. (1972). Fourier descriptors for plane closed curves. *Comput. Graph. Image Process.*, **21**, 269–281.

Zhu, S. and Yuille, A. (1996). Region competition – unifying snakes, region growing, and bayes/mdl for multiband image segmentation. *IEEE Trans. PAMI*, **18**, 884–900.

Zhu, S.C. (1999). Embedding Gestalt laws in Markov random fields. *IEEE Trans. PAMI*, **21**, 1170–1187.

Zhu, W. and Wang, F. (2007). On three types of covering-based rough sets. *IEEE Trans. on Knowledge and Data Engg.*, **19**, 1131–1144.

Zunic, J. and Rosin, P.L. (2003). Rectilinearity measurements for polygons. *IEEE Trans. PAMI*, **25**, 1193–1200.

Zunic, J. and Rosin, P.L. (2004). A new convexity measure for polygons. *IEEE Trans. PAMI*, **26**, 923–934.

Author's Statement

This thesis is based on some of the following publications and patents of the author (Arindam Biswas).

Journal Publications

1. A. Biswas, P. Bhowmick, and B. B. Bhattacharya. Archival Image Indexing with Connectivity Features using Randomized Masks. *Applied Soft Computing*, vol. 8(4), pages 1625–1636, 2008.

2. A. Biswas, P. Bhowmick, and B. B. Bhattacharya. Multigrid Shape Codes and Their Applications to Image Retrieval. *Electronic Letters on Computer Vision and Image Analysis (ELCVIA)* Vol. 7, No. 2, pp. 62-75, 2008.

3. P. Bhowmick, A. Biswas, and B. B. Bhattacharya. Thinning-free Polygonal Approximation of Thick Digital Curves Using Cellular Envelope. *Electronic Letters on Computer Vision and Image Analysis (ELCVIA)* Vol. 7, No. 2, pp. 76-95, 2008.

4. A. Biswas, P. Bhowmick, and B. B. Bhattacharya. Construction of Isothetic Covers of a Digital Object: A Combinatorial Approach. *Journal of Visual Communication and Image Representation* (accepted), 2009.

5. S. Pal, P. Bhowmick, A. Biswas, and B. B. Bhattacharya. Understanding Digital Documents Using Gestalt Properties of Isothetic Components. *International Journal of Digital Library Systems* (accepted), 2009.

6. A. Biswas, P. Bhowmick, Moumita Sarkar, and B. B. Bhattacharya. A Linear-time Combinatorial Algorithm to Find the Orthogonal Hull of an Object on the Digital Plane. (under revision), December 2009.

7. P. Bhowmick, A. Biswas, and B. B. Bhattacharya. Formation of a Pointillist Ensemble from a Digital Object for Shape Perception and Easy Reconstruction. (communicated), July 2008.

8. A. Biswas, S. Pal, P. Bhowmick, and B.B. Bhattacharya. Geometric Analysis and Efficient Indexing of Digital Documents. (under revision), 2009.

Conference Publications

1. S. Pal, P. Bhowmick, A. Biswas, and B. B. Bhattacharya. GOAL: Towards understanding of Graphic Objects from Architectural to Line drawings. In *Proc. Eighth International Workshop on Graphics Recognition (GREC 2009)*, City University of La Rochelle, France, pages 71-82, 2009.

2. A. Biswas, Mousumi Dutt, P. Bhowmick, and B. B. Bhattacharya. On Finding the Orthogonal Convex Skull of a Digital Object. In *Proc. 13th International Workshop on Combinatorial Image Analysis: IWCIA'09*, Research Publishing Services, Singapore, pages 25-36, 2009.

3. A. Biswas, Moumita Sarkar, P. Bhowmick, and B. B. Bhattacharya. Finding the Orthogonal Hull of a Digital Object: A Combinatorial Approach. In *Proc. 12th International Workshop on Combinatorial Image Analysis (IWCIA-2008)*, vol. 4958 of LNCS, Springer, Berlin, pages 124-135, 2008.

4. A. Biswas, P. Bhowmick, and B. B. Bhattacharya. Extraction of Regions of Interest from Face Images Using Cellular Analysis. In *Proc. ACM Compute 2008*, ACM Digital Library, IISc. Bangalore, 2008.

5. B. B. Bhattacharya, A. Biswas, P. Bhowmick, and T. Achraya. A Fast On-chip Mean Filter Requiring only Integer Operations. In *Proc. SPIE VCIP (Visual Communication and Image Processing) Conference*, vol. 6822, doi: 10.1117/12.776602, California, January, 2008.

6. P. Bhowmick, A. Biswas, and B. B. Bhattacharya. **DRILL: D**etection and **R**epresentation of **I**sothetic **L**oosely connected components without **L**abeling. In *Proc. 6th Intl. Conf. Advances in Pattern Recognition (ICAPR)*, World Scientific, Singapore, pages 343–348, 2007.

7. A. Biswas, P. Bhowmick, and B. B. Bhattacharya. **SCOPE: S**hape **C**omplexity of **O**bjects using isothetic **P**olygonal **E**nvelope. In *Proc. 6th Intl. Conf. Advances in Pattern Recognition (ICAPR)*, World Scientific, Singapore, pages 356–360, 2007.

8. P. Bhowmick, A. Biswas, and B. B. Bhattacharya. **ICE:** the **I**sothetic **C**onvex **E**nvelope of a digital object. In *Proc. Intl. Conf. Computing: Theory and Applications (ICCTA)*, IEEE CS Press, USA, pages 219–223, 2007.

9. P. Bhowmick, A. Biswas, and B. B. Bhattacharya. Ranking of optical character prototypes using cellular lengths. In *Proc. Intl. Conf. Computing: Theory and Applications (ICCTA)*, IEEE CS Press, USA, pages 422–426, 2007.

10. A. Biswas, P. Bhowmick, and B. B. Bhattacharya. Characterization of isothetic polygons for image indexing and retrieval. In *Proc. Intl. Conf. Computing: Theory and Applications (ICCTA)*, IEEE CS Press, USA, pages 590–594, 2007.

11. P. Bhowmick, A. Biswas, and B. B. Bhattacharya. **PACE: P**olygonal **A**pproximation of thick digital curves using **C**ellular **E**nvelope. In *5th Indian Conf. Computer Vision, Graphics and Image Processing (ICVGIP)*, volume 4338 of *LNCS*, Springer, Berlin, pages 299–310, 2006.

12. P. Bhowmick, A. Biswas, and B. B. Bhattacharya. Isothetic polygons of a 2D object on generalized grid. In *Proc. 1st Intl. Conf. Pattern Recognition and Machine Intelligence (PReMI)*, volume 3776 of *LNCS*, Springer, Berlin, pages 407–412, 2005.

13. A. Biswas, P. Bhowmick, and B. B. Bhattacharya. **MuSC**: **Mu**ltigrid **S**hape **C**odes and their applications to image retrieval. In *Proc. Intl. Conf. Computational Intelligence and Security*, volume 3801 of *LNCS*, pages 1057–1063, 2005.

14. A. Biswas, P. Bhowmick, and B. B. Bhattacharya. Reconstruction of torn documents using contour maps. In *Proc. Intl. Conf. Image Processing (ICIP)*, IEEE CS Press, USA, pages 517–520, 2005.

15. A. Biswas, P. Bhowmick, and B. B. Bhattacharya. **TIPS**: On finding a **T**ight **I**sothetic **P**olygonal **S**hape covering a 2D object. In *Proc. 14th Scandinavian Conf. Image Analysis (SCIA)*, volume 3540 of *LNCS*, Springer, Berlin, pages 930–939, 2005.

16. A. Biswas, P. Bhowmick, and B. B. Bhattacharya. **CONFERM**: **Con**nectivity **Fe**atures with **R**andomized **M**asks and their applications to image indexing. In *Proc. Indian Conf. Computer Vision, Graphics and Image Processing (ICVGIP)*, Allied Publishers Pvt. Ltd., New Delhi, pages 556–562, 2004.

Patents

1. T. Acharya, B. B. Bhattacharya, P. Bhowmick, A. Bishnu, A. Biswas, M. K. Kundu, C. A. Murthy, S. Das, and S. C. Nandy. Minutia matching using scoring techniques. United States Patent No. 7,359, 532, issued on April 15, 2008.