Geometric Primitives in Digital Images:
Analyses and Applications Using Digital Geometry

Partha Bhowmick

Advisor
Professor Bhargab B. Bhattacharya

A thesis submitted for the degree of
Doctor of Philosophy

Indian Statistical Institute
Kolkata 700108, India
March 2007

Dedication

To my teachers

Acknowledgements

Before stepping into my research life, I could hardly imagine, even in my sweetest dreams that someone might read my thoughts and ideas someday. Today, as I look around, I find friends, scholars, and academicians with whom I can share my views and thoughts, in order to enrich my ideas. Perhaps it'll not be an exaggeration to say that such a learning process is indeed one of the nicest experiences that one can have in his/her life.

I get amazed. I get encouraged to feel difference of me at present with myself five years back. And here comes the opportunity to express my gratitude to those who have supported me in becoming a member of the research community.

The first person I would like to thank is my thesis supervisor, Professor Bhargab B. Bhattacharya. I am fortunate enough to have been associated with him since the year 2000 when I started my Masters dissertation. His freedom of thought, spirited composure, and an inquisitive bent of mind have made a deep impression on me. I could see myself being transformed into a researcher from a learner.

The other distinguished persons, whom I would like to acknowledge in this thesis are Dr. Arijit Bishnu, Dr. Tinku Acharya, Professor Malay K. Kundu, Professor C. A. Murthy, Professor Subhas C. Nandy, Dr. Sandip Das, Sabyasachi Dey, and Jayanta Dey, with whom I have co-authored several publications and patents reported in this thesis. Amongst them, Professor Malay K. Kundu deserves a special mention for his open-minded discussions on the possible scopes and areas of my research and for his caring concern during my early research stage.

I preserve a very special feeling about my friend Dr. Arijit Bishnu. He is the person who had constantly kept me in the right spirit in my research pursuit. I feel happy to say that I got Arijit always beside me not only as a friend, but as a co-researcher, and for extending his valuable help whenever I asked for. I still cherish — and continue to do so for years to come — the company of Arijit, Subhas-da (Professor Subhas C. Nandy), and Sandip-da (Dr. Sandip Das) in those early days of my research. Whenever I recall their unputdownable spirit and passion for problem solving, I get cheered up. They made me realize how a good company helps to upgrade a lesser man.

In the later stage, I was further enthused by Arindam Biswas, one of my closest friends, with whom I possess the age-old relation of valued friendship. His inspiring words and cheerful expression have motivated me in performing my research work in a meaningful way.

I would also like to thank 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. In particular, I would like to express my special gratitude to Professor Amit Das for his moral support and inspiration in pursuing my doctoral work at right pace and finish the same in time. His enjoyable company made my life a pleasant one and provided the necessary ambience for me to concentrate on my research over the last few years.

Finally, I should also mention here a few words about my wife, Antara, who had constantly infused immense vitality into me in continuing my doctoral dissertation for this prolonged period; about my parents who had let me go free in choosing my professional trait; and about my in-laws, my brother, my sister, and my brother-in-law for their support and inspiration to see my thesis come true.

Partha Bhowmick
partha@cs.becs.ac.in
March 2007, India

Abstract

This thesis reports some new properties, observations, and algorithms on several geometric primitives in the digital plane, and their applications to the analysis of geometric information embedded in a digital image. Points, constituting one such class of primitives, can be used as features for matching two objects in the digital plane, for which certain intricate issues have been identified. For performing efficient approximate matching of two sets of points in the digital plane, a new data structure called *angular tree*, has been introduced. To speed up the matching, a digital disc can be substituted by a regular (real) polygon, and consequently, a (digitally) *circular range query* may be replaced by the corresponding *polygonal range query* in an angular tree. It is shown that such strategy, in turn, expedites the minutia-based fingerprint matching schemes. Depending on the digital geometric properties of ridge lines in a fingerprint, a technique to assign a score to the minutia is also proposed and evaluated. To further demonstrate the capability of a set of points as a characteristic feature of a digital image, a new algorithm for the detection of corners along with the directions of incident edges have been reported.

Apart from points, a set of (digitally) straight lines defining the boundary of a digital image, carries several strong geometric properties. However, in a real-world image, a "visually straight" digital curve segment is often fragmented into multiple digital straight line segments (DSS), because of the violation of the tight DSS requirements. To extract a fewer number of straight pieces, the concept of *approximate* DSS (ADSS) has been introduced here by relaxing certain properties of a DSS. Two algorithms, one for the extraction of ADSS from a digital curve and another for determining a polygonal approximation of the curve based on ADSS, have also been described along with their applications.

Next, a few number-theoretic properties of a digital circle have been derived, which lend new insight to the interpretation, analysis, and construction of a digitally circular arc. Finally, an application of a cubic curve such as B-spline, has been studied for the removal of aberrations in a digital curve.

The descriptions of all experiments and the results of investigations have been reported in the thesis in support of the theoretical findings.

Contents

List of Algorithms

List of Figures

List of Tables

Chapter 1

Introduction

If geometry were an experimental science, it would not be an exact science, it would be subject to continual revision ... the axioms of geometry are only definitions in disguise. What then are we to think of the question: Is Euclidean geometry true? It has no meaning. We might as well ask if the metric system is true and if the old weights and measures are false, if Cartesian coordinates are true and polar coordinates are false. One geometry cannot be more true than another; it can only be more convenient.

JULES HENRI POINCARÉ

IN M. J. GREENBERG, EUCLIDEAN AND NON-EUCLIDEAN GEOMETRIES: DEVELOPMENT AND HISTORY (SAN FRANSISCO, 1980)

The nine chapters comprising this doctoral thesis present a cohesive study on certain properties and applications of low-order geometric entities in the digital plane. The work reported here is relevant to the emergent fields of digital geometry, discrete structures in the digital plane, digital image processing, feature extraction, and pattern matching. This chapter summarizes the overall flow of the thesis. Section 1.1 puts forth the motivation and gives a preliminary discourse of the chapters to follow, Section 1.2 narrates the scope of the thesis, and Section 1.3 outlines its organization.

1.1 Geometric Primitives in the Digital Plane

Theoretical interpretations and practical applications of the simple yet prevalent geometric primitives in the digital plane, such as points, straight line segments, circles, cubic curves, etc., hail long back from their successful realization in electronic display [Aken and Novak (1985), Foley *et al.* (1993)]. The evolution of digital paradigms includes several new theoretical advances, such as:

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

(ii) digital topology [Kong (2001), Kong and Rosenfeld (1996), Rosenfeld (1979)];

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

(iv) computational imaging [Asano *et al.* (2003b), Klette (1982), Klette *et al.* (1998)];

(v) theory of words and numbers [Asano and Katoh (1993), Klette and Rosenfeld (2004b), Klette and Žunić (2000), Mignosi (1991)].

Today, with the proliferating digitization of graphical objects and visual imageries [Klette and Rosenfeld (2004a,b), Rosenfeld and Kak (1982), Rosenfeld and Klette (2001)], the analytical studies and experimental validation of these geometric primitives have become indispensable in order to correlate and harness them for efficient real-world applications.

1.2 Scope of the Thesis

In this thesis, we have reported some interesting properties and empirical observations on several (digital) geometric primitives, namely points, straight line segments, circles, and cubic curves (B-splines), and studied their usefulness in extracting the inherent geometric information present in different types of 2D digital images. An appropriate set of primitives may capture a strong geometric property of the underlying object, and thus can be used for an efficient high-level description of the object and for subsequent applications involving the object in the digital plane. Some of these applications abundantly found in today's growing digital world are the following:

(i) digital curve drawing and representation [Aken and Novak (1985), Bresenham (1965, 1977), Foley *et al.* (1993), Kawamura (2003), Klette and Rosenfeld (2004a), Voss (1991)].

(ii) digital imaging and modeling [Li and Holstein (2003), Tian *et al.* (2003)];

(iii) image registration [Banerjee *et al.* (1995), Likar and Pernus (1999), Maintz and Viergever (1998), Mount *et al.* (1998), Williams and Bennamoun (2001)];

(iv) shape analysis [Antoine *et al.* (1996), Bhowmick *et al.* (2007a), Biswas *et al.* (2005c, 2007b), Buvaneswari and Naidu (1998), Pavlidis *et al.* (1997), Rosin (2000, 2003)]

(v) biometrics [Bhowmick and Bhattacharya (2004a), Biswas *et al.* (2005b), Gao (2004), Liang *et al.* (2003), Luo *et al.* (2000), Maltoni *et al.* (2003), Manjunath *et al.* (1992), Zhang *et al.* (2002, 2003, 2005)];

(vi) bioinformatics [Hoffmann *et al.* (1999), Holm and Sander (1996)];

(vii) medical imaging [Likar and Pernus (1999), Maintz and Viergever (1998), Panek and

Vohradsky (1999)];

(viii) document analysis [Bhowmick *et al.* (2007c), Das and Chanda (2001), Garris and Wilkinson (1992), Märgner *et al.* (2005), Pitrelli *et al.* (2006), Plamondon and Srihari (2000), Zhu *et al.* (2006)];

(ix) image and video retrieval [Biswas *et al.* (2004, 2007a), Ducksbury and Varga (1997), Gu and Tjahjadi (1999), Mokhtarian and Mohanna (2002), Wolf *et al.* (2000)];

(x) human motion analysis [Gleicher (1999), Jobbágy *et al.* (1999), Richards (1999)];

(xi) vehicle tracking and classification [Coifman *et al.* (1998), McCane *et al.* (2002), Mohanna and Mokhtarian (2003), Smith and Brady (1995), Zang and Klette (2003)].

A set of points in the digital plane often serves as a useful feature in many interesting applications involving digital images. Matching of one object with another of similar type can be reduced to the problem of matching the set of points of the former with the latter, if each object provides suitable characteristic feature points either in the digital plane. There exist several methods for (exact/approximate) matching of sets of points [Agarwal and Erickson (1998)], most of which, however, consider matching in the real domain. For implementation of these real-domain algorithms in the digital plane, therefore, several complex issues have to be solved. To address some of them, we have introduced in Chapter 2 a novel data structure called *angular tree*, which is realizable in the digital plane as well as in the 2D real domain. The proposed tree can be used for approximate matching of two sets of points in the digital plane efficiently [Bhowmick and Bhattacharya (2007a)]. Such a data structure allows a *polygonal range query* invoked in the algorithm on point set pattern matching in the 2D digital space. Interestingly, it has been shown that a digitally circular range query can be well-approximated by a regular polygon, and so, a circular range query may be replaced by the corresponding polygonal range query [Bhowmick and Bhattacharya (2005a)]. Adoption of this strategy expedites the process of approximately matching two sets of points representing two digital images, and produces the desired results quickly. The method includes approximating a given digital circle by an enclosing polygon using some geometric properties of a digital circle [Bhowmick and Bhattacharya (2005a)] and polygonal range query on the angular tree, following an efficient algorithm for approximate matching of one set of digital points with another.

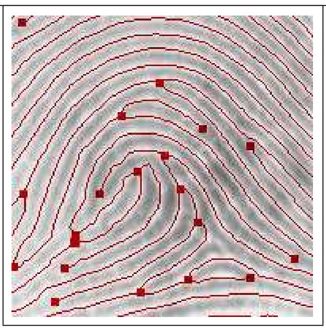
To demonstrate the usefulness of point set pattern matching (PSPM), we have considered two application domains, namely fingerprint matching and corner detection in a digital object. For example, in characterizing a fingerprint image, one of the important features is the set of minutiae [Maltoni *et al.* (2003)], which are merely sparsely occurring points (pixels) with certain unique characteristics, whereby the problem of identifying or matching a fingerprint image has a strong connection with the PSPM in the digital plane. Such an example showing the correspondence of fingerprint features with the set of digital points is given in Fig. 1.1, from which it is evident that an appropriate set of points in the digital plane can truly portray the characteristic features of the underlying object, thereby facilitating the further operations on the concerned object. It may be noted that the step-by-step generation of the final set of minutiae and their ridge directions (shown in Fig. 1.1(d)) from the gray-scale fingerprint image (shown in Fig. 1.1(a)) can be accomplished by running a series of procedures, details of which have been discussed in Chapter 3 [Bhowmick and Bhattacharya (2004a), Bhowmick *et al.* (2002, 2005a), Bishnu *et al.* (2002, 2006a)].

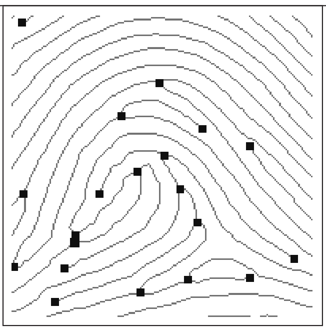
Another instance of a very pertinent problem on sets of points in the digital plane is detection of corners present in an image, as illustrated in Fig. 1.2. Given a digital image, the set of corners (along with directions of their incident edges) associated with the image is very likely to express the inherent geometry that might characterize a useful feature set of the image. Hence, detection of corners in a digital image is a contemporary problem in the area of computer vision. There exist several works and methodologies on corner detection [Alkaabi and Deravi (2004), Banerjee *et al.* (2004), Freeman and Davis (1977), Koplowitz and Plante (1995), Mokhtarian and Suomela (1998), Smith and Brady (1997), Zheng *et al.* (1999)]. We have developed and implemented a novel algorithm called CODE [Bhowmick and Bhattacharya (2004b, 2005b)], details of which are presented in Chapter 4. CODE can efficiently extract the corners along with directions of their incident edges from an image, whether gray-scale or binary. Some results on the detected corners and their incident edge directions obtained by running CODE are shown in Fig. 1.2.

Similarly, for a digital image with fairly straight edges, the set of (exact or approximate) digital straight line segments (DSS) carries a strong geometric property of the underlying image. There exist several DSS recognition algorithms till date, which can efficiently determine the (digital) straightness of a given one-pixel-thick digital curve. On a set of curves representing a real-life object/image, however, these algorithms often produce a large number of segments owing to the enforced constraints inherited from the fundamental properties of DSS. A curve segment that appears reasonably straight but fails to satisfy all the DSS properties, is fragmented into multiple DSS in these algorithms.

In Chapter 5 of this thesis, we have introduced the novel idea of extracting approximate digital straight line segments (ADSS) from a digital curve by relaxing certain digital geometric properties of DSS. In many instances, a long portion of the curve that looks visually straight, may fail to satisfy all the necessary conditions of being a single DSS,

(a) An 8-bit gray-scale image of a fingerprint.


(b) Minutiae and ridgelines detected in the fingerprint after processing and analyzing the gray-scale image.


(c) Minutiae are digital points, and the ridgelines are digital curves, which can sufficiently convey the information present in the gray-scale fingerprint image.

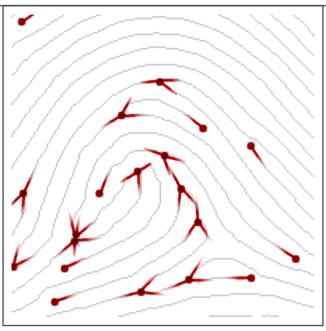

(d) Further simplification of the feature set consisting of the minutiae coupled with ridgelines is achieved by considering the orientation(s) of the incident ridgeline(s) for each minutia.

Figure 1.1: The set of minutiae (along with the ridge orientations) is merely a set of digital points that describes an effective set of features corresponding to a fingerprint image.

and hence is fragmented into multiple DSS. We have shown that the number of ADSS extracted from a real-world digital curve is remarkably fewer than that of DSS. A low-level algorithm to extract the segments that are *approximately straight* from a given set of digital curves is also presented so that the output complexity (i.e. number of extracted segments) is appreciably lesser than that of (exact) DSS.

Further, in Chapter 5, we have proposed an algorithm for polygonal approximation of the ADSS obtained from a given digital curve. Since the set of ADSS provides a fairly correct approximation of the input curve, it produces the desired approximate polygon corresponding to the input curve based on a specified *error tolerance* and an approxima-

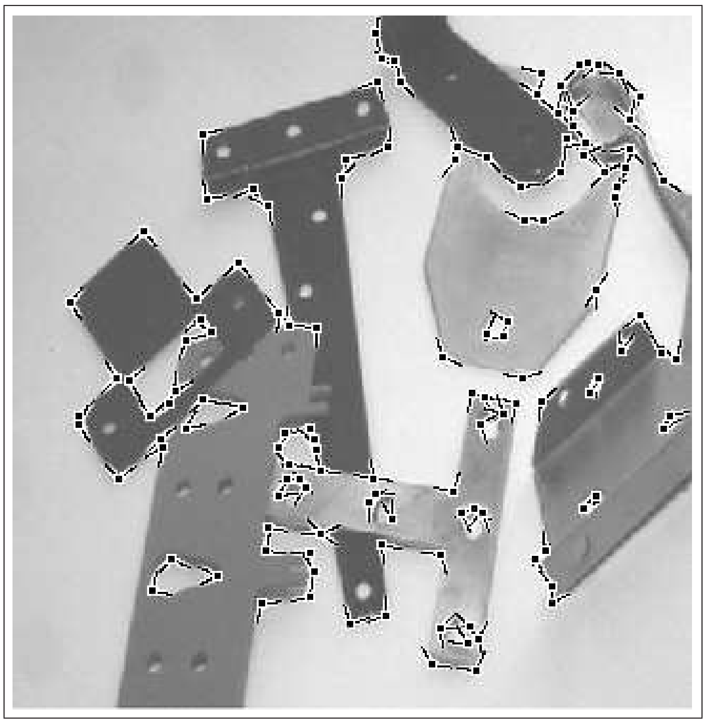


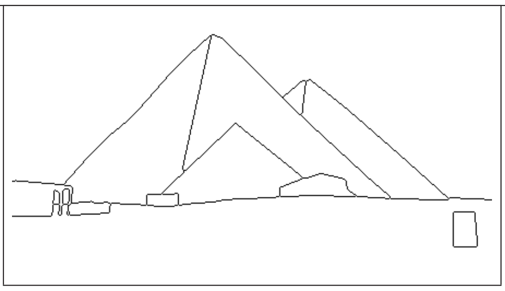
Figure 1.2: Corners and directions of their incident edges in a gray-scale image.

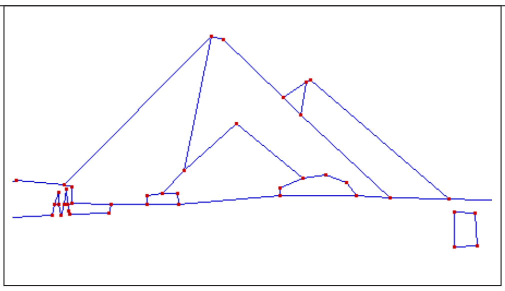
tion criteria. The overall time complexity being strictly linear in the number of points constituting the input digital curve, and since all computations involve primitive integer operations only, the entire process has been found to be very fast, efficient, and robust, as evident from the experimental results reported in that chapter. An application of the proposed method has been shown in Fig. 1.3, which depicts a polygonal approximation of a real-world digital curve defining the edge map of a "pyramid" image.

It may be noted in this context that several other methods [Chen and Chung (2001b), Climer and Bhatia (2003), Guru *et al.* (2004), Xie and Ji (2001)] have been proposed earlier for (approximate) line detection. However, the major difference of our method from the others lies in exploiting fundamental properties of DSS, whereas, the existing methods mostly used parametric approaches, such as distance criteria, usage of masks, eigenvalue analysis, Hough transform, etc.

After analyzing digital line segments, we have studied certain aspects of a digital circle. Several new number-theoretic properties of a digital circle are discussed in Chapter 6, which may have significant impact on the construction and analysis of a circular arc in the

(a) An 8-bit gray-scale “pyramid”.


(b) The edge map of “pyramid” is considered to be a real-world digital curve. Note that the edge map is subject to the parameter(s) specified in the edge extraction algorithm.


(c) Polygonal approximation of the edge map in (b) with the vertices shown in red color and the edges in blue color.

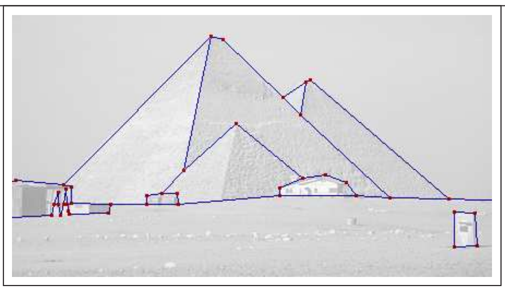

(d) Polygonal approximation superimposed on the original gray-scale image shows how well our algorithm can approximate a real-world image.

Figure 1.3: Polygonal approximation of an image “pyramid”.

digital plane. For instance, we have shown that the squares of abscissas of equi-ordinate grid points defining the circumference of a digital circle of a given radius r, lie in a unique interval (of positive integers), decided by the corresponding ordinate, and of course, by the radius of the concerned circle; also, the run length (i.e., the number) of grid points with ordinate $j-1$ never exceeds by more than one, the run length of its grid points with ordinate j.

The number-theoretic analysis, laid down in Chapter 6, establishes the relation of a digital circle with the distribution of square numbers in discrete intervals. This provides a new way of understanding a digital circle concerning its visualization and characterization. Further, two simple algorithms for construction of a digital circle, based on number-theoretic concepts, have been developed, both of which use a few primitive operations only, and are completely free from any floating-point operation. Several illustrations and test results have been furnished that elucidate the analytical power and algorithmic efficiency of the proposed approach. It has been also shown, how and why, for sufficiently high radius, the number-theoretic technique would expedite a circle construction algorithm significantly.

Regarding the polygonal approximation of a digital circle, we show that an ideal (regular and convex) polygon corresponding to a digital circle is possible for some of the digital circles, especially for the ones having smaller radii. For a circle whose ideal regular polygon does not exist, an approximate polygon, tending to the ideal one, can be constructed, in which the error of approximation can be controlled by the number of vertices of the approximate polygon. These results have been reported in Chapter 7. Polygonal enclosures of digital circles have several applications in approximate point set pattern matching. One such application is circular range query in the digital plane using angular tree, which we have discussed in Chapter 2. We have reported the conditions under which an ideal regular polygon certainly exists corresponding to a digital circle, and also the cases for which the existence of an ideal regular polygon becomes uncertain. Experimental results demonstrate the possibilities of approximation and the trade-off in terms of error versus approximation.

In Chapter 8, we have shown how a higher order curve, such as B-spline, can be used to aid the smoothing/processing of a digital curve. Such operations are often needed in several applications where digital curves play crucial roles. One such application where the digital curve (in the form of a fingerprint ridgeline) plays a significant role is the Automatic Fingerprint Identification System (AFIS). Most of the existing AFIS still suffer from improper treatment of poor quality fingerprint images. Several fingerprint image enhancement techniques are well known that aim at rectifying the deformities in a fingerprint image.

The work in Chapter 8 deals with one such problem besetting fingerprints — that of eliminating digital aberration. The proposed method, in short, mainly involves fitting of B-splines for a set of control points chosen appropriately for each ridgeline in a fingerprint image; and these fitted splines, in turn, output a less distorted, if not undistorted, view of the concerned fingerprint. Experimental results on several databases have been presented that justify the strength and elegance of the proposed algorithm.

1.3 Organization of the Thesis

Keeping in view the order of the geometric primitives, namely points, line segments, circles, and cubic curves, we have organized the rest of the thesis as follows. In Chapter 2, we deal with sets of points in the digital plane, and show the construction and the usage of an angular tree in the polygonal range query on a set of points in the 2D plane [Acharya *et al.* (2003a), Bhowmick and Bhattacharya (2004a, 2005a, 2007a)]. In Chapter 3, we show how the minutiae comprise a representative set of digital points in a fingerprint image for fingerprint matching [Acharya *et al.* (2003b, 2004, 2006), Bhowmick *et al.* (2002, 2005a), Bishnu *et al.* (2002)]. In Chapter 4, we describe how the corners can be extracted in the form of digital points in the case of a general image [Bhowmick and Bhattacharya (2004b)]. In Chapter 5, we discuss in brief a few digital-geometric properties of a digital straight line segment (DSS), and show how these properties can be relaxed to define a new concept of approximate digital straight segments (ADSS), and subsequently, how the latter can be exploited to approximate a digital curve by a polygon using an approximation parameter [Bhowmick and Bhattacharya (2007b)]. Chapter 6 focuses on some number-theoretic properties of a digital circle, which, can be used to develop an alternative algorithm for constructing a digital circle [Bhowmick and Bhattacharya (2006a)]. In Chapter 7, the feasibility and accuracy of polygonal approximation of a digital circle have been studied [Bhowmick and Bhattacharya (2005a)]. These results are needed to substantiate the polygonal range query as discussed in Chapter 2. Chapter 8 presents how B-splines can be used for smoothing fingerprint ridges, before invoking an automated fingerprint identification system [Bhowmick and Bhattacharya (2006b)]. Finally, in Chapter 9, we draw the concluding notes on the summary of this thesis, and discuss future directions.

Chapter 2

Sets of Points and their Approximate Matching in the Digital Plane

It is the mark of an instructed mind to rest assured with that degree of precision that the nature of the subject admits, and not to seek exactness when only an approximation of the truth is possible.

ARISTOTLE

2.1 Introduction

Approximate Point Set Pattern Matching (APSPM) is a challenging problem that has numerous interesting applications in the digital plane. Some of these applications are as follows:

(i) image registration [Banerjee *et al.* (1995), Likar and Pernus (1999), Maintz and Viergever (1998), Mount *et al.* (1998), Williams and Bennamoun (2001)];

(ii) model-based object recognition [Baird (1985), Eric and Grimson (1990)]

(iii) biometrics, e.g. fingerprint matching [Bhowmick and Bhattacharya (2004a), Jain *et al.* (1997), Jiang and Yau (2000), Luo *et al.* (2000), Maltoni *et al.* (2003)];

(iv) bioinformatics, e.g. protein structure and function determination [Hoffmann *et al.* (1999), Holm and Sander (1996)];

(v) medical imaging [Likar and Pernus (1999), Maintz and Viergever (1998), Panek and Vohradsky (1999)];

(vi) human motion representation for gait analysis, sports studies, animation, computer games, etc. [Gleicher (1999), Jobbágy *et al.* (1999), Richards (1999)];

(vii) digital imaging and modeling [Li and Holstein (2003), Tian *et al.* (2003)];

(viii) drug design [Finn *et al.* (1997)];

(ix) vehicle tracking [Coifman *et al.* (1998)].

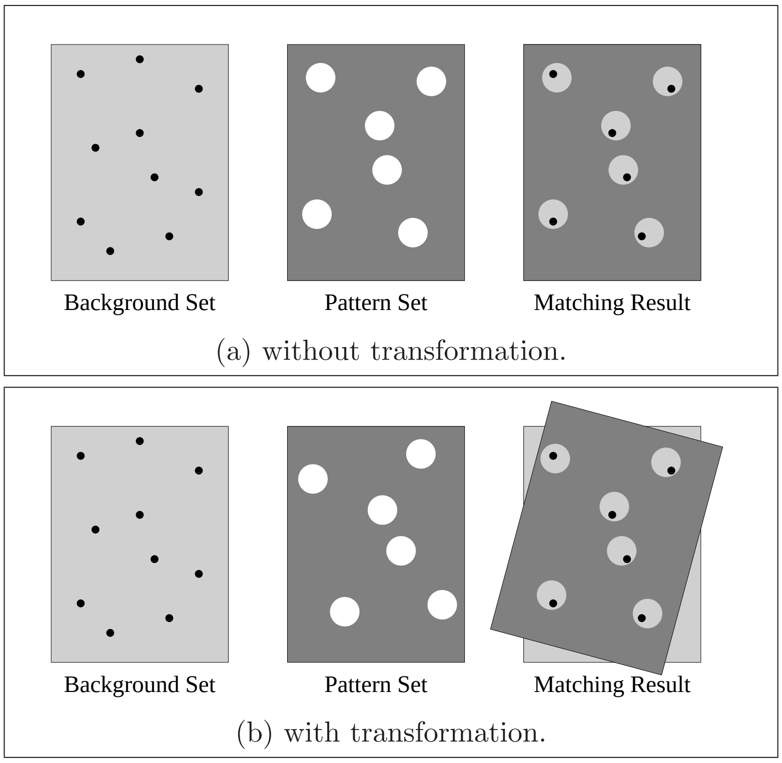


Figure 2.1: Approximate matching of two sets of points in 2D (real) plane.

The underlying abstract problem of APSPM may be envisaged as follows. If a board has some pins randomly fixed on it (background set, Q), and another has some randomly located holes (pattern set, P), then, using the term "board" synonymously with the term "set of points", the APSPM problem is to check whether the board P can somehow be placed on the board Q so that in each hole of P, we can see exactly one pin of Q.

If no transformation is allowed, then checking for a match simply means that we have to place the board P over the board Q, and count the number of holes containing one pin each (Fig. 2.1(a)). Finally, (approximate) matching decision can be taken based on the number of holes containing the pins. For matching under transformation, the board P has to be translated and/or rotated appropriately so as to maximize the number of holes containing pins (Fig. 2.1(b)).

In a discrete domain, the location of each pin in Q is constrained by its integer coordinates, and each hole in P is a digital disc having integer radius and a center with integer coordinates. Further, for faster query processing, each digital disc may be replaced by a suitable regular (real) polygon with its center coinciding with the center of the corresponding disc. The regular polygon should be such that it should not produce a matching result different from the one obtained with the digital disc. This can be guaranteed if and only if there exists no grid point in the digital disc that lies outside the polygon, and conversely, there exists no grid point outside the digital disc that lies on or inside the polygon. Such a polygon is called "ideal". In Chapter 7, we have shown that an ideal regular polygon corresponding to a digital disc is analytically possible for some of the digital discs, especially for the ones having lower radii; for larger digital discs, approximate polygons, tending to ideal ones, are possible, which would have very low error rates. The usage of a regular polygon instead of a digital disc for range query on a digital plane simplifies and expedites the process. Polygonal range query can be efficiently serviced by using an *angular tree*, a new data structure, which is introduced in this work. The proposed angular tree is marked by its admissibility to a query on the 2D plane using any convex query polygon of arbitrary shape and size. We have also shown that, for a query in set Q containing n points using such a query polygon having an arbitrarily large number of vertices, the worst-case time complexity of the proposed range query is bounded by $O(n)$, and hence, asymptotically independent of the complexity (i.e., number of vertices) of query polygon. For a smaller size query polygon, of course, the query time is less, and is bounded by $O(\log n)$.

In this work, we have chosen a regular (convex) polygon instead of a (general — regular or irregular) convex polygon to approximate a digital disc in order to substitute a circular region of query by a polygonal range query, since the former offers some symmetry (in resemblance to the 8-axis symmetry of a digital disc) as opposed to the latter. In fact, we have used a regular $2k$-gon ($k \geq 2$) to implement our APSPM algorithm, since each edge e of a regular $2k$-gon has a symmetric edge e' in the same direction as that of e. This, in effect, reduces the dimensionality of the associated angular tree (when $k \geq 3$), and hence reduces subsequent computations because of fewer transformed axes (as the directions of the axes are given by those of the edges of the query polygon). Thus, a polygon with an even number of vertices is easy to handle.

For better approximation of the digital disc, the number of vertices of the regular polygon may be increased for improved matching. However, the space and time complexities of the allied algorithms and the dimensionality of the supporting data structures will increase significantly. Having an approximating regular polygon with fewer vertices (with a trade-off on the acceptable error part) will simplify the query process and expedite the subsequent matching process with desired accuracy. In most of the practical applications related to digital images, the circular ranges used in approximate matching are found to have reasonably small radii. Thus, an approximating polygon requires only a few vertices to achieve a desired level of matching accuracy. For instance, in fingerprint matching [Bhowmick and Bhattacharya (2004a), Bhowmick *et al.* (2005a)] using the algorithms of APSPM, the circular range query is evoked for a circular range with radius not exceeding 10 pixels, which can be "ideally" replaced by a regular polygon.

Several methods on APSPM are available in the literature, most of which aim at matching in the real domain. Discussions on these algorithmic techniques for shape matching, simplification, and morphing using range search may be found in the survey paper by [Agarwal and Erickson (1998)]. Typically, rectangles, half-spaces, simplices, or balls are used for range queries. Implementations of these real-domain algorithms in the digital plane, however, pose difficult problems because of the complexity of the underlying data structures, which have to be tailored appropriately to suit the digital domain. A few empirical methods for geometric pattern matching algorithms in the digital space have been reported earlier [Hagedoorn and Veltkamp (1999), Huttenlocher and Rucklidge (1993), Mount *et al.* (1998)], which are usually based on an exhaustive search of the transformation space with suitable pruning techniques.

In this chapter, we present a novel data structure called *angular tree* that can be deployed in the 2D digital space (and in the 2D real space also, if required) and whose (angular) precision can be controlled depending on the requirement of the application. The angular tree can be used for (regular) polygonal partitioning of 2D digital points to enable the *circular range query* required for the algorithm on APSPM in the digital plane. Since it has been shown [Bhowmick and Bhattacharya (2005a)] that a digitally circular range query can be well-approximated by a regular (real) polygon in 2D, a circular range query may be replaced by the corresponding polygonal range query, thereby producing the desired results of approximate matching (of sets of points) for practical applications related to digital images. The process of matching has been further expedited by consideration of the farthest point pair (as the first correspondence between two sets of points) for initial anchoring, whose rationale is also given in this chapter. Relevant discrete structures and methodologies of computational geometry have been used along with some geometric features of digital discs for successful realization of the APSPM algorithm.

A short schematic diagram of our solution to the APSPM problem has been shown in Fig. 2.2 on a small instance of the pattern set and the background set. Here the approximation parameter is $\varepsilon = 3$, which is the radius of the circular region/disc centered about the concerned point of the set of points in the digital plane. The pattern set P contains 9 points (shown in small black disks) and the background set Q contains 10

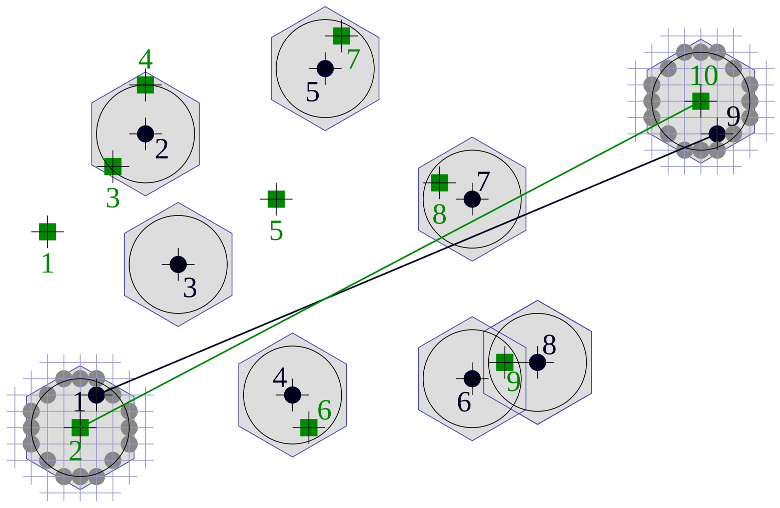


Figure 2.2: A small instance demonstrating our matching scheme for approximation parameter $\varepsilon = 3$.

points (shown in small green squares). The points of both P and Q have been labeled with their index numbers increasing from left to right just for sake of easy readability. In the actual problem, however, the points of P and Q are arbitrarily numbered. After anchoring the farthest point pair $\langle p_1, p_9 \rangle$ of P with (the ε-neighborhood of) the point pair $\langle q_2, q_{10} \rangle$ of Q, each other point of P is searched in Q using a (regularly shaped) polygonal (hexagonal in this instance) range query that replaces the corresponding (digital) circular range query in the set Q. Each circular/polygonal range is defined with the corresponding point of P as center. The polygonal range query in Q is realizable by the *angular tree* defined on Q proposed in this chapter (not shown in this diagram). It may be noticed that there may be some point (p_3) in P for which the (circular range) query does not return any point from Q. There may be also some point (p_2) in P for which the query may return multiple points (q_3 and q_4) from Q, or there may be multiple points (p_6 and p_8) in P returning the same point (q_9) from Q. We have considered all these cases with proper treatments in our algorithm.

The rest of the chapter is organized as follows. We start with the discussion on how to approximate a digital disc of given radius by a tight (regular) enclosing polygon in Sec. 2.2. Section 2.3 elucidates the method of approximate matching of sets of digital points. In Sec. 2.4, the angular tree and the circular range query in 2D digital space have been described. Experimental results have been demonstrated in Sec. 2.5. Conclusions and discussions on future directions appear in Sec. 2.6.

2.2 Approximation of a Digital Circle by a Regular Polygon

An efficient approximation of a real circle from the continuous domain to the digital domain owes its origin to the early adoption of scan-conversion technique (Chapter 6 and Chapter 7). In a recent work [Bhowmick and Bhattacharya (2005a)] that explores and exhibits the subtleties and various possibilities on approximation of a digital circle by a (real) regular (convex) polygon, we have shown that such ideal regular polygons are analytically possible for some of the digital circles, especially for the ones having lower radii; and for circles with higher radii, ideal regular polygons are rarely possible. However, in reality (as revealed by exhaustive procedural results), approximate polygons, tending to ideal ones, are possible, which would have very low error rates. In Chapter 7, we have reported the conditions for which an ideal regular polygon definitely exists corresponding to a digital circle, and the conditions under which the existence of an ideal regular polygon becomes uncertain. Furthermore, when the complexity (i.e., number of vertices) of the ideal regular polygon is high, or when an ideal regular polygon does not exist, a regular polygon with smaller number of vertices can be a useful approximation of the ideal regular polygon, which have been discussed and substantiated with necessary experimental findings in Chapter 7.

A pertinent question in the context of this work is whether there always exists such a regular polygonal enclosure for a digital circle $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$. As shown by Bhowmick and Bhattacharya (2005a), a regular polygonal enclosure for $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$ certainly exists, provided there exists a grid point μ in $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$ such that μ has simultaneously maximum isothetic distance, namely δ_μ, and maximum radial distance, namely ε_μ, from $\mathcal{C}^{\mathbb{R}}(\alpha, \rho)$. Here, the isothetic distance of μ from $\mathcal{C}^{\mathbb{R}}(\alpha, \rho)$ is given by the minimum between the horizontal distance and the vertical distance (in Euclidean metric space on $\mathbb{R}^2$) of μ from $\mathcal{C}^{\mathbb{R}}(\alpha, \rho)$, whereas, its radial distance indicates its (Euclidean) distance from $\mathcal{C}^{\mathbb{R}}(\alpha, \rho)$. In short, if there exists a grid point $\mu \in \mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$, such that $\delta_\mu = \max\left\{\delta_\nu : \nu \in \mathcal{C}^{\mathbb{Z}}(\alpha, \rho)\right\}$ and $\varepsilon_\mu = \max\left\{\varepsilon_\nu : \nu \in \mathcal{C}^{\mathbb{Z}}(\alpha, \rho)\right\}$, then the existence of a regular polygonal enclosure for $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$ is certain; otherwise it is uncertain. In Chapter 7, we have given some experimental results on (exact/approximate) polygonal enclosures of digital circles, which testifies the feasibility of replacing a (digitally) circular query by a regular polygonal range query for effectuating an efficient APSPM algorithm in 2D digital space, as explained in Sec. 2.3.

Since it has been shown that for all digital circles, especially for most of the circles with radii exceeding 10 units, ideal enclosing polygons are not possible, but very good approximate polygons with minor error margins are possible, specifying a proper shape (i.e., number of vertices) of the approximate polygon is important. The permissible error in the process of approximation, as explained in and testified by the error distribution plot in Chapter 7, indicates the number of vertices of the approximate polygon. That is, given the radius $(\rho := \varepsilon)$[1] of a digital circle and the error incurred in approximating the digital circle by a regular polygon, we can determine the number of vertices (which is $2k$ for even number of vertices of the respective polygon.

2.3 APSPM in 2D Digital Plane

In Approximate Point Set Pattern Matching, given two sets of points, P and Q, the problem is to find a transformation, matching each point $p \in P$ into the ε-neighborhood ($\varepsilon \geqslant 0$) of some unique point $q \in Q$. It has two broad variations, namely (i) pattern matching problem and (ii) largest common point set problem. In a pattern matching problem, given two sets of points, P and Q, where $|P| = m, |Q| = n$, with $m \leqslant n$ and $\varepsilon \geqslant 0$, it is required to find a transformation T such that $|d(T(P), Q)| \leqslant \varepsilon$. On the other hand, in a problem involving the largest common set of points, it is required to find a transformation T and a set $P' \subseteq P$, such that $|P'| \geqslant n_{\min}$ and $|d(T(P'), Q)| \leqslant \varepsilon$. For these two problems, T is the required transformation given to P, $T(P)$ is the transformed set of points after applying the transformation T to P, and $|d(T(P), Q)|$ denotes the maximum distance in the set of distances between the corresponding points of $T(P)$ and Q.

If $\varepsilon = 0$, then each of the above problems reduces to Exact Point Set Pattern Matching (EPSPM); otherwise, ε is a positive quantity, and the problem is referred to as Approximate Point Set Pattern Matching (APSPM) under ε. Each of the above problems again can have two different variations, namely the *decision problem* and the *optimization problem*. In the case of a decision problem, the value of ε is supplied, and the output of the program is either a “match” or a “mismatch”, depending on whether or not the two sets of points approximately match for the given value of ε. On the contrary, the optimization problem determines the optimized value of ε, so that P and Q match approximately under

[1] ρ is used to represent the radius of a digital circle in this Section, whereas ρ is replaced by ε in the subsequent Sections, since the latter is conventionally used to indicate the measure of approximation point set pattern matching (APSPM), and we use this to define the radius of a digital circle involved in the approximation procedure.

ε. In this chapter, we focus on the decision version of the APSPM problem, with references to EPSPM depending on the relevance.

2.3.1 Existing Algorithms

Several methods have been developed in the past for solving the pattern matching problem. It is not difficult to design and implement a polynomial time algorithm for EPSPM under rigid motion, but the challenge lies in the APSPM version. A brief overview of some existing algorithms for both types of pattern matching is given below.

2.3.1.1 String Matching Algorithm

A problem on EPSPM can be easily reduced to string matching, provided both P and Q have the same number of points, say n. In the literature, there exist several forms of this class of algorithms with $O(n \log n)$ time complexity, proposed independently by several authors [Atkinson (1998), de Rezende and Lee (1995)]. In this approach, the polar coordinates of the points in P and Q with respect to their geometric centroids are used to prepare the corresponding ordered lists (i.e., lexicographically sorted sequences — angle first, distance second), say $\mathcal{L}_P$ and $\mathcal{L}_Q$, to find out the exact match between P with Q, by checking whether $\mathcal{L}_Q$ is a substring of $\mathcal{L}_P$ (under any cyclic shift), using a fast string matching algorithm. It is easy to see that P and Q are exactly congruent if and only if the algorithm gives a positive answer.

2.3.1.2 Alignment Scheme in EPSPM

In the alignment scheme [Bishnu *et al.* (2003, 2006b)], the number of points in the two sets P and Q, namely m and n respectively, are not necessarily equal; wherefore, w.l.o.g., if $m \leqslant n$, then $P \subseteq Q$ is considered as a valid match, which is an improvement over the algorithm based on string matching (Sec. 2.3.1.1). In the alignment scheme, the Euclidean distances for all point pairs in the background set Q are stored in an ordered list (primary data structure); moreover, for each point q of Q, the distances of all other points of Q from q are stored in a separate ordered list (secondary data structure). During matching, an arbitrary point pair of P is aligned with each of the k matching point pairs of Q, as found from the primary structure. After applying the necessary transformation to P, if the (transformed) points of P are found to match with points of Q, which is verified from the secondary structure, then Q matches with P under the transformation. On the contrary, if no such transformation exists, then Q is said to have no match with P. In this method, the construction of the primary and secondary data structures needs $O(n^2 \log n)$ time, whereas, the time complexity of the (successful or unsuccessful) matching process is bounded by $O(km \log n) = O(mn^{4/3} \log n)$, since the maximum number k of a particular Euclidean distance present in Q is bounded by $\lceil n^{4/3} \rceil$ [Szekely (1997)].

2.3.1.3 Alignment Scheme in APSPM

Different alignment schemes for problems on APSPM have been reported earlier [Alt *et al.* (1988), Arkin *et al.* (1992), Erafat and Itai (1996)]. In an alignment scheme, given ε as the Euclidean tolerance for matching under arbitrary rigid motions between the sets of points, P and Q, a valid matching of P with Q is said to exist, provided each point in P lies in the (ε-) neighborhood of some point in Q. Mapping two points $p, p' \in P$ onto the (circular) boundaries of $q, q' \in Q$ leaves one degree of freedom, which is parameterized by the angle $\phi \in [0, 2\pi)$ between the vector from q to p and the horizontal line through q. If we consider any other point $p'' \in P$, then p'' will trace an algebraic curve of degree 6 that may intersect the boundary of the disc of radius ε with center at some point $q'' \in Q$ at most 12 times. Thus, there can be at most 6 intervals of the parameter ϕ for which p'' lies in the neighborhood of q''. The parameter space $[0, 2\pi)$ is, therefore, partitioned into $O(n^2)$ intervals, each interval corresponding to a unique point pair in P, so that for all ϕ in one interval, the points of P are mapped into the neighborhoods of points in Q. All these relationships are represented as edges in a bipartite graph whose two sides of nodes are Q and P. The decision problem has a positive solution exactly, if there is some ϕ for which the corresponding graph has a perfect matching. This is checked by finding the graph for the first sub-interval of $[0, 2\pi)$ and constructing a maximum matching for it. Next, while traversing the sub-intervals from left to right, the maximum matching is updated until a perfect matching is found or it turns out that none exists.

Apart from the fact that the total running time of the above algorithm is $O(n^8)$, determining the intersection points of the curve of degree 6 with circles poses nontrivial numerical problems. For easier variations of the one-to-one matching problem, simpler and faster algorithms were found; e.g., for the case of translation only, geometric arguments have been used by Erafat and Itai (1996), whereas in the algorithm proposed by [Arkin *et al.* (1992)], the ε-neighborhoods of the points are assumed to be disjoint. In addition to these algorithms, there are also several others based on the alignment scheme, such as Basic Alignment [Goodrich *et al.* (1994)], Multiple Grid [Indyk *et al.* (1999)], a comparative study of which appeared in the survey paper by Alt and Guibas (1996).

2.3.1.4 Hausdorff Metric

A widely used metric for approximate matching of one set of points with another is the Hausdorff distance [Huttenlocher and Rucklidge (1993), Huttenlocher *et al.* (1992), Mount *et al.* (1998)]. Given two sets of points, P and Q, the Hausdorff distance *from P to Q* is given by $h(P,Q) = \max\{\min\{d(p,q) : q \in Q\} : p \in P\}$, where $d(p,q)$ is the Euclidean distance between points p and q, and the number of points in P does not exceed that in Q. Thus, $h(P,Q)$ denotes the smallest amount by which we need to "grow" the points of Q in order that all points in P are covered by the grown set. Hence, given two sets of points, P and Q, and the space of transformations $\mathit{\Gamma}$, P is said to match with Q within a tolerance parameter ε, provided $h^{(\mathit{\Gamma})}(P,Q) = \min\{h(T(P),Q) : T \in \mathit{\Gamma}\} \leqslant \varepsilon$.

For approximate matching under arbitrary rigid motion, the concept of a "dynamic" Voronoi diagram is used [Huttenlocher *et al.* (1992)], which is the subdivision of the 3-dimensional space-time whose cross section at a time t equals the Voronoi diagram at t. The upper bound on the complexity of the dynamic Voronoi diagram, hence obtained from the number of topological changes undergone by the Voronoi diagram with passage of time, is subsequently used to show that the time complexity of an optimal match between two sets of points using Hausdorff distance is $O((m+n)^6 \log(mn))$.

2.3.1.5 β-approximate Pattern Matching

In the framework of β-approximate pattern matching [Heffernan and Schirra (1994)], for a given tolerance $\varepsilon > 0$, the approximate congruence between P and Q is defined as $\max\left(h^{(\mathit{\Gamma})}(P,Q), h^{(\mathit{\Gamma})}(Q,P)\right)$. The process of approximation is relaxed by allowing to "give up", if ε is found to be "too close" to $\varepsilon^* = h^{(\mathit{\Gamma})}(P,Q)$ at any point of time, which, however, may incur a large computing time in case of no approximation. Further, an extra parameter, namely $\beta > 0$, that corresponds to the accuracy of approximation is required to take the matching decision of P with Q as follows: (i) if $\varepsilon^* \leqslant \varepsilon$, then return a transformation $T \in \mathit{\Gamma}$, such that $h(T(P),Q) \leqslant (1+\beta)\varepsilon$; (ii) if $\varepsilon < \varepsilon^* \leqslant (1+\beta)\varepsilon$, then return the required transformation T; (iii) if $\varepsilon^* > (1+\beta)\varepsilon$, then return NONE. The rationale is based on the fact that, determining an answer close to optimal is harder than determining an answer that is further away. Hence, if ε^* lies in the given approximation range $[\varepsilon^*, (1+\beta)\varepsilon]$, then this approach gives an answer close to the prescribed error bound.

There exist several other techniques for solving APSPM and related problems. For example, Parui *et al.* (1993) constructed the digital Voronoi diagram of a set using Euclidean distances between pairwise points/pixels of an image. From this diagram a certain planar graph is constructed, which is a subgraph of the Delaunay triangulation of the set of points. Finally, the shape of the set of points is computed as a subgraph of the planar graph.

2.4 Proposed Method

We have proposed here a novel algorithm on APSPM in 2D digital space that extends the conventional orthogonal range query (e.g., kd-tree [Bentley (1975, 1979), Berg *et al.* (2000)]) to a (digitally) circular range query. The circular range query, in turn, can be replaced by a (regular and convex) polygonal range query, where the polygon (in $\mathbb{R}^2$) may be judiciously selected depending on the value of the digital tolerance $\varepsilon := \rho$, since a digital disc of radius ρ can be replaced by a suitable regular polygon, as explained in Sec. 2.2.

The proposed algorithm uses the angular tree, denoted as $\mathcal{T}_\theta(Q)$, defined on a set of points, $Q := \{q_1, q_2, \ldots, q_n\} \subset \mathbb{Z}^2$, that augments the structure of a kd-tree [Bentley (1975)] as follows. A kd-tree refers to a k-dimensional tree. If a set of points is defined in 2-dimensional space (real or digital), then a kd-tree is a balanced tree, meant for answering 2-dimensional orthogonal range query. Now, given the set of points Q, the kd-tree defined on Q permits us to search for points contained within a specified orthogonal box. Since we need circular range query in Q, where the number of vertices of the regular polygon that replaces a (digital) disc of radius ε depends on ε and the desired degree of precision associated with the query, we require angular tree $\mathcal{T}_\theta(Q)$ to suit our purpose. The speciality of $\mathcal{T}_\theta(Q)$ is that it accepts all possible values of $k(\geqslant 2)$ for a $2k$-gonal range query in Q. For instance, if $k = 3$, then we can search for points in Q within a (regular) hexagonal region. Here, θ indicates the internal angle of the respective query polygon in the corresponding tree $\mathcal{T}_\theta(Q)$; that is, $\theta = ((k-1)/k)\,180^o$. Hence, specifying the value of k as well describes (dimensionality of) $\mathcal{T}_\theta$, wherefore we have used k synonymously with θ in the subsequent discussions.

2.4.1 Construction of Angular Tree

To illustrate the construction of an angular tree and associated operations, we first start with $k = 2$, and then go for the generalization. For $k = 2$, we split the given set of points,

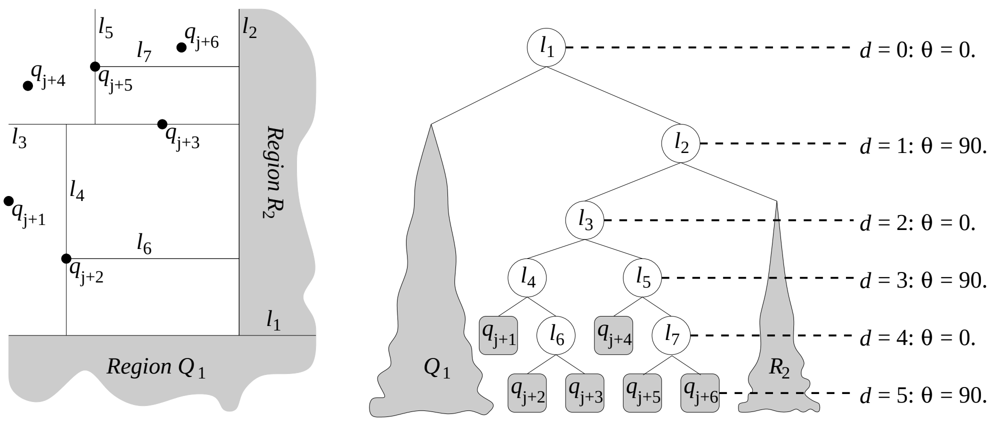


Figure 2.3: A set of points, Q, and its corresponding angular tree, $\mathcal{T}_\theta$, for $\theta = 90^o (k = 2)$.

Q of n points, by a horizontal (median) line at the root ($depth = 0$) of the tree into two subsets, namely Q_1 and Q_2, whose sizes differ by at most unity. The splitting line being stored at the root, and Q_1 and Q_2 being stored in the respective left and right sub-trees of the root, Q_1 and Q_2 are subsequently partitioned recursively by appropriate vertical and horizontal (median) lines alternately to obtain the desired $\mathcal{T}_\theta$ for $k = 2$. Thus, for $k = 2$, we split a region by a horizontal line (i.e., 0^o line) at a node if its depth is even, and by a vertical line (i.e., 90^o line) at a node if its depth is odd. The general structure of $\mathcal{T}_\theta$ for $k = 2$ is shown in Fig. 2.3, where R_1 and R_2 represent the respective left and right sub-trees of Q_2 (R_1 is shown in detail and R_2 is similar). It is easy to observe that, for $k = 2$, $\mathcal{T}_\theta$ is structurally same as the corresponding 2-dimensional kd-tree.

In general, if the splitting line be inclined at an angle ϕ with the horizontal line, then at depth d, we have $\phi = (d \ mod \ k) \times (180/k)$. For example, for $k = 3$, we split a set of points, Q, by a 0^o line at depth $d = 0$ (root), by a 60^o line at $d = 1$, by a 120^o line at $d = 2$, by a 0^o line at $d = 3$, by a 60^o line at $d = 4$, and so on. The angular tree for $k = 3$ corresponding to a small instance of the set of points, Q, is shown in Fig. 2.4. If a point q lies exactly on the median line, then q belongs to one of the two subsets of the partition, which is shown by turning the median line around q towards the region in which it is included, for sake of clarity. The dashed lines indicate the limiting lines of $region(root)$ associated with the root node of $\mathcal{T}_\theta(Q)$; and the points $\{v_t : 1 \leq t \leq 2k = 6\}$ denote its vertices obtained from the intersections of these limiting lines (Sec. 2.4.2).

The algorithm BUILD-ANGULAR-TREE for construction of $\mathcal{T}_\theta(Q)$ for any given value of $k \geqslant 2$ is shown in Fig. 2.5. In order to accelerate the repetitive procedure of median

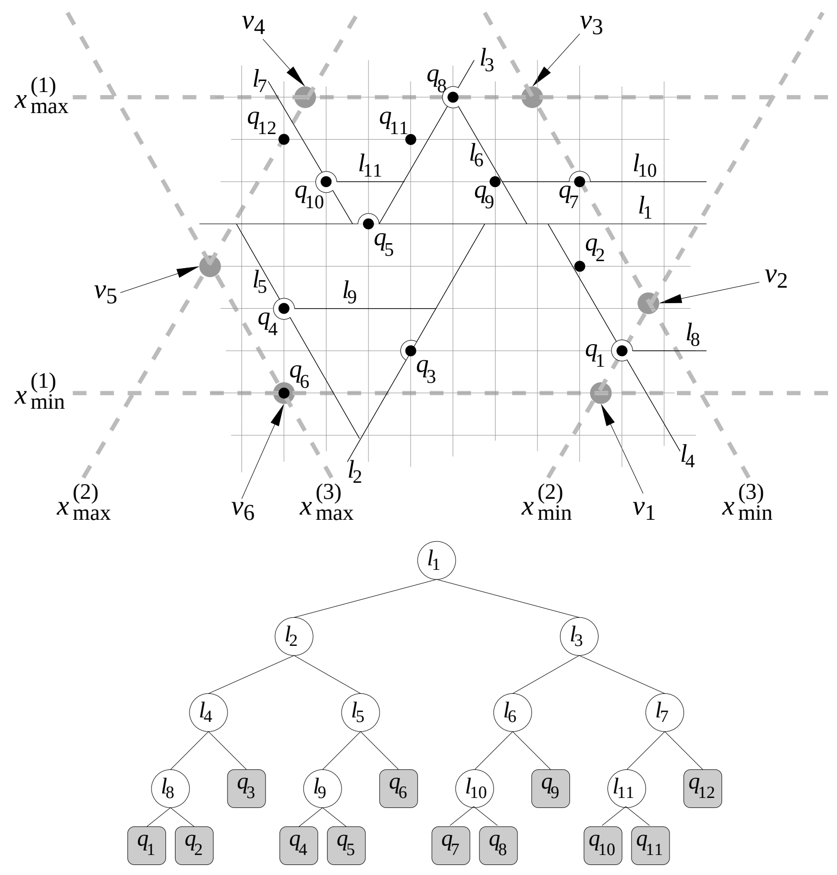

v_4
v_3
$x^{(1)}_{\max}$
$x^{(1)}_{\min}$
$x^{(2)}_{\max}$
$x^{(2)}_{\min}$
$x^{(3)}_{\max}$
$x^{(3)}_{\min}$
v_5
v_2
v_6
v_1
l_1 l_2 l_3 l_4 l_5 l_6 l_7 l_8 l_9 l_{10} l_{11}
q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8 q_9 q_{10} q_{11} q_{12}

Figure 2.4: The angular tree $\mathcal{T}_\theta(Q)$ for $\theta = 120^o (k = 3)$ corresponding to Q containing 12 points on 2D grid.

extraction in Step (5) of the algorithm BUILD-ANGULAR-TREE, we have pre-sorted the points of Q in k ordered lists, namely $\mathcal{L}_\phi(Q)$ for $\phi = 0, \varphi, 2\varphi, \ldots, (k-1)\varphi$, where $\varphi = 180/k$. Each list $\mathcal{L}_\phi(Q)$ stores the points in nondecreasing order of their projections (transformed coordinates) along the line (transformed axis) at an angle of $90^o + \phi$ measured w.r.t. $+x$-axis in the counterclockwise direction. So, the overall time complexity for the entire preprocessing (i.e., pre-sorting) phase is $T_1 = k \times O(n \log n) = O(kn \log n)$.

It may be observed that each of the recursive calls for building the left and the right subtrees (Steps 7 and 8 in Fig. 2.5) has an input set containing at most $\lceil n/2 \rceil$ points, and that all other operations (including the median finding on a pre-sorted list) require constant time. Hence, after preprocessing, the time to construct $\mathcal{T}_\theta$ is given by

```
Algorithm BUILD-ANGULAR-TREE $(Q, d, k)$
Input: set $Q$, current depth $d$, number of directions $k$.
Output: the root of $\mathcal{T}_\theta(Q)$.
Steps:
1.  if $Q$ contains only one point
2.      then return a leaf storing this point
3.  else
4.      $\phi \leftarrow (d \ mod \ k) \times (180/k)$
5.      find the median line $\lambda_\phi$ from $\mathcal{L}_\phi(Q)$
6.      split $Q$ into $Q_1$ and $Q_2$ by $\lambda_\phi$
7.      $\nu_1 \leftarrow$ BUILD-ANGULAR-TREE $(Q_1, d+1, k)$
8.      $\nu_2 \leftarrow$ BUILD-ANGULAR-TREE $(Q_2, d+1, k)$
9.      create a node $v$ storing $\lambda_\phi$
        make $\nu_1$ the left child and $\nu_2$ the right child of $\nu$
10. return $\nu$
```

Figure 2.5: Algorithm for constructing the angular tree $\mathcal{T}_\theta(Q)$ corresponding to the set of points, Q.

$$T_2(n) = \begin{cases} O(1) & \text{if } n = 1, \\ O(1) + 2T_2(\lceil n/2 \rceil) & \text{if } n > 1, \end{cases}$$

which solves to $O(n)$; wherefore, after including the preprocessing time, the total complexity for building $\mathcal{T}_\theta(Q)$ containing n points, is given by $T(n) = T_1(n) + T_2(n) = O(kn \log n) + O(n)$, or, $T(n) = O(kn \log n)$. However, it may be noted, if we do not use any pre-sorting (of the projections of points in Q) for median finding in Step 5 of the algorithm, then each recursive call in Step 5 requires $O(n)$ time, using the worst-case linear time median finding algorithm [Cormen *et al.* (2000)]. Hence, $T_2(n)$ follows the recursive equation $T_2(n) = O(n) + 2T(\lceil n/2 \rceil)$ if $n > 1$, or $T_2(n) = O(n \log n)$, thereby giving total complexity for building $\mathcal{T}_\theta(Q)$ as $T(n) = T_2(n) = O(n \log n)$.

Regarding storage, it is easy to observe that at each step of the algorithm, a region is recursively split into two subregions (subtrees) whose numbers of points differ by at most unity. The height of $\mathcal{T}_\theta(Q)$ is, therefore, bounded by $O(\log n)$, whence the total number of (internal and terminal) nodes is $O(n)$. Hence follows the following theorem on time and

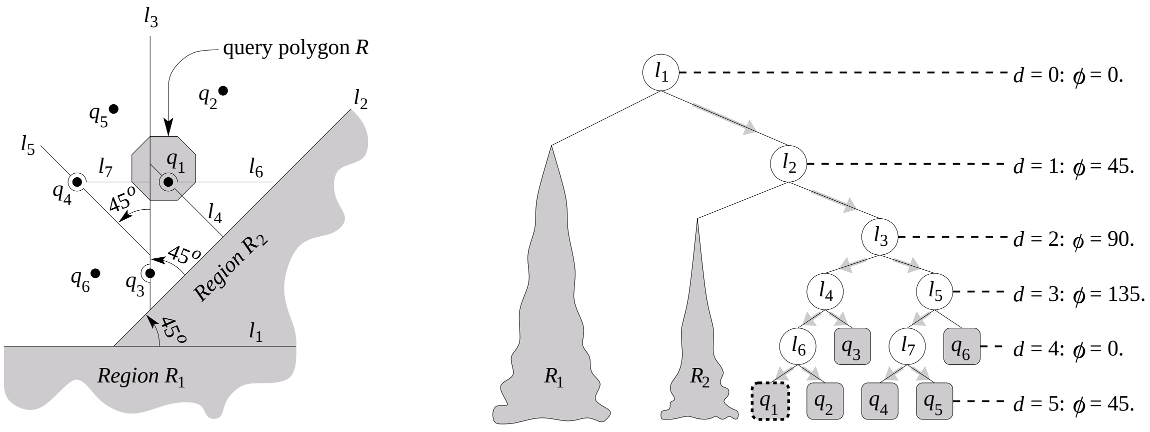


Figure 2.6: An instance of range query with the query octagon R ($k = 4$) in an angular tree $\mathcal{T}_\theta$ corresponding to a set of points, Q.

space complexities of the angular tree.

Theorem 2.4.1 *An angular tree $\mathcal{T}_\theta(Q)$ for a set of points, Q of n points, needs $O(n)$ space, and the tree can be constructed in $O(n \log n)$ time with no preprocessing, or alternatively, in $O(kn \log n)$ time with preprocessing for pre-sorting the points w.r.t. their transformed coordinates.*

For a given value of k, it is evident that the entire process of building the angular tree $\mathcal{T}_\theta(Q)$ needs $O(n \log n)$ time, which is independent of k (and independent of preprocessing, thereof). Hence, in our implementation, we have incorporated the preprocessing phase in building $\mathcal{T}_\theta(Q)$. Without preprocessing, the construction of $\mathcal{T}_\theta(Q)$ needs linear-time median finding algorithm, which is, however, computationally effective only when n is sufficiently large — e.g., when n is 2048 or more as found in our implementation. This is due to large values of the hidden constants associated with the running time of the algorithm having inherent procedural complexities.

2.4.2 Searching in an Angular Tree

For $k = 2$, searching of points would be done in $\mathcal{T}_\theta$ inside a rectangular boundary. Similarly, for $k = 3$ or $k = 4$, a searching has to be performed in a hexagonal region or an octagonal region respectively. Example of an octagonal query ($k = 4$) in $\mathcal{T}_\theta$ is shown in Fig. 2.6, in which the query polygon R is sufficiently small so that there lies at most one point of Q in R. The search starts from the root of $\mathcal{T}_\theta(Q)$, and follows the right path for $d = 0$ and

$d = 1$, and bifurcates at $d = 2$ as the corresponding node (l_3) intersects R indicating that R has intersections with both the regions divided by l_3. At each of the subsequent nodes, depending on whether or not the corresponding line of division intersects R, the decision is taken for traversing the left path and/or the right path from the concerned node. In the leaf-level, q_1 corresponds to the leftmost node and q_5 to the rightmost node at which the respective searches along the leftmost path and along the rightmost path terminate.

The query algorithm takes the root of the $\mathcal{T}_\theta$ and a query range R as input. The query range R is specified by (i) the center $p \in \mathbb{Z}^2$ of the corresponding (regular) polygon, (ii) the perpendicular distance d of an edge of the polygon from its center p, and (iii) the number of edges ($2k$) of the polygon. From these specifications of R, the ordered set of $2k$ real vertices, namely $V_R := \langle v_t := (x_t, y_t) \in \mathbb{R}^2 \rangle_{t=1}^{2k}$, is obtained for low-level description of R, where the first element v_1 in V_R corresponds to the left-bottom vertex of R. Using a recursive procedure, the query process reaches each node (i.e., region) intersected by/contained in R. If the region $region(\nu)$ corresponding to a node ν lies entirely inside R, i.e., if $region(\nu) \subseteq R$, then all the points (leaf nodes) stored in the sub-tree rooted at ν are reported in $O(n_\nu)$ time, where n_ν is the number of points in the corresponding sub-tree. On the other hand, if $region(\nu) \cap R \neq \emptyset$, then the search is carried on along the appropriate subtree of ν, and if $region(\nu) \cap R = \emptyset$, then the search does not proceed down below ν (but, possibly proceeds from some other node ν' of $\mathcal{T}_\theta$). The query algorithm, in short, is given in Fig. 2.7, where the symbols have their usual meanings.

The main test in the query algorithm is for the type of intersection between the query range R and the region corresponding to a node ν. In order to do this test, we compute $region(\nu)$ for each node ν during construction of $\mathcal{T}_\theta(Q)$ using the splitting line at ν and the region of its predecessor node. To start with the root node of $\mathcal{T}_\theta(Q)$, the corresponding region is given by (the ordered set of) k pairs of minimum and maximum coordinates along the (transformed) axes. In other words, if $x^{(e)}_{\min}$ and $x^{(e)}_{\max}$ denote the respective minimum and maximum $x^{(e)}$-coordinates ($1 \leq e \leq k$) for the set of points, Q, as shown in Fig. 2.4, then the region associated with the root of $\mathcal{T}_\theta(Q)$ is given by $region(root) = \left\{ \left[x^{(e)}_{\min}, x^{(e)}_{\max} \right] \right\}_{e=1}^{k}$.

From the above set, we can define the (ordered set of) vertices of $region(root)$ as $V_{root} = \{v_t := (x_t, y_t)\}_{t=1}^{2k}$, where the point of intersection between the lines[1] $x^{(t)}_{\min}$ and $x^{(t+1)}_{\min}$ is v_t ($1 \leq t \leq k-1$); between $x^{(k)}_{\min}$ and $x^{(1)}_{\max}$ is v_k; between $x^{(t)}_{\max}$ and $x^{(t+1)}_{\max}$ is v_t ($k+1 \leq t \leq 2k-1$); and between $x^{(2k)}_{\max}$ and $x^{(1)}_{\min}$ is v_{2k} (Fig. 2.4).

[1]Here we denote the line $x^{(t)} = x^{(t)}_{\min}$ by $x^{(t)}_{\min}$ for simplicity.

```
Algorithm SEARCH-ANGULAR-TREE ($\nu, R$)
Input:   the root of (a subtree of) $\mathcal{T}_\theta(Q)$, and
         a polygonal range $R$.
Output: all points in $\mathcal{T}_\theta(Q)$ lying inside $R$.
Steps:
1.  if $\nu$ is a leaf
2.      then return $\nu$ if it lies in $R$
3.  else if $region(left(\nu)) \subseteq R$
4.      then report all nodes in $region(left(\nu))$
5.      else if $region(left(\nu)) \cap R \neq \emptyset$
6.          then SEARCH-ANGULAR-TREE($left(\nu), R$)
7.      if $region(right(\nu)) \subseteq R$
8.      then report all nodes in $region(right(\nu))$
9.      else if $region(right(\nu)) \cap R \neq \emptyset$
10.         then SEARCH-ANGULAR-TREE($right(\nu), R$)
```

Figure 2.7: Algorithm for searching in the angular tree $\mathcal{T}_\theta(Q)$.

Now, the description of $region(l_r)$ (and of $region(r_r)$) in terms of its vertices can be obtained from the vertices of $region(root)$ as follows. Since each vertex (v_t) in V_{root} is the point of intersection between the limiting lines (say, $x^{(t)}_{\min}$ and $x^{(t+1)}_{\min}$, w.l.o.g.), the $x^{(t)}$- and $x^{(t+1)}$-coordinates of v_t become $x^{(t)}_{\min}$ and $x^{(t+1)}_{\min}$ respectively, which are used to obtain its (x, y) coordinates for computing the other $k-2$ coordinates of v_t. For instance, in Fig. 2.4, the first two $(x^{(1)}, x^{(2)})$ coordinates of v_1 are $x^{(1)}_{\min}$ and $x^{(2)}_{\min}$, and its third $(x^{(3)})$ coordinate is obtained from these two; similarly, for v_2, we have $x^{(2)} = x^{(2)}_{\min}$ and $x^{(3)} = x^{(2)}_{\min}$, etc. It may be noted that the other $k-2$ coordinates of v_t are needed to find the sub-regions of $region(root)$ as discussed next.

When the first median line $x^{(1)}_{\text{med}}$ is used to split $region(root)$, we search in V_{root} for two pairs of consecutive vertices, namely (v_t, v_{t+1}) and $(v_{t'}, v_{t'+1})$, such that $x^{(1)}_{\text{med}}$ lies between $x^{(1)}$-coordinates of (v_t, v_{t+1}) and those of $(v_{t'}, v_{t'+1})$. For example, in Fig. 2.4, the first median line is l_1 and the corresponding pairs of consecutive vertices are (v_2, v_3) and (v_4, v_5). Subsequently, the fact that the median line $x^{(1)}_{\text{med}}$ intersects the two edges (of the polygon $region(root)$) defined by (v_t, v_{t+1}) and $(v_{t'}, v_{t'+1})$ at two points, namely v' and v'' respectively, is used to construct the respective vertex sets for the left and the right sub-regions of $region(root)$ as $V_{l_r} = \{v_1, \ldots, v_t, v', v'', v_{t'+1}, \ldots, v_{2k}\}$ and $V_{r_r} =$

$\{v', v_{t+1}, \ldots, v_{t'}, v''\}$. Continuing this process, we obtain the vertex set V_ν for each node ν of $\mathcal{T}_\theta(Q)$ from the vertex set $V_{\pi(\nu)}$ of its parent node $\pi(\nu)$ by considering the points of intersection of the median line at ν with the two concerned edges of $V_{\pi(\nu)}$.

The low-level description of the query polygon R, as described above, enables checking the type of intersection with $region(\nu)$, which is performed as follows:

Case (i) if all vertices in V_ν lie inside R, then $region(\nu) \subseteq R$;

Case (ii) if no vertex in V_ν lies inside R and no vertex in R lies inside V_ν, then $region(\nu) \cap R = \emptyset$;

Case (iii) if Case (i) and Case (ii) fail, then $region(\nu)$ has partial intersection with R.

In order to ascertain whether a vertex $u := (x_u, y_u) \in V_\nu$ lies inside the polygon R (and to check whether a vertex of R lies inside $region(\nu)$), we check whether u lies to the left of each edge e_t directed from $v_t := (x_t, y_t) \in V_R$ to $v_{t+1} := (x_{t+1}, y_{t+1}) \in V_R$. This is done by simply computing the sign of the determinant

$$D(e_t, u) := \begin{vmatrix} 1 & x_t & y_t \\ 1 & x_{t+1} & y_{t+1} \\ 1 & x_u & y_u \end{vmatrix},$$

since the point u lies to the left of the directed edge e_t if and only if $D(e_t, u) > 0$.

It may be noted that, for a circular range query, the query R is specified by the center (q) and radius (ε) of the query (digital) disc. Therefore, whether a vertex (or a grid point) p lies in R (in the above three cases and also in Step 2 of Fig. 2.7) can be checked in $O(1)$ time by testing whether $round\,(d(p,q)) = \lfloor \sqrt{d^2(p,q)} + 0.5 \rfloor \leq \varepsilon$. Such a simple check is faster and easier w.r.t. the procedure using the signed value of the determinant D as mentioned above, which needs $O(k)$ time when R contains $O(k)$ edges. However, we have to look at D when R is a general convex polygon and not induced by a digital disc.

To estimate the time complexity of searching with a query polygon R of $2k$ vertices and arbitrary size (i.e., an arbitrarily large value of the perpendicular distance d of an edge of the polygon from its center p) in $\mathcal{T}_\theta(Q)$, let $n' (\leqslant n)$ be the number of reported points. Hence, the total time for reporting these n' points after (complete) traversal of the associated subtrees (Steps 4 and 8) of $\mathcal{T}_\theta(Q)$ is $T_1(n) = O(n')$. Further, apart from the traversal of these subtrees, the searching procedure recursively visits some nodes that are not in these subtrees (Steps 6 and 10). Since R has non-empty intersection with $region(\nu)$ corresponding to each such node ν (by the conditions of **if** statements in Steps 5 and 9), the maximum path length (in $\mathcal{T}_\theta(Q)$) for the searching recurrence is given by the maximum number of regions intersected by an edge of R, or equivalently, by a line having any one of the k possible slopes corresponding to k numbers of transformed axes. W.l.o.g., let us consider a horizontal line l for this purpose. Such a horizontal line intersects the regions whose nodes represent horizontal splitting lines in $\mathcal{T}_\theta(Q)$. Clearly, the first horizontal splitting line is the root node (depth $= 0$), the second one is at depth $= k$, the third one at depth $= 2k$, and so on. For instance, for $k = 4$, in Fig. 2.6, the node representing l_1 is the first horizontal splitting line, the nodes representing l_6 and l_7 are the next ones, etc.

Now, at depth $k(\geqslant 0)$, $\mathcal{T}_\theta(Q)$ contains 2^k nodes, each representing a region containing at most $\lceil \ldots \frac{1}{2}\lceil \frac{1}{2}\lceil \frac{n}{2}\rceil\rceil \ldots \text{upto } k\rceil = \lceil n/2^k\rceil$ points. If the horizontal line l lies below the first horizontal splitting line, namely l_1, then in the worst case, l may intersect all the regions below l_1. Similar arguments hold if l lies above l_1. Hence, half of the nodes at depth k, i.e., 2^{k-1} nodes, correspond to maximum number of regions intersected by the horizontal line l. Thus, we have to count the number of intersected regions in the subtrees rooted at these 2^{k-1} nodes recursively (Steps 6 and 10). Also, the regions from (and including) level $= 0$ to (and including) level $= k-1$ that are intersected by l are those corresponding to the root, one of the two children of root, two of the four grandchildren of the root, and so on, which counts to $1 + \frac{1}{2}(2^1 + 2^2 + \ldots + 2^{k-1}) = 1 + 2^{k-1} - 1 = 2^{k-1}$.

Hence, the recursive equation for the number of regions intersected by any line (the time complexity for recursive traversal of $\mathcal{T}_\theta(Q)$, there of) is given by

$$T_2(n) = \begin{cases} O(n) & \text{if } n < k, \\ 2^{k-1} + 2^{k-1}T_2\left(n/2^k\right) & \text{if } n \geqslant k, \end{cases}$$

which solves to $T_2(n) = O(n^{\log_{2^k} 2^{k-1}}) = O(n^{(k-1)/k})$. For an arbitrarily large value of k, therefore, we get $T_2(n) = O(n)$. Hence, the total searching time in $\mathcal{T}_\theta(Q)$ is given by $T(n) = T_1(n) + T_2(n) = O(n^{(k-1)/k} + n')$, or $T(n) = O(n^{(k-1)/k})$ if $n' \leqslant O(n^{(k-1)/k})$, wherefore, for an arbitrarily large size polygonal query region with an arbitrarily large number of vertices, we get $T(n) = T_1(n) + T_2(n) = O(n + n') = O(n)$. The above findings, therefore, are put together in the following lemma.

Lemma 2.4.2 *The worst-case polygonal range query with an arbitrary regular polygon in an angular tree $\mathcal{T}_\theta(Q)$, representing a set of points, Q of n points, needs $O(n)$ time.*

For searching with small polygonal range queries where a query polygon always contains (and reports) at most one point of Q (which is found in many a practical application, e.g., fingerprint matching [Bhowmick and Bhattacharya (2005a)]), however, the recursive searching procedure would proceed along a single path from the root to the leaf node. The leaf node where the search terminates corresponds to the point that actually lies inside the (query) polygon. This is owing to the fact that each edge of the polygon, being suitably small in length, would intersect just one or two regions at each level of $\mathcal{T}_\theta(Q)$, thereby curtailing the recursive searches (Steps 6 and 10) in the search algorithm (see, for example, the illustration in Fig. 2.6). To be precise, there will be at most $2k$ distinct regions (corresponding to $\mathcal{T}_\theta(Q)$) with non-empty intersections with the $2k$-gonal query region R, provided R is sufficiently small in size.

For such a query polygon with proper adjustment of its size, therefore, the above search time complexity does not reflect the actual run time. Even if the query polygon contains at most c points, where c is a small constant, then the recursive searching procedure would proceed at most along c paths from the root to the leaf node. Thus, here the above bound for $T_2(n)$ is given by $c \times 2k \times O(\log n) = O(k \log n)$, which reduces to $O(\log n)$ for a prescribed value of k, and $T_1(n)$ reduces to $O(1)$ (since $n' \leqslant c$). Hence the total search time (for a single point of Q) drops down to $T(n) = O(1) + O(\log n) = O(\log n)$, which gives the following lemma:

Lemma 2.4.3 *A polygonal range query in an angular tree $\mathcal{T}_\theta(Q)$, representing a set of points, Q of n points, needs $O(\log n)$ time with the hypothesis that the size of the query polygon is small enough so that it contains at most one point or a constant number of points of Q.*

In the APSPM algorithm (Sec. 2.4.3) and in the subsequent discussions, we have assumed that the circular range query is evoked with an appropriately small size of the query range so that the number of points reported against the query is a small constant. This in turn, ensures $O(\log n)$ time for performing each query in the angular tree.

2.4.3 APSPM Algorithm Using Circular Range Query

For approximate matching of the pattern set P containing m points with the background set Q containing n points, we construct the angular tree $\mathcal{T}_\theta(Q)$ corresponding to Q, which is used for circular/polygonal range query for each point of P. To negotiate the correspondence among the points of P and Q, we adopt an alignment policy that speeds up the matching process. We pick the point pair in P, having the largest (Euclidean) distance (in P), to be anchored near and aligned along a point pair in Q having its distance matching within a tolerance of ε. Such an act of choosing the largest distance, failure of which is followed by repetition(s) with the next largest distance(s), has experimental bearing on the frequency of distances in a (discrete/real) set of points — the test results being shown in Sec. 2.5, and also has a theoretical basis as shown below.

Let S be a finite subset of the Euclidean metric space $\mathbb{R}^2$. Let a and b be any two distinct points in S. Let $\mathbf{L}$ be the (real) line segment joining a and b, and having length L. Let a' and b' be two distinct points lying on the line segment $\mathbf{L}$. Hence, if the points a' and b' do not simultaneously coincide with the points a and b, then the length L' of the line segment $\mathbf{L}'$ joining a' and b' follows the relation

$$0 < L' < L. \tag{2.1}$$

Now there may exist infinitely many such point pairs (a', b') on the line segment $\mathbf{L}$, each of which having distance equal to L'. That is, given two real-valued distances, namely L and L', satisfying Eqn. 2.1, we get infinitely many point pairs of distance L' *contained by* a line segment of length L that lies in S. Further, since there may exist multiple (infinitely many, possibly) instances of line segments having length L in S, we might get multiple assemblages of infinitely many smaller line segments of length L' *contained by* the former line segments (i.e., those with length L).

Apart from the distance L (i.e., line segments of length L) in S, there may exist only two types of distances in S, which may be classified into two disjoint sets given by

$$\underline{D} = \{\underline{L} : \underline{L} < L\} \text{ and } \overline{D} = \{\overline{L} : \overline{L} > L\}$$

so that $\underline{D} \cup \{L\} \cup \overline{D}$ contains all possible distances in S. No distance $\underline{L}$ in $\underline{D}$ will *contain* the distance L in S, because a line segment of length L cannot be made to lie on a line segment of smaller length $\underline{L}$. However, some of the distances in $\underline{D}$ may *contain* the distance L' in S so far as the former distances are greater than the latter (i.e., L'). Thus, the set $\underline{D}$ may contribute distance L' but never contributes the distance L.

Regarding the set $\overline{D}$, each of its distances being greater than L, each distance $\overline{L}$ in $\overline{D}$ always contains (infinitely many instances of) the distance L, each of which, in turn, contains (infinitely many instances of) the distance L'. Therefore, the frequency of distance L' contributed by $\overline{D}$ is always greater than that of the distance L contributed by $\overline{D}$. In particular, the factor by which frequency of L' exceeds the frequency of L depends on the ratio $L : L'$ — higher the ratio, greater will be the frequency of L' compared to that of L contributed by the set $\overline{D}$.

As a result, in totality, the (continuous) probability density of the distance L' always exceeds that of the distance $L(> L')$ in the finite set S. The above arguments will also hold true for the 2D digital space, although the aforesaid monotonically increasing nature of frequency with decreasing distances corresponding to the continuous case would not satisfy in the discrete case due to the anisotropic nature of the isothetic grid lines. Hence, we state the following lemma, whose bearing with the reality is evidenced by few sets of plots shown in Fig. 2.9 (Sec. 2.5).

Lemma 2.4.4 *In a set of real or discrete points, the frequency of distances in the interval* $[L - c, L + c]$ *decreases with increase in* L*, for an appropriate constant* c*.*

```
Algorithm APSPM-Angular-Tree $(P, Q, \varepsilon)$
Input: sets of points, $P, Q$, and error tolerance $\varepsilon$.
Output: whether or not $P$ matches $Q$.
Steps:
  1. Determine (/decide) $k$ from the given value of $\varepsilon \in \mathbb{Z}^+$ (Sec. 2.3).
  2. Construct $\mathcal{T}_\theta(Q)$ for the background set $Q := \{q_u\}_{u=1}^{n}$, with $k$ from Step 1
     (Sec. 2.4.1).
  3. Compute the Euclidean distances for all distinct pairs of points in $Q$,
     and store them in the (non-increasing) ordered list, namely $\mathcal{L}(Q) :=$
     $\langle d(q', q'') : q' \in Q, q'' \in Q \rangle$.
  4. Similarly, construct the ordered list $\mathcal{L}(P) := \langle d(p', p'') : p' \in P, p'' \in P \rangle$ in non-
     increasing order of the distances in $\mathcal{P}$.
  5. Perform the one-dimensional range query with interval $[d_P - \varepsilon, d_P + \varepsilon]$, with $d_P$
     starting from the first (maximum) distance in $\mathcal{L}(P)$, on $\mathcal{L}(Q)$.
  6. For each matching distance $d_Q \in [d_P - \varepsilon, d_P + \varepsilon]$ in $\mathcal{L}(Q)$, find the necessary
     transformation $T$ in order to align the point pair in $P$ (corresponding to $d_P$)
     with the matching pair in $Q$ (corresponding to $d_Q$), and apply $T$ on all points in
     $P$ to obtain $T(P)$.
  7. Apply circular/polygonal range query in $\mathcal{T}_\theta(Q)$ for each point in $T(P)$ to find
     the total number of matching points, namely $\mu(T(P), Q)$, in $Q$.
  8. If $\mu(T(P), Q)/(\min(m, n)) > \varrho$, then a match exists between $P$ and $Q$ under the
     transformation $T$. Otherwise, next matching distance $d_Q$ obtained in Step 5 is
     tried for, until a match is found or all distances in $\mathcal{L}(P)$ are exhausted.
```

Figure 2.8: Algorithm on APSPM using angular tree.

The theoretical justification being given above, the major steps of the algorithm have been given in Fig. 2.8. In this algorithm, we consider $T(P)$ to be a valid match with Q (Step 8), provided the number of points in correspondence between $T(P)$ and Q exceeds a certain factor, namely ϱ, of the maximum number of points (i.e., $\min(m, n)$) possible to produce an ideal match between them. Evidently, a match found with a small value of ϱ for two given sets, P and Q, signifies a slackened approximation, whereas that with a large value of the same corresponds to a tight approximation. It may be mentioned here that all the results shown in Sec. 2.5 are with $\varrho = 1$.

The time complexity of the proposed algorithm is influenced by the number of failures for the misalignments arising out of the mismatching transformations in Step 6. For each such transformation T, all points in P are subject to polygonal range queries in $\mathcal{T}_\theta(Q)$, which incurs $O(\log n)$ time per query, producing $O(m \log n)$ time for m queries corresponding to P. Each possible alignment of a point pair $\langle p', p'' \rangle$ of P with a matching pair $\langle q', q'' \rangle$ of Q involves $O(\varepsilon^2)$ possible positions of p' in the (ε-)neighborhood of q' (i.e., inside the digital disc with radius ε and center q'). Each such position of p', in turn, allows $O(\varepsilon)$ positions of p'' in the neighborhood of q'' (which are the grid points of the digital disc, having radius $d(p', p'')$ and center p', lying inside the neighborhood of q''). This involves $\kappa(\varepsilon) = O(\varepsilon^3)$ possible orientations of $p'p''$ for (approximate) alignment with $q'q''$. Here, $\kappa(\varepsilon)$ is a cubic function of ϵ, which has not an effect on the asymptotic time complexity when ε is considered to be a constant [Li and Klette (2007)]. Thus, the total complexity for all possible alignments for a given (approximately matching) pair of point pairs is $\kappa(\varepsilon) \times O(m \log n)$.

Since there are $O(n^2)$ point pairs in P and each of these point pairs may have to be approximately aligned with any of the $O(m^2)$ possible point pairs of Q, the worst case time complexity for finding out an approximate match (successful or unsuccessful) between the sets of points, P and Q, in the 2D digital space is given by $T(m, n, \varepsilon, k) = O(m^2 n^2) \times O(m\varepsilon^3 \log n) = O(\varepsilon^3 m^3 n^2 \log n)$, which reduces to $O(m^3 n^2 \log n)$ for a fixed value of ε. Hence, we have the following theorem on the time complexity for approximately matching a set of 2D points P with another set Q.

Theorem 2.4.5 *For ε-approximation matching of P containing m points with Q containing n points in 2D digital space using angular tree $\mathcal{T}_\theta(Q)$, the worst-case time complexity is given by $\kappa(\varepsilon)O(m^3 n^2 \log n)$ for a given value of the approximation parameter ε.*

Table 2.1: Comparison of existing methods with the proposed APSPM algorithm

Method	Time	Space	APSPM	Difficulty in implementation
string matching[a]	$O(n \log n)$	$O(n)$	no	simple
distance alignment[b]	$O(mn^{4/3} \log n)$	$O(n^3)$	no	moderate
ε-alignment[c]	$O(n^8)$	$O(n)$	yes	complex
Hausdorff metric[d]	$O\left((m+n)^6 \log(mn)\right)$	$O(m+n)$	yes	very complex
proposed method	$O(m^3 n^2 \log n)$	$O(m^2+n^2)$	yes	moderate

[a][Atkinson (1998), de Rezende and Lee (1995)]
[b][Bishnu *et al.* (2003)]
[c][Alt *et al.* (1988), Arkin *et al.* (1992), Erafat and Itai (1996)]
[d][Huttenlocher and Rucklidge (1993), Mount *et al.* (1998)]

From the above results and discussion, it is clear that the angular tree and the APSPM algorithm proposed here are having reasonable complexities, and are readily realizable for relevant applications involving sets of points in a digital plane. We have implemented and tested the algorithm along with the angular tree, which have produced encouraging results, as shown in Sec. 2.5. A comparative study of some existing point set pattern matching algorithms with the proposed algorithm using angular tree is given in Table 2.1, which indicates its readiness and usability in the digital plane.

2.5 Experimental Results

2.5.1 Polygonal Approximation of Digital Circles

As shown in Chapter 7, each of the digital circles with radii from $\rho = 1$ to $\rho = 10$ has a regular polygonal enclosure. Further, for $\rho = 12-16, 20-25, 32, 33, 40, \ldots$, the corresponding digital circles possess regular polygonal enclosures as shown in Fig. 7.7 and Fig. 7.11. In case of a digital circle for which the regular polygonal enclosure is not possible ($\rho = 11, 17-19, 26-31, 34-39, \ldots$), the error associated with an approximate polygonal enclosure can be decreased by increasing its number of vertices, as exhibited in Fig. 7.9. Thus, depending on the related application, the error can be made to lie below the desired limit by choosing an appropriate number of vertices of the polygonal enclosure.

2.5.2 Performance of Circular Range Query.

Given a set of points, $Q \subset \mathbb{R}^2$, containing n points and given a Euclidean distance d, it has been shown [Szekely (1997)] that the number of distinct point pairs having distance d may be as high as $\lceil n^{4/3} \rceil$. Our experimental results, however, reveal that the maximum observed frequency of a particular distance in $\mathbb{Z}^2$ (or, for a better realization in the 2D plane, the maximum frequency of a particular range of distances) is far short of the theoretical value.

We have generated 1000 synthetic sets of points in 2D digital space, each set containing 10 to 500 randomly located grid points, and have studied the change in frequency versus distance in these sets of points. The following three parameters are considered by us to examine the structural/geometric properties of sets of digital points:

(i) n = number of points (in Q), $10 \leqslant n \leqslant 500$;
(ii) $d_{\min}$ = minimum Euclidean distance between two distinct points, $5 \leqslant d_{\min} \leqslant 25$;
(iii) A = the discrete orthogonal domain (minimum-area finite region) containing all points, $50 \times 50 \leqslant A \leqslant 500 \times 500$.

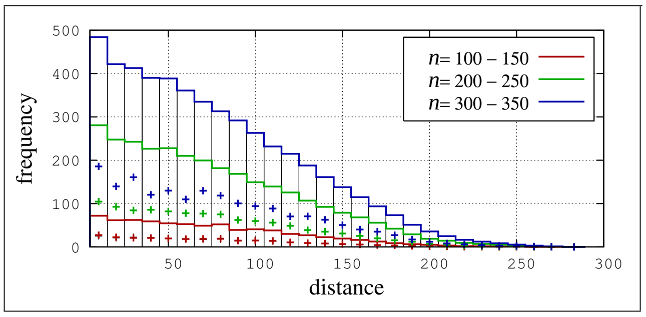
For a few sample sets of points, the trend of variation of (average) frequency and the corresponding standard deviation versus (average) distance have been shown in Figs. 2.9(a)–(c). From these plots, it is evident that, for a given set of points, the number of occurrences of a particular distance in the set of points decreases with the increase in the distance, and vice versa. This is instead a practical confirmation of the theoretical interpretation of the frequency pattern of point-pair distances given in Sec. 2.4.3. In the matching algorithm, therefore, we select the point pair having the maximum distance (since its frequency is likely to be very small) from the pattern set as the first matching distance (the anchoring distance), for applying the alignment scheme (Step 5 of the algorithm in Sec. 2.4).

2.5.3 Execution Time

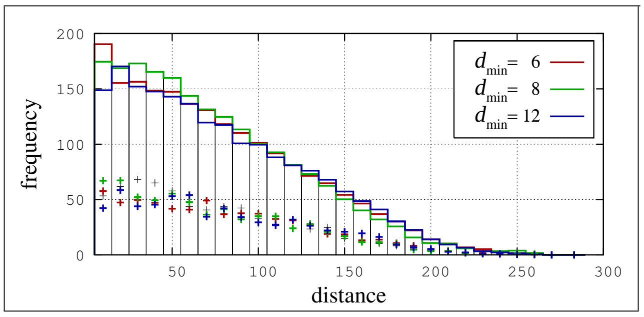
Given a prescribed value of ε for ε-approximation match, the run-time of the proposed APSPM algorithm depends upon two parameters:

(i) the number of points m in the pattern set P, and
(ii) the number of points n in the background set Q.

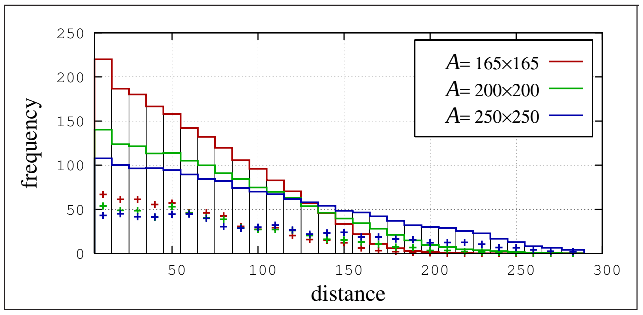
Varying these two parameters (i.e., m and n), we have obtained two sets of plots — one for m and the other for n — illustrating the variation of CPU time versus the anchoring



(a) Frequency vs. distance of point pairs for n varying in three intervals ($A = 200 \times 200$, $d_{\min} = 5$).



(b) Frequency vs. distance of point pairs for three different values of $d_{\min}$ ($A = 200 \times 200$, $n = 200$).



(c) Frequency vs. distance of point pairs for three different sizes of A ($n = 200$, $d_{\min} = 10$).

Figure 2.9: Frequency distribution of distances (shown in uniform intervals of size 10) for different values of n, $d_{\min}$, and A. The histogram profiles show the average frequencies, and the points indicate the respective standard deviations.

distance. These two sets of plots are shown in Fig. 2.10(a), (b). The algorithm has been implemented in C on SunOS Release 5.7 Generic of Sun_Ultra 5_10, Sparc, 233 MHz.

Apart from testing the variation of CPU time with m and n, we have also considered the dimension k of the angular tree $\mathcal{T}_\theta(Q)$ defined on Q, and that its effect on the CPU time is almost negligible is evident from the plots shown in Fig. 2.10(c).

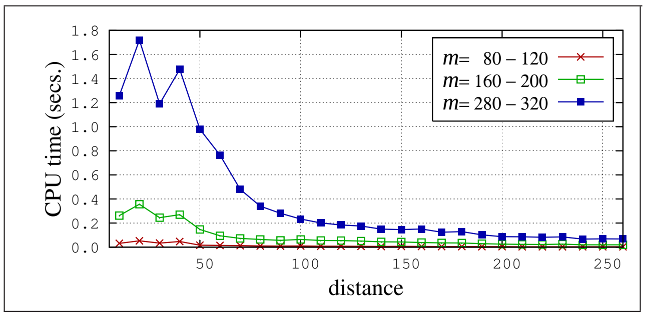
From the above plots, it may be concluded that the running time of the proposed APSPM algorithm increases with the decrease in the distance of the selected point pair for anchoring. Hence, instead of selecting the initial point pair randomly from P, first we select the pair that has the maximum distance in P; in the case it does not yield a successful match, we choose the pair(s) with the next maximum distance, and so on. Such a policy of selecting the furthest point pair from the pattern set for anchoring with some suitable point pair in the background set reduces the time requirement to a great extent in real-time matching applications, which is reflected in the plots of Fig. 2.10.

2.6 Conclusion

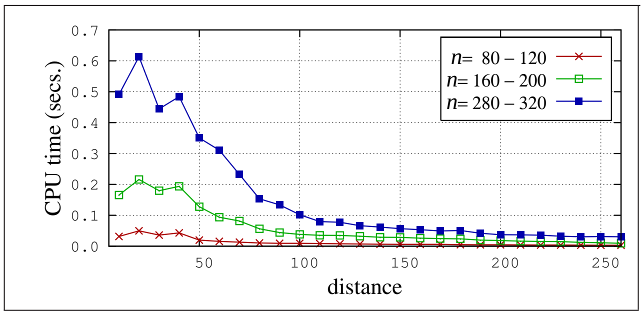
We have shown how a digital disc can be substituted by an appropriate regular polygon so that a (digitally) circular range query in $\mathbb{Z}^2$ can be efficiently replaced by a polygonal range query. Such a query for each point in the pattern set P can be performed efficiently on the background set Q by using an appropriate data structure, namely the angular tree $\mathcal{T}_\theta(Q)$, defined on Q. To sum up, the primary aspects of this chapter are as follows.

Replacing a circular range query by a polygonal range query is the major contribution of this work. Most of the existing approaches rely on orthogonal range query, whereas, here a (regular) polygonal approximation of a circular query in the digital plane is computed. This ensures improved matching of a point in the digital pattern set with the circular neighborhood of a similar point in the digital background set. Further, in the case of a small disc, its polygonal approximation is almost error-free, whereas for a large disc, the approximation incurs a very small error. The associated error can also be estimated easily from the level of approximation.

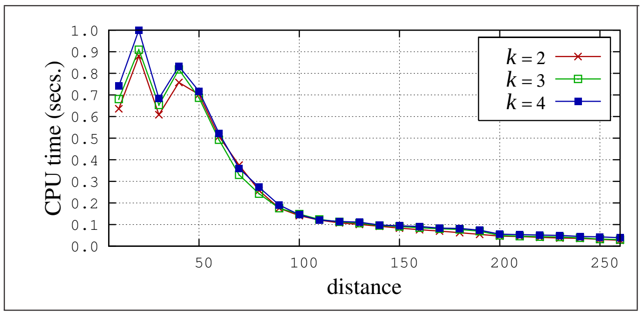
Regarding polygonal partitioning of points, we have proposed an efficient search tree in which the 2D points are stored in accordance to their distribution in the digital plane. The circular neighborhood of a point is defined by the grid points of the corresponding digital disc with the concerned point as center, which are finite (and small for a small disc) in number. Hence, the approximate matching of points in the digital plane can be nicely implemented by using an angular tree that effectively limits the difficult circular



(a) CPU time vs. anchoring distance for different values of m ($n = 200, k = 4$).

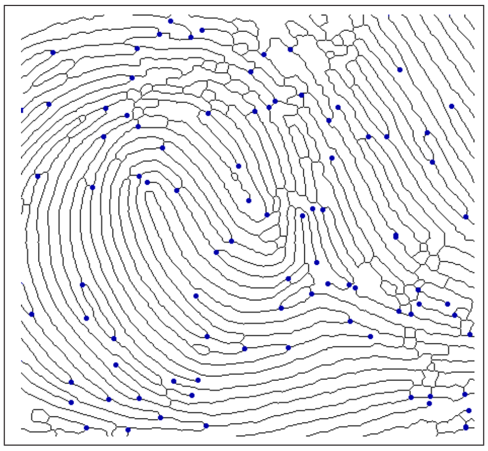


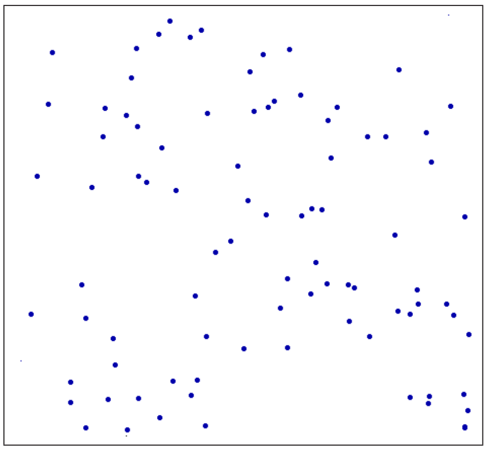
(b) CPU time vs. anchoring distance for different values of n ($m = 200, k = 4$).



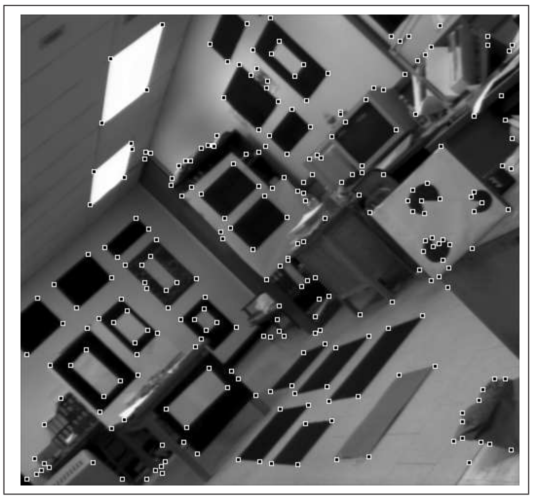
(c) CPU time vs. anchoring distance for different values of k ($m = 80 - 320, n = 80 - 320$).

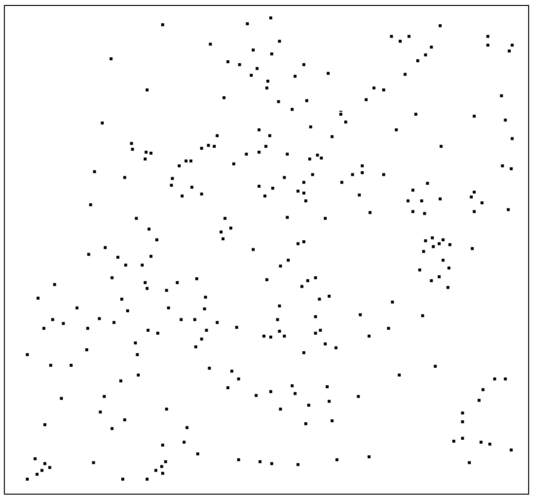
Figure 2.10: CPU time versus anchoring distance (shown in uniform intervals of size 10) in case of successful matching of the set of points, P, containing m points with the set of points, Q, containing n points for different values of m, n, k ($A = 200 \times 200, \varepsilon = 6$).


Minutiae detected after analysis of the ridge lines in a fingerprint image.


The set of digital points formed by the minutiae.

(a) The set of digital points in a fingerprint identification system.


Corners detected in a sample gray scale image.


The set of digital points formed by the corners.

(b) The set of digital points in an application where corners are considered.

Figure 2.11: Two instances of applications where the set of digital points plays a crucial role.

query to an easier polygonal query. Keeping in mind the trade-off with space and time complexities, the angular precision (and the error, thereof) can be controlled by varying the specification of the query polygon, depending on the application.

It has been already made apparent in various works on processing and applications of sets of digital points that partial approximate matching of sets of points in digital images is really a captivating problem, and that such a matching algorithm has numerous applications in the digital paradigm. In Fig. 2.11, we have furnished two typical instances of sets of points in the digital plane — one related with fingerprint identification system [Candela *et al.* (1995), Maltoni *et al.* (2003)] and the other related with corner points in digital images [Argyros *et al.* (2001), Gu and Tjahjadi (1999), Mohanna and Mokhtarian (2002)].

The proposed approach may be further studied for finding the general solution of the APSPM problems in higher dimensions. The possibilities lie in deriving the regular (convex) polytope that would replace the regular 2D polygonal range query, and in constructing the (generalized) angular tree based on median hyperplanes instead of 2D median lines. The allied procedures and associated data structures, of course, would invoke higher complexities, both in terms of developing theory and subsequent implementation.

Chapter 3

Minutiae in a Fingerprint Image as Geometric Points

All err the more dangerously because each follows a truth. Their mistake lies not in following a falsehood but in not following another truth.

Blaise Pascal
In W. H. Auden and L. Kronenberger, The Viking Book of Aphorisms (New York, 1966)

3.1 Introduction

Fingerprints, produced by the ridge and valley patterns on the tip of the fingers, have been used for biometric authentication for quite a long time [Galton (1892)]. Owing to their uniqueness and immutability [Lin *et al.* (1982)], coupled with easy acquisition procedure, fingerprints provide the most widely used biometric features till today [Maltoni *et al.* (2003)]. In the recent years, apart from criminal identification extensively used by law enforcement agencies, fingerprint verification has become more popular in day-to-day civilian applications, such as access control, financial security, employee identification, verification of firearm purchasers, driver license applicants, etc. In the past, fingerprint verification was performed manually by professional fingerprint experts. However, the manual matching of fingerprints is very tedious, time consuming, and expensive. A fingerprint image database may contain as high as several million records, thereby making the manual fingerprint verification an intractable task.

In order to ensure a much faster and efficient fingerprint matching process, Automatic Fingerprint Identification Systems (AFIS) have evolved in recent times. Most of them are based on minutiae matching. Minutiae, also called Galton's characteristics [Galton (1892)], are points of discontinuities of ridgelines in a fingerprint pattern. The American National Standards Institute (ANSI) has proposed a classification of minutiae based on four classes:

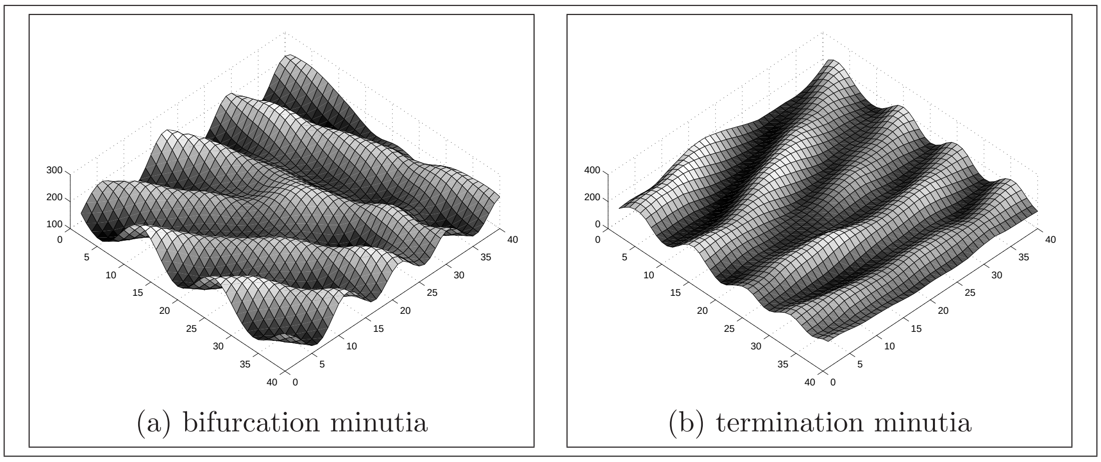


Figure 3.1: Minutiae in the gray-scale topography of a fingerprint image.

terminations, *bifurcations*, *trifurcations* (or, *crossovers*), and *undetermined* [ANSI (1986)]. In practice, however, most of the AFIS follow the two-class minutiae classification, namely termination and bifurcation, used by the Federal Bureau of Investigation (FBI) [Wegstein (1982)]. Since in a fingerprint image, trifurcation and undetermined minutiae, if present, are very few in number, we have adopted the model used by the FBI, and have not considered a trifurcation minutia or an undetermined minutia as a valid minutia. The bifurcation and termination minutiae, located in two small regions of size 40×40 pixels each, of a gray-scale fingerprint image having 500 dpi resolution are shown in Fig. 3.1. These two regions have been carefully selected from a noise-free portion of an image to have a better understanding of how a bifurcation minutia or a termination minutia looks like in the gray-scale image topography. In many situations, however, the minutiae are located in a noisy area when it becomes difficult to recognize them.

In a fingerprint identification system, the fingerprint image is captured by some inking method or a sensor. The acquired image often contains unclean patterns or noise caused by under-inking, over-inking, wrinkles, scars, uneven pressure at fingertip, limitations in the digitization system, etc. Extraction of valid minutiae that characterize a fingerprint image is a primary task before starting a matching process. The minutiae extraction can be done either on a gray-scale image or on a binary image. Whatever may be the technique, the set of extracted minutiae may serve as a distinct feature characterizing the fingerprint image, and can be finally used in the minutiae matching process for fingerprint identification [Farina *et al.* (1999), Kovács-Vajna (2000), Maio and Maltoni (1997), Mehtre and Murthy (1986), Pernus *et al.* (1980), Jain *et al.* (1997)].

Minutiae detection directly from a gray-scale fingerprint image was considered by Maio and Maltoni (1997). The basic idea is to trace the ridgelines on the gray-scale image by "sailing" according to certain directional features. In this scheme, no binarization or thinning is required. Instead, a Gaussian mask for regularizing the uneven "volcano silhouette" of the transverse section of a ridge is used to locate the local maximum corresponding to the ridge center. Computation of local ridge direction is done at every step on each ridgeline to traverse successfully along the ridge centerline, except when, although very rarely, the ridge has excessive bending. The set of minutiae detected in this approach is then passed through three filters to remove the invalid minutiae, such as low contrast minutiae, pairs of close termination minutiae, and bifurcation minutiae crowded in a small neighborhood.

Both the stages of minutiae extraction and minutiae matching have been described by Jain *et al.* (1997) for an online fingerprint verification system. For minutiae extraction, an improved algorithm is implemented by Ratha *et al.* (1995) for processing an input gray-scale fingerprint image captured with an online ink-less scanner. A new hierarchical method is shown for estimation of local orientation field of flow patterns, followed by a segmentation algorithm to locate the region of interest from the fingerprint image. Binarization is done by convolving the fingerprint image with two masks, adapted to the local ridge width, to accentuate the local maximum gray-level values along the direction normal to the local ridge direction. From the binarized ridge map, the holes and speckles, arising due to the noise present in the input image, are removed before ridge thinning for efficient minutiae extraction. The thinned ridge map undergoes a smoothing procedure to remove spikes and to join broken ridges. A final refinement, based on the structural information, is done to eliminate the spurious minutiae, viz., clustered minutiae in a small region, and two close minutiae facing each other. For each surviving minutia, its coordinates, orientation (i.e., local ridge orientation of the associated ridge), and the associated ridge are recorded for matching purpose. The matching procedure consists of two stages, namely, the alignment stage and the matching stage. In the alignment stage, transformations such as translation, rotation, and scaling between an input image and a database template are estimated in order to make the input image aligned with the template minutiae according to the estimated parameters (coordinates, orientation, and associated ridge). In the matching stage, both the input minutiae and the template minutiae are converted to polygons in polar coordinate system and an elastic string matching algorithm is used to match the resulting polygons.

In the fingerprint minutiae extraction method developed by Farina *et al.* (1999), a local ridge distance map has been derived from the skeleton image. The local ridge distance map captures the average ridge distance in each region of the image [Kovács-Vajna *et al.* (2000)]. In this method, the skeleton image is processed pixel by pixel to find the number of outgoing branches that indicate whether the candidate pixel is a minutia or not. This is followed by (i) pre-filtering to delete one minutia from a pair of minutiae lying close to each other; (ii) skeleton enhancement or ridge repair to identify ridge breaks, eliminate bridges, spurs and short ridges; and finally, (iii) removal of islands and validation of bifurcations and end-points. The final valid set of minutiae is classified as either "highly reliable" or "less reliable".

He *et al.* (2003) developed a fingerprint image enhancement and matching algorithm, which is a modification over the method used by Jain *et al.* (1997), and is divided into two phases, off-line and on-line. In the off-line phase, a fingerprint image is acquired, enhanced using orientation fields of ridge directions. Thereafter, features of the fingerprint in terms of minutiae coordinates, its orientation, and relation of the minutiae to some points on the associated ridge are extracted and stored in a database as a template. In the next on-line phase, a fingerprint is acquired, enhanced and the same features of the fingerprint are extracted, fed to a matching model and matched against template models in the database. The matching phase is akin to the method by Jain *et al.* (1997), excepting three aspects. The difference is in the method of alignment, use of ridge information in the matching process, and the use of a variable bounding box that is more robust to non-linear deformations between two fingerprints.

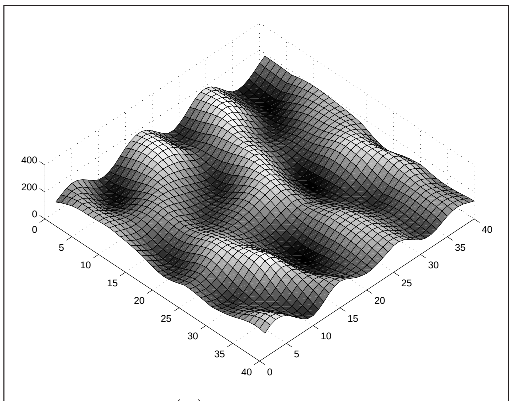
Another approach to speed up fingerprint identification problem is the use of indexing. An indexing algorithm, based on the features of triangles formed by the triplets of minutiae, and its performance on two different data sets in a black-box approach have been reported by Bhanu and Tan (2003). The triangle features that are used are its angles, handedness, type, direction, and maximum side. Experimental results on live-scan fingerprint images of varying quality and NIST special database 4 show that the indexing approach efficiently narrows down the number of candidate images in the presence of translation, rotation, scale, shear, occlusion, and clutter. Thus, the indexing technique significantly reduces the number of hypotheses to be considered for the verification algorithm. In other words, in a complete fingerprint recognition system, an indexing technique can be used as front-end processing, which would be then followed by a back-end verification processing.

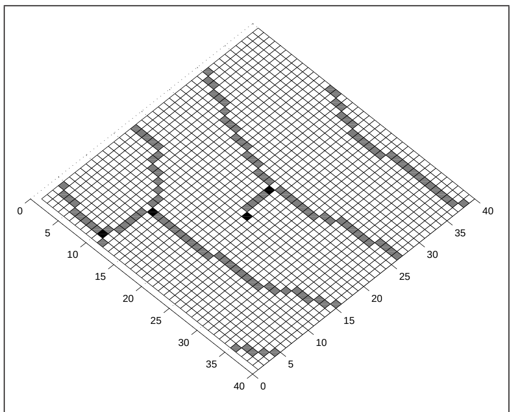
Apart from minutiae-based fingerprint matching, there exist other matching algorithms that largely depend on the ridge and valley topography of a fingerprint image. Traditional minutiae-based methods suffer from the following shortcomings: (i) they do not fully utilize a significant component of the rich discriminatory information available in the ridge and valley structure of fingerprints; and (ii) they cannot produce a quick solution for two fingerprint images containing different number of unregistered minutiae points [Jain *et al.* (2000)]. In a procedure by by Jain *et al.* (2000), a bank of Gabor filters is used to capture both local and global details in a fingerprint as a compact fixed length "FingerCode". This scheme tessellates an image to extract its FingerCode, which is the ordered enumeration of the local features contained in the tessellated sectors; the Euclidean distance between two FingerCodes is used for matching. This has been designed with an objective of computationally attractive matching and indexing. In the matching scheme, the concept of scores in the form of vectors is used to express the degree of matching between two fingerprints. Similarly, Willis and Myers (2001) used the total image, or a better representation thereof, for recognition of low-quality fingerprints. After necessary smoothing and enhancing of imperfect images by a threshold FFT technique, valid minutiae are detected to find a reference point by computing a weighted centroid of all valid minutiae and ridge pixels. The reference point is used for a wedge ring overlay minutia detector, and finally, a number of statistical and neural network classifiers are tested to classify the relevant feature vectors for the recognition task. Ceguerra and Koprinska (2002) follows another approach, which also combines local as well as global features of a fingerprint image by integrating minutiae and shape signatures that are used in a neural network for final recognition.

3.2 Preliminaries

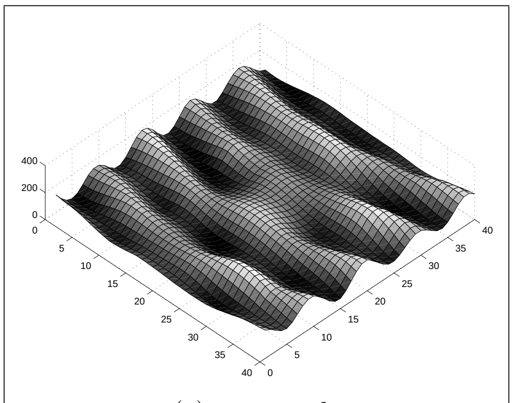
A fingerprint image essentially consists of a set of minutiae on the xy plane. Minutiae are the terminations and bifurcations of ridgelines in a fingerprint image. The ridgelines, appearing in the foreground of the gray-scale topography, are separated by valley lines appearing in the background. In a fingerprint image, there exists a striking duality in the sense that the valley lines also have minutiae (terminations and bifurcations) and flow patterns similar to the ridgelines [Hrechak and McHugh (1990), Hung (1993)]. The ridge and valley characteristics, such as ridge and valley flow directions, inter-ridge and inter-valley distances, ridge and valley breaks, etc., are very useful properties that indicate the validity criteria of a minutia detected by any algorithm. These parameters have been used extensively in a number of earlier works. For enhancing a gray-level fingerprint image, orientation of ridges is used for designing a filter [O'Gorman and Nickerson (1989)], and for using directional images [Mehtre *et al.* (1987)]. In the work by Hung (1993), ridge enhancement is done based on ridge directions, and noise removal and pattern purification are performed with the help of both ridge and valley characteristics.

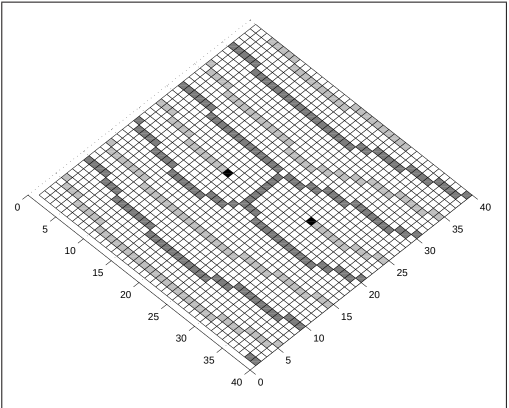
A gray-scale fingerprint image often undergoes binarization, followed by thinning, in the preprocessing stage, in order to extract the minutia points [Bishnu *et al.* (2002, 2006a), Farina *et al.* (1999), Jain *et al.* (1997)]. In the process of binarization and thinning, several ridge deformations, such as spurs, bridges, short ridges, loops, ridge breaks, become prominent that give rise to false minutiae. These undesired spurious elements in the ridge


(a) gray-scale


(b) binary and skeletonized

A spur in a fingerprint image.


(c) gray-scale


(d) ternary and skeletonized

A bridge in a fingerprint image.

Figure 3.2: Magnified views of a spur and a bridge in a fingerprint image.

skeleton owe their origin to the noise present in the original gray-scale image. A spur originating from a point p on a ridge gives rise to a false minutia at q. Among the three branches incident at p, only two branches are aligned while the direction of the third branch that corresponds to a spur is generally different. Moreover, the length of a branch, if forming a spur at p, is within some specified magnitude that helps to identify it as a spur [Bhowmick *et al.* (2002, 2005a)]. In general, if λ is the local inter-ridge distance of the corresponding minutia, then the length of a spur is not more than $3\lambda/2$.

A small region of a gray-scale fingerprint image with 500 dpi resolution that gives rise to a spur is shown in Fig. 3.2(a), and the corresponding skeletonized version in Fig. 3.2(b). It is evident from Fig. 3.2(b) that the spur length is 6 pixels, which is about $\lambda/2$ for the associated block of region. The spur has given rise to two false minutiae: one being a false bifurcation minutia on the ridge from where it has originated, and the other being a false termination minutia where it ends. The two false minutiae, arising out of the spur, are shown as black pixels in Fig. 3.2(b).

Another small region of a gray-scale image topography that contains a bridge and the corresponding skeletonized image are shown in Figs. 3.2(c) and 3.2(d). The latter is basically a ternary image, where, the darker lines are ridges and the fainter ones are valleys against a white background. The use of ternary image (ridge, valley, and background) can be found in many existing techniques on fingerprint matching [Bhowmick *et al.* (2002), Haralick (1983), Hung (1993)]. A bridge is nearly orthogonal to the pair of ridges it is connected to, and its length is, in general, not more than $3\lambda/2$ [Bhowmick *et al.* (2005a), Farina *et al.* (1999)]. Moreover, a bridge gives rise to a valley break [Bhowmick *et al.* (2005a), Hung (1993)], thereby creating two valley terminations on its two sides. The two valley terminations on two sides of the bridge (of ridge skeleton) are shown as black pixels in Fig. 3.2(d). The bridge itself gives rise to two false minutiae on the pair of ridges it is connected to. Similarly, short ridges, loops, and ridge breaks are also very common image impurities, giving rise to false minutiae, whose characteristic nature and invalidation techniques can be found in the literature [Bhowmick *et al.* (2005a), Farina *et al.* (1999), Hung (1993)].

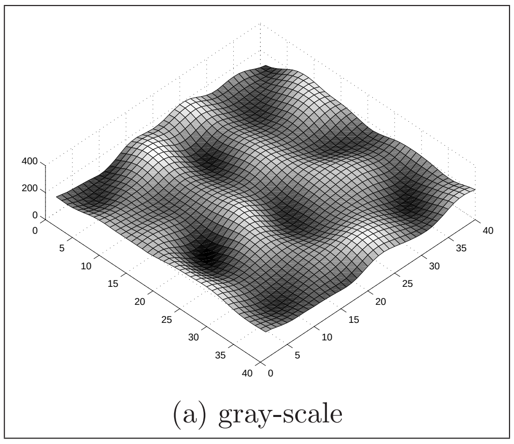
During preprocessing, apart from spurs, bridges, loops, etc., several spurious and misleading lines appear in the thinned image because of the noise present in the original gray-scale image. These lines are mere aberrations that often give rise to poor or not-so-obvious minutiae, thereby delaying the process of minutiae matching, or reporting a poor fingerprint match. Spurs, bridges, loops, etc. are easily detectable in a less noisy region. In a substantially large noisy part of an image, several crisscrosses may arise that are not always detectable as spurs, bridges, or loops. A small region from such a noise-affected area is shown in Fig. 3.3. There may also exist some minutiae in a noise-free region (apparently, by the naked eye) that are feebly recognizable in the gray-scale image because of erratic gray-value pattern in that locality. As a result, an ambiguity may arise regarding the inclusion or exclusion of a minutia depending on its visual clarity in the original gray-scale image.

The post-processing job on authentication of the detected set of minutiae is also performed in the gray-scale domain under supervised learning. The minutiae-based matching procedure reported by Prabhakar *et al.* (2003) uses feed-forward of original fingerprint image to the feature (minutiae) verification stage. The verification stage re-examines the gray-scale profile in the neighborhood of a minutia using Learning Vector Quantization. An overall quality score is assigned to an image, but not to any individual minutia. On the contrary, our method assigns scores to all valid minutiae, which can be used to evaluate the overall quality of the concerned image by an efficient technique. Further, it does not need any training. An important assertion by Prabhakar *et al.* (2003) that reinforces our findings is that, they have mentioned about the possibility of improving the accuracy of the fingerprint verification system by some modified matching algorithm that can take the confidence value (referred to as *score* in our work) of the individual minutiae into account. Another technique proposed by Jiang and Ser (2002) improves the fingerprint templates by merging and averaging minutiae of multiple fingerprints. The weighted averaging scheme enables the template to change gradually with time according to the change of skin and imaging conditions. This reduces the storage and computation requirements by its inherent recursive nature. It can be said that our work just echoes the same idea when only one fingerprint image is available for an individual being. In fact, the method of assigning scores to the minutiae will be of greater significance when we have a series of fingerprint images for the same being over a prolonged period of time.

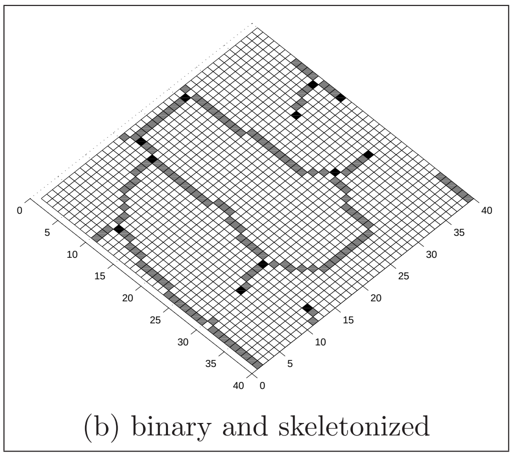
In order to circumvent the aforesaid uncertainty, we propose a methodology of assigning a score value to each minutia, after elimination of spurs, bridges, loops, etc. Each minutia is assigned an integer score in the scale $[1, 100]$ depending on its topographical characteristics in the skeletonized ternary image (ridge, valley, and background), which in turn, are derived from its visual prominence in the original gray-scale image. It may be noted that the proposed score-based technique can be used to expedite both the fingerprint identification problem (1-to-N) and the verification problem (1-to-1).

3.3 Minutiae Scores in Fingerprint Matching

Let P be the *pattern set* of minutiae that has to be checked for a match with some subset of the *database set* or the *background set*, namely Q, the latter being stored in



(a) gray-scale



(b) binary and skeletonized

Figure 3.3: Gray-scale topography of a noisy zone and its corresponding skeletonized binary structure.

the fingerprint database. The existing matching schemes do not discriminate among the minutiae apropos their quality either in the database set or in the pattern set. In these schemes, a match is reported if the coordinates, the types and angles of minutiae of pattern set P are found to be agreeing with those of database set Q under certain transformations like translation, rotation, or scaling [Hrechak and McHugh (1990), Jain *et al.* (1997), Kovács-Vajna (2000), Maio and Maltoni (1997), Pernus *et al.* (1980)]. The authenticity of the minutiae, in general, is not taken into consideration.

In order to consider the relative quality of a minutia in a fingerprint image as a practical matching criterion, we define a minutia p as a 5-tuple, namely $p = \langle x, y, t, \theta, s \rangle$, where, $\langle x, y \rangle$ = coordinates of p, θ = angle made by the tangent to the corresponding ridge at the point p, and s = an integer score associated with the minutia p.

The angle θ corresponding to a bifurcation minutia and a termination minutia are shown in Fig. 3.4(a) and Fig. 3.4(b) respectively. For each valid minutia p, the corresponding value of the associated ridge direction θ, can be estimated by the conventional linear regression technique.

The score values are normalized within a scale of 1 to 100, where, a minutia with score nearing 100 is of the highest significance compared to any other minutia with a lower score value. In other words, if a minutia p_1 has a score s_1, and another minutia p_2 has a score s_2, and if $s_1 < s_2$, then p_1 is a less dependable minutia than p_2. Fig. 3.5(a) exhibits a small region of a ternary skeletonized image (ridge, valley, and background) which is free

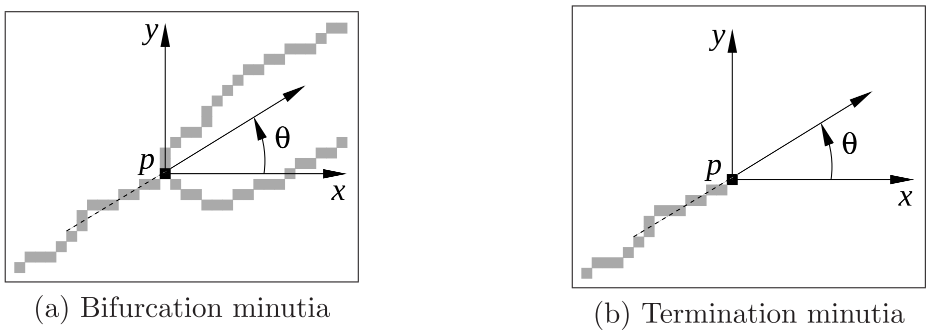


Figure 3.4: Ridge direction θ for a minutia p.

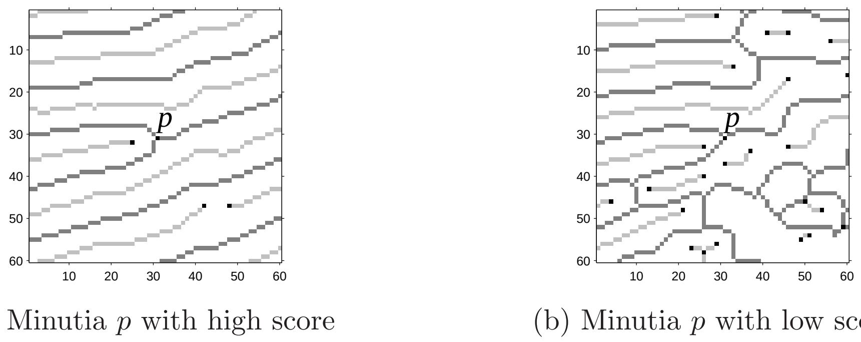


Figure 3.5: Typical ridge and valley skeletons in the local neighborhood of a strong minutia (with high score) and a weak minutia (with low score).

of noise, and Fig. 3.5(b) shows a similar region affected by noise. It is quite evident from this figure that the minutiae p in Fig. 3.5(a) will have a fairly high score, whereas, the minutiae p in Fig. 3.5(b) will have a poor score value. The ridgelines as well as the valley lines in the neighborhood of p in Fig. 3.5(b) have erratic and irregular flow patterns that indicate the presence of noise in this region. On the contrary, the ridge and valley lines in Fig. 3.5(a) show a smooth flow pattern that speaks of the tidiness of the region. In Figs. 3.5(a) and 3.5(b), the darker (fainter) lines represent the ridges (valleys). Both the ridge minutiae and valley minutiae in these figures are highlighted by black pixels.

While applying a fingerprint matching procedure based on minutiae, the scores of minutiae of P and those of Q can be used to predict how good or bad the match is. Let Q' be a subset of Q, and P' be a subset of $P^{R,T,S}$, where, $P^{R,T,S}$ has been obtained from P after suitable transformations of rotation (R), translation (T), and scaling (S), such that Q' and P' form the best possible matching pair of subsets. Let, $|P| = m$, $|Q| = n$, and $|P'| = |Q'| = \widehat{n}$. If a minutia (x_i, y_i) with score s_i in set P' is a potential match with a minutia (x_j, y_j) with score s_j in set Q', then the difference between s_i and s_j ($|s_i - s_j|$) indicates the quality of matching of (x_i, y_i) and (x_j, y_j). For a matching between P and Q with $\widehat{n}$ minutiae, we define the *Matching Index* by

$$MI = \frac{1}{n_{\max}} \sum_{i=1}^{\widehat{n}} \{\frac{1}{2}(s_i + s_j) - \omega|s_i - s_j|\}, \tag{3.1}$$

where, $n_{\max} = \max(m, n)$, and $0 \leq \omega \leq \frac{101}{198}$.

The rationale of Eqn. 3.1 is as follows. Since $1 \leq s_i, s_j \leq 100$ for $1 \leq i \leq \widehat{n}$, the maximum value of $(s_i + s_j)$ is 200, whereas, the minimum value is 2. Thus, for a match between two best possible minutiae, each with score 100, a value of 100 is contributed to the matching index. Similarly, a value of 1 is contributed due to a match between two worst possible minutiae, each with score 1. The parameter ω represents the weight attached to the difference of scores of two matching minutiae in their way of participating in the estimate of MI. It can be shown that the value of ω should lie between 0 and $\frac{101}{198}$ so that $\{\frac{1}{2}(s_i + s_j) - \omega|s_i - s_j|\}$ is never negative. The case of worst matching (hence a minimum contribution to MI in Eqn. 3.1) between two corresponding minutiae arises when one between s_i and s_j is 100 and the other is 1; that is, s_i and s_j differ by maximum extent. Therefore, in order to reduce the contribution to MI for the worst case to as low as zero (the lowest possible in our procedure), we should have (as per Eqn. 3.1) $\frac{1}{2}(100 + 1) - \omega|100 - 1| = 0$, or, $\omega = \frac{101}{198}$. A high value of ω nearing $\frac{101}{198}$ signifies that not only the average score of the matching minutiae is considered in the estimate of MI, but also their difference in scores is taken into consideration with a high weight. For instance, for $\omega = \frac{1}{2}$, if $s_i > s_j$, then the contribution to MI by the corresponding match is $\frac{1}{2}(s_i + s_j) - \frac{1}{2}(s_i - s_j) = s_j = \min(s_i, s_j)$. Similarly, for $\omega = \frac{1}{2}$, if $s_j > s_i$, then the contribution to MI by the corresponding match is $\frac{1}{2}(s_i + s_j) - \frac{1}{2}(s_j - s_i) = s_i = \min(s_i, s_j)$. Thus, for $\omega = \frac{1}{2}$, if one of s_i and s_j is very high, and the other one very low, the contribution to MI by the matching pair (x_i, y_i) and (x_j, y_j) is as poor as the score of the minutia of worse quality. However, as ω approaches 0, the contribution to MI by the matching pair (x_i, y_i) and (x_j, y_j) approaches the average score of this pair of minutiae.

Furthermore, from Eqn. 3.1, it is also evident that, if $\widehat{n}$ is quite small compared to $n_{\max}$, then MI will be also quite low even though the scores of all matching minutiae may be very high. Similarly, MI will also be quite low if scores of each pair of matching minutiae vary widely instead of $\widehat{n}$ being close to $n_{\max}$. MI will be high only if $\widehat{n}$ is close to $n_{\max}$ and all matching minutiae are of good quality. The ideal case for $MI = 100$ occurs only when $m = n = \widehat{n}$ and $s_i = s_j = 100$, for $i = 1, 2, \ldots, \widehat{n}$. A score-based generic structure of an AFIS is shown in Fig. 3.6.

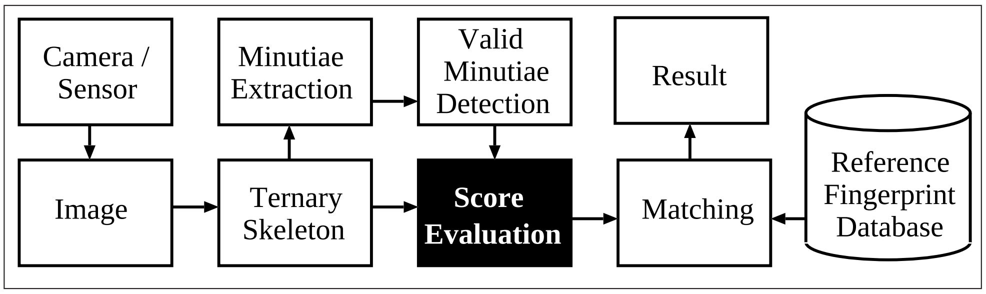


Figure 3.6: Generic structure of an AFIS considering minutiae scores.

3.4 Evaluation of Score

The score s of a minutia p is estimated based on the following properties:

(i) pattern of ridge flow in and around p;
(ii) pattern of valley flow in and around p;
(iii) noise level in the locality of p.

If the ridge and valley lines in the local neighborhood of p have a smooth nature of flow, then the corresponding minutia p will have a genuine contribution in the fingerprint matching. On the contrary, if the ridge and valley lines in some region have an erratic or uneven nature of flow, then a minutia p' in that region should not predominate in the matching procedure. The former minutia (p), being located in a tidy region, lends more confidence in the matching procedure than the latter (p'), which is located in a noisy region.

For a minutia $p(x, y)$, the score is given by the equation

$$s = \lfloor s_\text{r} \rfloor + \lfloor s_\text{v} \rfloor + \lfloor s_\text{n} \rfloor, \tag{3.2}$$

where, s_r, s_v, and s_n are the score components due to ridge flow, valley flow, and noise level respectively in the local neighborhood of p. The components s_r and s_v denote measures of (ideality of) ridge and valley flows respectively, which are evaluated based on some distances estimated in the local ridge and valley topography around the minutia p. To take into account the noise of the region in and around p, the component s_n is estimated in a local window centered at p. Noise imparts a negative effect on the score.

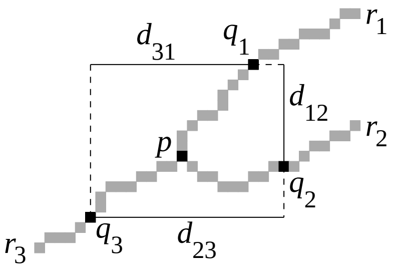


Figure 3.7: Ridges r_1, r_2, and r_3 incident at the bifurcation minutia p.

3.4.1 Score of a Bifurcation Minutia

Let λ be the average inter-ridge distance of a fingerprint image. First, we find the three neighbor pixels p_1, p_2, p_3 of a bifurcation minutia p, considering 8-neighborhood connectivity. p_1, p_2, p_3 are the three starting pixels of the ridges r_1, r_2, r_3 respectively, incident at p. We explore a walk along each of r_1, r_2, r_3 starting from p_1, p_2, p_3 respectively, each walk being of length λ. Let these walks be named as w_1, w_2, and w_3 respectively. If during some walk $w_i, 1 \leq i \leq 3$, any bifurcation or termination minutia is encountered, the walk is halted. Let, $l_i, 1 \leq i \leq 3$, denote the length of the walk w_i. Let, $l_{\min}$ be the minimum of $l_i, 1 \leq i \leq 3$, and μ be the number of walks whose lengths are less than λ. If p is a minutia of good quality, then each l_i should be at least $\lambda/2$, and at least two of them should be λ. So, if $l_{\min} < \lambda/2$ or, $\mu \geq 2$, we assign 0 to score s and return from this point. Otherwise, if $l_{\min} < \lambda$, then we walk for a length $l_{\min}$ along each of the three ridges r_1, r_2, r_3 starting from p_1, p_2, p_3 respectively, so that after the (re-)walks, each of the points q_1, q_2, q_3, reached on the three ridges r_1, r_2, r_3 respectively, is at equal distance from p (Fig. 3.7).

We need to identify the ridgeline that bifurcates at p. In Fig. 3.7, the three ridges are shown as r_1, r_2, and r, where, w.l.o.g., $r(= r_3)$ has been depicted as the pre-bifurcated ridge, and r_1 and r_2 are its two bifurcations at p. To identify the pre-bifurcated ridge, we define $d_{\min} = \min(d_{12}, d_{23}, d_{31})$, where, d_{ij} denotes the (isothetic) distance between $q_i(x_i, y_i)$ and $q_j(x_j, y_j)$ given by $d_\top(q_i, q_j) = \max\{|x_i - x_j|, |y_i - y_j|\}$, $1 \leq i, j \leq 3, i \neq j$. If q_1 and q_2 are on the two bifurcated ridges r_1 and r_2, then $d_{12} < d_{23}$ and $d_{12} < d_{31}$. However, this condition may fail if p is a poor minutia candidate, viz., when the ridges incident at p are of uneven nature, and it is difficult to ascertain the pre-bifurcated ridge among r_1, r_2, r_3. Hence, if $d_{\min} > 3l_{\min}/2$, then we assign 0 to score s, and return.

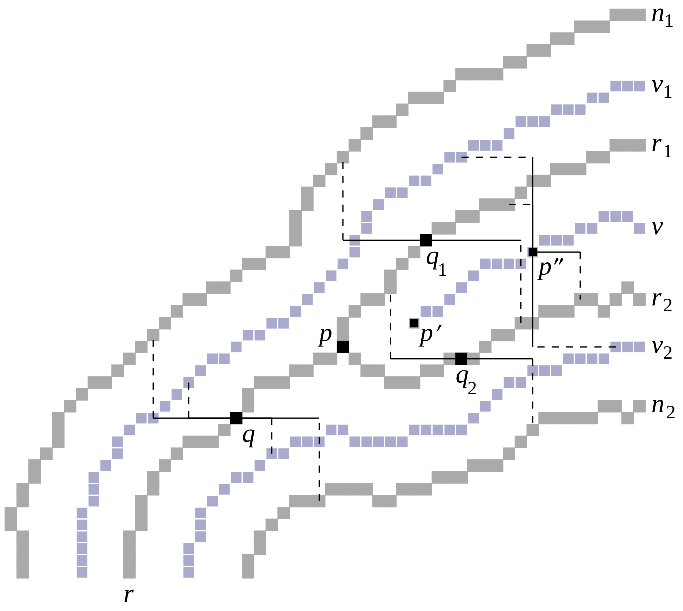


Figure 3.8: Ridge and valley characteristics in the local neighborhood of a bifurcation minutia p (see text for explanation).

In order to compute the score s_{r} for a bifurcation minutia p, we define the following distances (Fig. 3.8):

d_{qn_1} = distance from q to neighbor ridge n_1;
d_{qn_2} = distance from q to neighbor ridge n_2;
$d_{q_1n_1}$= distance from q_1 to neighbor ridge n_1;
$d_{q_2n_2}$= distance from q_2 to neighbor ridge n_2;
$d_{q_1r_2}$= distance from q_1 to bifurcated ridge r_2;
$d_{q_2r_1}$= distance from q_2 to bifurcated ridge r_1.

For a strong minutia, the above distances should be close to λ. So, s_{r} is assigned to p depending on the closeness of $d_{\mathrm{BM},ri} \in \{d_{qn_1}, d_{qn_2}, d_{q_1n_1}, d_{q_2n_2}, d_{q_1r_2}, d_{q_2r_1}\}$ w.r.t. λ. Thus, for a bifurcation minutia p, the score w.r.t. the ridge characteristics can be chosen as

$$s_{\mathrm{r}} = \alpha_{ri} \sum_{d_{\mathrm{BM},ri}} \frac{1}{\lambda}(\lambda - |\lambda - d_{\mathrm{BM},ri}|), \tag{3.3}$$

where, α_{ri} is the *ridge score multiplier for bifurcation minutiae.*

Similarly, the score s_{v} for the bifurcation minutia p is based on the following set of distances:

d_{qv_1} = distance from q to neighbor valley v_1;
d_{qv_2} = distance from q to neighbor valley v_2;
$d_{pp'}$ = distance from p to valley termination minutia p', if any, lying near p in between r_1 and r_2;
$d_{p''r_1}$ = distance from p'' to bifurcated ridge r_1;
$d_{p''r_2}$ = distance from p'' to bifurcated ridge r_2;
$d_{p''v_1}$ = distance from p'' to neighbor valley v_1;
$d_{p''v_2}$ = distance from p'' to neighbor valley v_2;

where, p'' is the point along the valley v at a distance λ from p', or, a bifurcation or termination of v appearing within the target walk-length of λ. The distances $d_{pp'}$, $d_{p''r_1}$, $d_{p''r_2}$, $d_{p''v_1}$, and $d_{p''v_2}$ exist only if p' exist near p in between r_1 and r_2. We consider a valley termination p' to be valid corresponding to a ridge bifurcation p if the isothetic distance between p and p' does not exceed $3\lambda/2$ and p' lies between the bifurcated ridgelines, r_1 and r_2, as shown in Fig. 3.8.

While the parameter $\{d_{\mathrm{BM},ri}\}$ represents some kind of inter-ridge distance, we define other distance measures with a subtle difference. Distances in the set $\{d^1_{\mathrm{BM},va}\} = \{d_{p''v_1}, d_{p''v_2}\}$ are inter-valley distances, which should be ideally close to λ. The other set $\{d^2_{\mathrm{BM},va}\} = \{d_{qv_1}, d_{qv_2}, d_{pp'}, d_{p''r_1}, d_{p''r_2}\}$ contains distances from a ridge point to a valley line, or from a valley point to a ridgeline, and therefore, allowed for a flexibility in their contribution to s_v. Hence, distances in $\{d^1_{\mathrm{BM},va}\}$ participate in a way similar to distances in $\{d_{\mathrm{BM},ri}\}$ in the process of estimating s_v. Their contribution to score may be chosen as

$$s^1_\mathrm{v} = \alpha_{va} \sum_{d^1_{\mathrm{BM},va}} \frac{1}{\lambda}(\lambda - |\lambda - d^1_{\mathrm{BM},va}|); \tag{3.4}$$

and that due to $\{d^2_{\mathrm{BM},va}\}$ is

$$s^2_\mathrm{v} = \sum_{d^2_{\mathrm{BM},va}} s_{d^2_{\mathrm{BM},va}}, \tag{3.5}$$

where, $s_{d^2_{\mathrm{BM},va}}$ is chosen as

$$s_{d^2_{\mathrm{BM},va}} = \begin{cases} \alpha_{va}.1 & \text{if } \lambda/4 \le d^2_{\mathrm{BM},va} \le 3\lambda/4 \\ \alpha_{va}\,\frac{1}{\lambda}(d^2_{\mathrm{BM},va} - \lambda/4) & \text{if } d^2_{\mathrm{BM},va} < \lambda/4 \\ \alpha_{va}\,\frac{1}{\lambda}(3\lambda/4 - d^2_{\mathrm{BM},va}) & \text{if } d^2_{\mathrm{BM},va} > 3\lambda/4, \end{cases} \tag{3.6}$$

and α_{va} is the *valley score multiplier for a bifurcation minutia.*

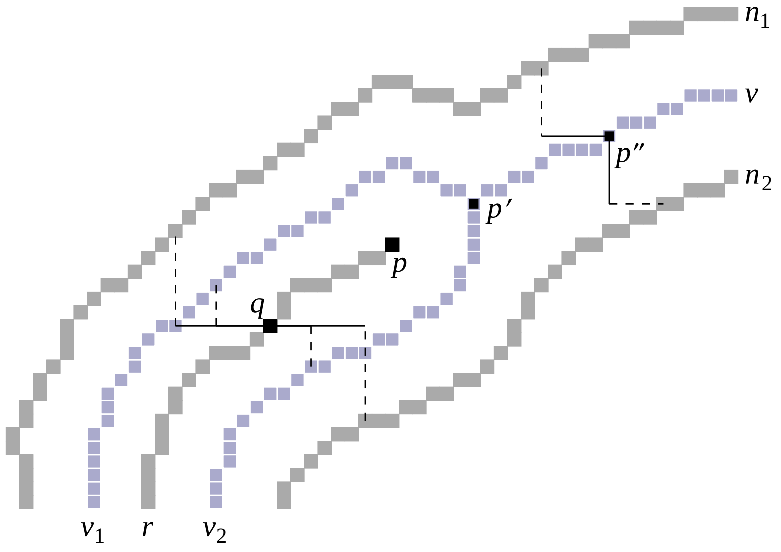


Figure 3.9: Ridge and valley characteristics in the local neighborhood of a termination minutia p.

A brief reasoning for the development of Eqn. 3.6 is as follows. Since $d^2_{\mathrm{BM},va}$ is the distance from a ridge point to a neighboring valley line, or from a valley point to a neighboring ridgeline, it should be ideally equal to $\lambda/2$. But by allowing a tolerance of $\pm\lambda/4$, a contribution of α_{va} to the score is made if $d^2_{\mathrm{BM},va}$ lies within $\lambda/4$ and $3\lambda/4$. On the other hand, the contribution to score is made negative if $d^2_{\mathrm{BM},va}$ is not within the desired bounds. Furthermore, to incorporate the deviation of $d^2_{\mathrm{BM},va}$ from the desired value, the contribution by $d^2_{\mathrm{BM},va}$ is made increasingly negative as it goes farther and farther from $\lambda/4$ or $3\lambda/4$.

3.4.2 Score of a Termination Minutia

Let p be a termination minutia and N be the adjacent ridge pixel of p, considering 8-neighborhood. Since p is a termination minutia, there will be only one ridgeline, say r, incident at p (Fig. 3.9). We walk along r starting from N, for a length λ, and designate the walk as w. Let l denote the length of the walk. Since a skeletonized fingerprint image should be devoid of spurs and bridges, l should always be equal to λ.

Let q be the point on the ridge r reached after the walk w. For estimation of the score s_{r} for the termination minutia with respect to ridgelines in the region containing p, we define the set $\{d_{\mathrm{TM},ri}\}$ of following distances:

d_{qn_1}= distance from q to neighbor ridge n_1;
d_{qn_2}= distance from q to neighbor ridge n_2.

For p to be a termination minutia of good quality, the above distances should be close to λ. These distances are basically inter-ridge distances similar to $\{d_{\text{BM},ri}\}$ in case of bifurcation minutiae. Hence, the score s_r is assigned to p based on the following equation that resembles with Eqn. 3.3 in form.

$$s_\text{r} = \beta_{ri} \sum_{d_{\text{TM},ri}} \frac{1}{\lambda}(\lambda - |\lambda - d_{\text{TM},ri}|); \tag{3.7}$$

where, β_{ri} is the *ridge score multiplier for termination minutiae.*

Similarly, the score s_v for the termination minutia p is based on the set $\{d_{\text{TM},va}\}$ of following distances:

d_{qv_1} = distance from q to neighbor valley v_1;
d_{qv_2} = distance from q to neighbor valley v_2;
$d_{pp'}$ = distance from p to valley termination minutia p', if any, lying near p in between n_1 and n_2;
$d_{p''n_1}$= distance from p'' to neighbor ridge n_1;
$d_{p''n_2}$= distance from p'' to neighbor ridge n_2;

where, p'' is the point along the valley v at a distance λ from p', or, a bifurcation or termination of v appearing within the target walk-length of λ. The distances $d_{pp'}$, $d_{p''n_1}$, and $d_{p''n_2}$ exist only if p' is found near p in between n_1 and n_2. The criterion for such a valley minutia p' is similar to that discussed in Sec. 3.4.1.

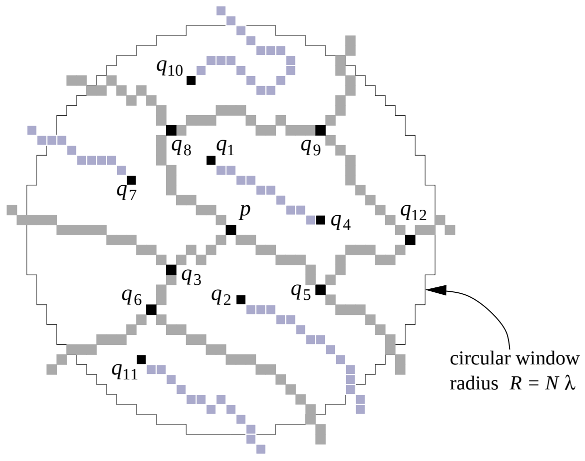
The above set of distances are measured either from a ridge point to a valley line or from a valley point to a ridgeline. Hence, their contribution to score s_v is given by

$$s_\text{v} = \sum_{d_{\text{TM},va}} s_{d_{\text{TM},va}}, \tag{3.8}$$

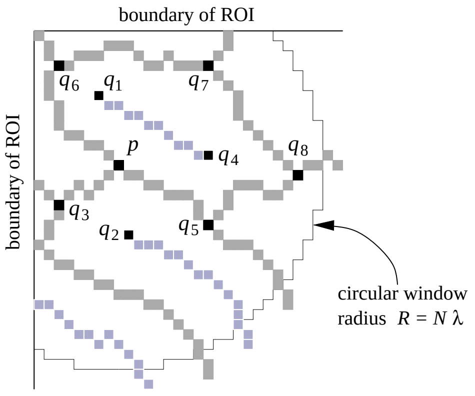
where, $s_{d_{\text{TM},va}}$ is chosen as

$$s_{d_{\text{TM},va}} = \begin{cases} \beta_{va}.1 & \text{if} \quad \lambda/4 \le d_{\text{TM},va} \le 3\lambda/4 \\ \beta_{va}\,\frac{1}{\lambda}(d_{\text{TM},va} - \lambda/4) & \text{if} \quad d_{\text{TM},va} < \lambda/4 \\ \beta_{va}\,\frac{1}{\lambda}(3\lambda/4 - d_{\text{TM},va}) & \text{if} \quad d_{\text{TM},va} > 3\lambda/4, \end{cases} \tag{3.9}$$

and β_{va} is the *valley score multiplier for a termination minutia*, the rationale of Eqn. 3.9 being same as that of Eqn. 3.6.



(a) The window W lies entirely within the region of interest.



(b) The window W lies partially within the region of interest.

Figure 3.10: Example showing contributing points ($\{q_1, q_2, \ldots, q_{12}\}$ in case (a) and $\{q_1, q_2, \ldots, q_8\}$ in case (b)) in a circular window W centered around the minutia p.

3.4.3 Estimation of Noise

Let p be a bifurcation or termination minutia having a positive score after the evaluation of s_r and s_v. If p does not have a positive score, we need not evaluate s_n, since s_n will contribute a negative score to p; finally we will consider only the set of minutiae with positive scores. Consider a circular window W of radius $R = N\lambda$ centered at $p(x, y)$, as shown in Fig. 3.10(a) and Fig. 3.10(b). In Fig. 3.10(a), W lies entirely within the region of interest (ROI), whereas, in Fig. 3.10(b), W has a partial overlap with the ROI of the corresponding image. Let W' be the region of overlap between W and ROI of the image. Let $\{q_i : q_i \text{ lies within } W'; i = 1, 2, \ldots, \eta\}$ be the set of points, with each point q_i satisfying any one of the following 3 properties (Figs. 3.10(a) and 3.10(b)):

(i) q_i is a ridge minutia with $s_\text{r} + s_\text{v} \leq 0$;
(ii) q_i is a non-minutia ridge point having three or more ridges incident upon it;
(iii) q_i is either a valley bifurcation or a valley termination minutia.

The above definition enables us to use $|\{q_i\}| = \eta$ as a measure of noise level in the window W centered around p. We define another parameter ν, called the *noise factor*, which is used to find the *noise threshold*, τ_{noise}, given in the equation below, that will indicate whether or not the window W associated with a minutia p is noisy.

$$\tau_{noise} = \nu \frac{A}{\lambda^2} \tag{3.10}$$

where, A is the net area of overlap between W and the region of interest (ROI) of the image. In Fig. 3.10(a), since W is entirely within the ROI, $A = \pi R^2$, whereas, in Fig. 3.10(b), since W lies partially within the ROI, A is less than πR^2. Contributing points will be more in number as the overlap between W and ROI is high and will be less in number when W is centered at a minutia p located near the border of ROI. Hence, the factor $\frac{A}{\pi R^2}$ is directly related to the number of contributors within the circular window W. Also, since $R = N\lambda$, the area of window W is directly proportional to N^2 for the concerned image. Hence, on the assumption that the number of contributors within W varies directly with the area of W, Eqn. 3.10 is derived as follows.

$$\tau_{noise} = [constant] \times N^2 \frac{A}{\pi R^2} = [constant] \times \frac{A}{\pi \lambda^2} = \nu \frac{A}{\lambda^2}.$$

If η is higher than τ_{noise} in W corresponding to $p(x, y)$, the noise level in W is considered high enough and each point $q_i(x_i, y_i)$, $i = 1, 2, \ldots, \eta$, is accounted one by one for their individual contribution to the noise-induced (negative) score s_n of p. Thus, Eqn. 3.11 can be used to find s_n^i attributed by each q_i, and Eqn. 3.12 sums up the individual scores to compute the total score component due to noise.

$$s_\text{n}^i = \gamma(R - d_\top(p, q_i)), \tag{3.11}$$

$$s_\text{n} = \begin{cases} 0 & \text{if} \quad \eta \leq \tau_{noise}, \\ \sum_{i=1}^{\eta} s_\text{n}^i & \text{if} \quad \eta > \tau_{noise}, \end{cases} \tag{3.12}$$

where, γ is the *noise score multiplier*, and $d_\top(p, q_i) = \max\{|x - x_i|, |y - y_i|\}$.

3.4.4 Normalization of Score

In order to bound the minutiae scores in the range of $[1, 100]$, the value of α_{ri} $(= \alpha_{va})$ has been chosen as $\frac{100}{13}$. The reason is as follows. For evaluating the score of a bifurcation minutia, we need to compute 6 distances in the set $\{d_{\text{BM},ri}\}$, measured w.r.t. different ridgelines, and 7 distances in the set $\{d_{\text{BM},va}\}$, measured w.r.t. different valley lines. In each of the three Eqns. 3.3, 3.4, and 3.6, value of each of the 13 score elements before getting multiplied by α_{ri} or α_{va} is at most unity. Since there are 13 such score elements, 6 from $\{d_{\text{BM},ri}\}$ and 7 from $\{d_{\text{BM},va}\}$, maximum score obtainable by a bifurcation minutia, before being multiplied by the *ridge score multiplier* and *valley score multiplier*, is 13. Assigning $\alpha_{ri} = \alpha_{va} = \frac{100}{13}$ fixes the total maximum score to 100. Another suggestive choice can be $\alpha_{ri} = \frac{50}{6}$ and $\alpha_{va} = \frac{50}{7}$, giving equal weight to the total ridge score component and total valley score component. Our choice for $\alpha_{ri} = \alpha_{va} = \frac{100}{13}$ comes from the principle of assigning equal weight to all 13 distance measures instead of assigning equal weight to the two different score components.

Similarly, for finding the score of a termination minutia with respect to the ridge and valley skeletons, we measure 7 distances in total, 2 distances in the set $\{d_{\mathrm{TM},ri}\}$ and 5 in $\{d_{\mathrm{TM},va}\}$, respectively. In choosing the values for β_{ri} and β_{va}, we have adopted to equal weightage for all these 7 distances, rather than equal weightage to total ridge score component and total valley score component. Therefore, in our experiments, we have taken $\beta_{ri} = \beta_{va} = \frac{100}{7}$.

3.5 Fingerprint Matching

Given an existing database representation of a fingerprint image and a similar input representation extracted from a pattern image, the matching stage in a fingerprint verification system determines the similarity of the two fingerprint representations and decides whether they are from the same finger. The most elegant and efficient representation of fingerprints, which is also adopted in the conventional automatic fingerprint identification systems (AFIS), is based on a common hypothesis called minutiae owing to their unique ability to capture the invariant and discriminatory information present in a fingerprint image. In this work, we have considered two most prominent kinds of minutiae, namely, ridge termination and ridge bifurcation, which are used by the Federal Bureau of Investigation [Wegstein (1982)] and adopted in many AFIS. There exist several techniques [Bishnu *et al.* (2002), Jiang *et al.* (1999), Maio and Maltoni (1997)] to extract the minutiae from a gray-scale fingerprint image.

Being a non-ideal mapping of some part of a three-dimensional finger to a two-dimensional plane, the acquired fingerprint image, and its corresponding representation there of, inevitably suffers from unpredictable complications, some of which are:

(i) occurrence of spurious minutiae and absence of genuine minutiae during minutiae detection phase;
(ii) global translation, rotation, and scaling of the minutiae pattern due to unknown alignment of the finger during image acquisition;
(iii) local non-linear deformations due to uneven pressure at the fingertip or/and digiti-

zation error in the acquisition mechanism;

(iv) reduction in actual region of interest due to partial overlap between the two impressions;

(v) cut marks and skin imperfections developed in the time span elapsed between acquisitions of the two impressions.

Considering all these factors, a necessary foundation for achieving a good matching performance is to construct a reliable, realistic, and robust model of fingerprint matching that takes into account all sort of deviations and anomalies, which are very likely to occur between two sets of minutiae extracted from different impressions of the same finger. A number of approaches have been proposed, and most of them are by minutiae matching based on some variety of point pattern matching [Jain *et al.* (1997)], and by structural matching [Hrechak and McHugh (1990), Wahab *et al.* (1998)]. However, even these methods fail to perform well [Jain *et al.* (2001)], because they did not make use of the rich information content present in a fingerprint pattern. Presently, most fingerprint matching algorithms follow a dual strategy that combines the minutiae matching with some method that captures the ridge structure properties in order to improve the overall matching performance [Bhowmick and Bhattacharya (2004a), Jain *et al.* (2001), Jiang and Yau (2000), Luo *et al.* (2000)].

3.6 Score-based Fingerprint Matching

We have developed a new fingerprint matching technique [Bhowmick and Bhattacharya (2004a)] that exploits both the local topological structures of a valid minutia and the global geometric structure of the minutiae set as a whole. Both the local and global structures have been used adaptively in our algorithm, making the matching procedure more meaningful, efficient, and robust. Though our method also follows a dual strategy by combining both the local and global perspectives, it is notably different from the existing methods [Jain *et al.* (2001), Jiang and Yau (2000), Luo *et al.* (2000)] as explained below.

We describe each minutia point p_i, detected from a fingerprint image, by a 6-element feature vector given by

$$p_i = \langle x_i, y_i, t_i, \phi_i, \lambda_i, s_i \rangle, \tag{3.13}$$

where, x_i, y_i are the rectangular coordinates of p_i w.r.t. image frame; t_i is the type of minutia (ridge termination or bifurcation); $\phi_i \in [0, 360)$ is the local ridge direction at p_i, measured in the counterclockwise direction w.r.t. +ve x-axis; λ_i is the local inter-ridge distance at p_i; s_i is the score associated with p_i as mentioned in Sec. 3.4.

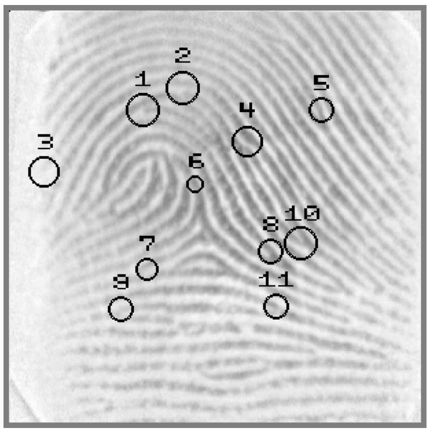


Figure 3.11: Detected minutiae with impressions of scores in a gray-scale fingerprint image. The score of a minutia is proportional to the radius of the circle shown around it.

Our consideration of local structural characteristics of a minutia, therefore, differs from the existing ones [Jain *et al.* (2001), Jiang and Yau (2000), Luo *et al.* (2000)]. In the procedure by by Jain *et al.* (2001), the gray-scale variance within a tessellated cell quantifies the underlying ridge structure and is used as a feature; Luo *et al.* (2000) considers only the associated ridge on which the minutia lies; the inter-minutia distance and ridge count are considered by Jiang and Yau (2000) as local structural measure to facilitate the preliminary matching. On the contrary, in our work, for all detected minutiae, we have considered their scores, which embrace all the relevant and useful topological properties and discard the detrimental ones, thereby enriching the minutiae-based matching by final evaluation of a matching score. An example of detected set of minutiae with positive scores is shown in Fig. 3.11, where, the minutiae with positive scores are encompassed by circles, the radius of the circle centered at a minutia being proportional to its score, and label 'i' of a minutia corresponds to the leaf node 'pi' in $\mathcal{K}_k$ in Fig. 3.12.

Let there be N images in the database, and let $\mathcal{F}_k$, $1 \leq k \leq N$, represent the kth database image having n_k minutiae described by the feature set $\mathcal{S}_k = \{p_i\}_{i=1}^{n_k}$. In the preprocessing (off-line) phase, a primary data structure $\mathcal{T}_k$ is defined over each $\mathcal{S}_k$. $\mathcal{T}_k$ is an AVL tree that permits 1-dimensional range searching [Berg *et al.* (2000)] in fastest possible time. The Euclidean distance $d(p_i, p_j)$ between two distinct minutiae p_i and p_j is stored in a node ν_{ij} of $\mathcal{T}_k$, which has, therefore, $\binom{n_k}{2}$ nodes in total. Apart from the usual attributes present in an AVL tree, each node ν_{ij} contains two additional links pointing to the corresponding feature vectors of p_i and p_j present in $\mathcal{S}_k$.

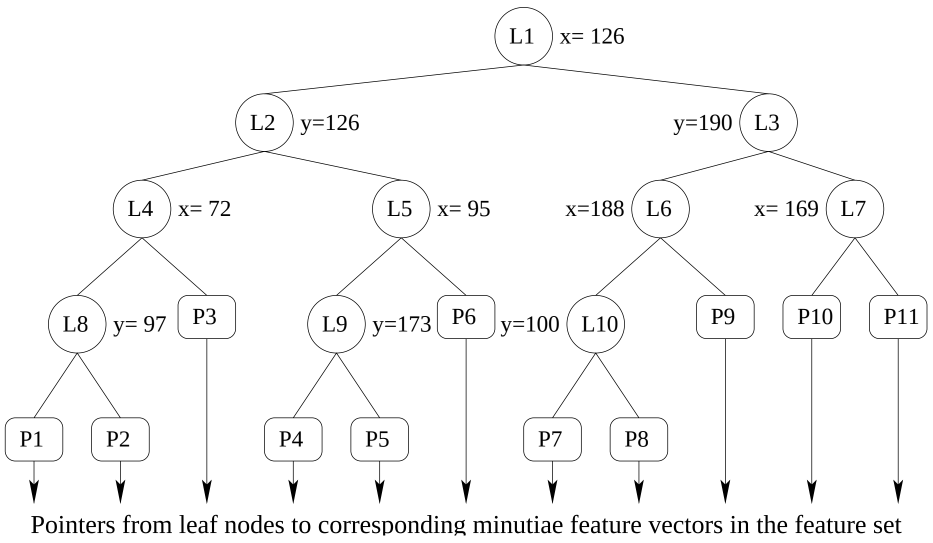


Figure 3.12: The kd-tree $\mathcal{K}_k$ storing the minutiae in the fingerprint image shown in Fig. 3.11.

When a pattern image $\mathcal{F}_l$ comes in the online stage, it is first processed to extract the minutiae feature set $\mathcal{S}_l = \{q_{i'}\}_{i'=1}^{n_l}$, where, each minutia $q_{i'}$ is represented conforming to Eqn. 3.13. The extracted set of minutiae are enumerated in $\mathcal{S}_l$ in non-increasing order of their score values, which is necessary to systematically generate all $\binom{n_l}{2}$ inter-minutia Euclidean distances in an ordered list, namely $\mathcal{L}_l = \langle d(q_{i'}, q_{j'}) : 1 \leq i' < j' \leq n_l$ & $d(q_{i'}, q_{j'})$ precedes $d(q_{i''}, q_{j''})$ only if $(s_{i'} + s_{j'}) \geq (s_{i''} + s_{j''})\rangle$. Each entry $d(q_{i'}, q_{j'})$ in $\mathcal{L}_l$ additionally has two links pointing to the corresponding minutiae $q_{i'}$ and $q_{j'}$ occurring in $\mathcal{S}_l$. The ordering of the inter-minutiae distances in $\mathcal{L}_l$ ensures the maximum likelihood of finding a matching distance(s) in $\mathcal{T}_k$ in the soonest possible time, without considering a core point. Since a core point may not be always present, or may be present in a noisy region, in a given fingerprint impression.

It is very likely that, if $\mathcal{F}_k$ and $\mathcal{F}_l$ are from the same finger, then in their overlapped region, a minutia with high score in one is present in the other, though in the later it may or may not have a high score. Hence, it will be wise to search for those minutiae which are having high scores in $\mathcal{S}_l$ into the database set $\mathcal{S}_k$. Based on this novel idea, the search space in the parameter domain is reduced.

3.6.1 Range query of a minutia in a kd-tree

A kd-tree stores a set of points in k-dimensional space, and a 2-dimensional kd-tree can be used for partial match queries in $O(\sqrt{n}+m)$ time, where, n is the number of points stored, and m is the number of (partial) matches reported in the query [Berg *et al.* (2000)]. In our work, we consider the coordinates of the minutiae for dimensional splitting while storing them in a kd-tree. For each database image $\mathcal{F}_k, 1 \leq k \leq N$, we maintain a secondary data structure $\mathcal{K}_k$, which is a 2-dimensional kd-tree containing n_k leaf nodes, where, each leaf node of $\mathcal{K}_k$ represents a distinct feature vector of $\mathcal{S}_k$. The structure of $\mathcal{K}_k$ for the image in Fig. 3.11 is shown in Fig. 3.12, where, a non-leaf node splits the preceding region by an appropriate abscissa or ordinate line whose equation is given in Fig. 3.12, and each leaf node contains coordinates of a distinct minutia and points to the corresponding feature in the list $\mathcal{S}_k$.

3.6.2 Matching algorithm based on kd-tree

If any fingerprint image has less than 6 minutiae, it is not considered as a candidate image in our matching algorithm. The major steps of our algorithm are stated below.

STEP 1. Estimate the respective mean scores $\overline{s}_k$ and $\overline{s}_l$ for $\mathcal{S}_k$ and $\mathcal{S}_l$ as follows.

$$\overline{s}_k = \sum_{i=1}^{n_k} s_i \quad \text{and} \quad \overline{s}_l = \sum_{i'=1}^{n_l} s_{i'}.$$

STEP 2. For each distance $d(q_{i'}, q_{j'})$, starting from the beginning of $\mathcal{L}_l$, till $\frac{1}{2}(s_{i'}+s_{j'}) \geq \overline{s}_l$, evaluate the average λ across the respective pair of minutiae: $\lambda_{i'j'} = \frac{1}{2}(\lambda_{i'} + \lambda_{j'})$, and the corresponding tolerance: $\varepsilon_{i'j'} = \frac{1}{2}\lambda_{i'j'}$, and search (range query) for every possible match $d(p_i, p_j)$ with $\lambda_{ij} = \frac{1}{2}(\lambda_i + \lambda_j)$, in $\mathcal{T}_k$, such that

$$\frac{d(q_{i'}, q_{j'})}{\lambda_{i'j'}} - \varepsilon_{i'j'} \leq \frac{d(p_i, p_j)}{\lambda_{ij}} \leq \frac{d(q_{i'}, q_{j'})}{\lambda_{i'j'}} + \varepsilon_{i'j'}. \tag{3.14}$$

Let $\mathcal{L}_{kij} = \langle d(p_{i_1}, p_{j_1}), d(p_{i_2}, p_{j_2}), \ldots \rangle$ be the ordered sequence of the matching distances found in $\mathcal{T}_k$, where, $(s_{i_1} + s_{j_1}) \geq (s_{i_2} + s_{j_2}) \geq \ldots$. For each such possible match $d(p_i, p_j) \in \mathcal{L}_{kij}$, starting from the beginning of $\mathcal{L}_{kij}$, if $|(\phi_i - \phi_{i'}) - (\phi_j - \phi_{j'})| \leq 22\frac{1}{2}^0$ (angular tolerance in our experiments [Bhowmick and Bhattacharya (2004a)] and also agreed to by others [Jain *et al.* (2001), Jiang and Yau (2000)]), then transform the entire pattern/query set $\mathcal{S}_l$, such that $x_{i'} = x_i$

and $y_{i'} = y_i$, i.e., the pattern vector $\overrightarrow{q_{i'}q_{j'}}$ exactly fits the matching database vector $\overrightarrow{p_ip_j}$. Eqn. 3.15 provides the necessary translation (Δx along x-axis and Δy along y-axis), rotation ($\Delta\phi$), and scaling ($\Delta\zeta$), depending on the current match.

$$\begin{bmatrix}\Delta x\\ \Delta y\\ \Delta\phi\\ \Delta\zeta\end{bmatrix}_{\left[\begin{smallmatrix}i\\j\end{smallmatrix}\right]\mapsto\left[\begin{smallmatrix}i'\\j'\end{smallmatrix}\right]} = \begin{bmatrix}\frac{1}{2}(x_i+x_j)-\frac{1}{2}(x_{i'}+x_{j'})\\ \frac{1}{2}(y_i+y_j)-\frac{1}{2}(y_{i'}+y_{j'})\\ tan^{-1}\left(\frac{y_j-y_i}{x_j-x_i}\right)-tan^{-1}\left(\frac{y_{j'}-y_{i'}}{x_{j'}-x_{i'}}\right)\\ d(p_i,p_j)/d(q_{i'},q_{j'})\end{bmatrix} \tag{3.15}$$

STEP 3. After all the points in $\mathcal{S}_l$ are given the subject transformation to obtain a new ordered feature set $\tilde{\mathcal{S}}_l = \langle \tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_{n_l} \mid \tilde{s}_1 \geq \tilde{s}_2 \geq \ldots \geq \tilde{s}_{n_l}\rangle$, for each $\tilde{q}_v = \langle \tilde{x}_v, \tilde{y}_v, \tilde{t}_v, \tilde{\phi}_v, \tilde{\lambda}_v, \tilde{s}_v\rangle \in \tilde{\mathcal{S}}_l$, a 2-dimensional range searching is applied in $\mathcal{K}_k$ with an orthogonal square query box with the diagonal dimension $\tilde{\lambda}_v$ and centered about the coordinates $(\tilde{x}_v, \tilde{y}_v)$ of $\tilde{q}_v$. We keep a counter $m(i \mapsto i', j \mapsto j')^{(k,l)}$, initialized to 0, which gets incremented by unity if a match $p_u, 1 \leq u \leq n_k$, is found.

STEP 4. For each matching pair of minutiae $(p_u \in \mathcal{S}_k, \tilde{q}_v \in \tilde{\mathcal{S}}_l)$ obtained in STEP 3, a matching score function $\xi(s_u, \tilde{s}_v)$ is defined by Eqn. 3.16, which contributes into the final overall matching score $\mu^{(k,l)}$ between the two corresponding feature sets $\mathcal{S}_k$ and $\tilde{\mathcal{S}}_l$ as shown in Eqn. 3.17.

$$\xi(s_u, \tilde{s}_v) = \frac{1}{2}(s_u + \tilde{s}_v) - \omega|s_u - \tilde{s}_v| \tag{3.16}$$

$$\mu^{(k,l)} = \frac{1}{m(i \mapsto i', j \mapsto j')^{(k,l)}} \sum_{u,v=1}^{m(i\mapsto i', j\mapsto j')^{(k,l)}} \xi(s_u, \tilde{s}_v) \tag{3.17}$$

STEP 5. Reverse the roles of $q_{i'}$ and $q_{j'}$ in STEP 2 such that $x_{i'} = x_j$ and $y_{i'} = y_j$, i.e., the query vector $\overrightarrow{q_{i'}q_{j'}}$ exactly fits the matching database vector $\overrightarrow{p_jp_i}$, and repeat the process. If the feature set matching score, thus obtained, is greater that the previous one, then update the final matching score given in Eqn. 3.17 accordingly.

STEP 6. If $\max\{m(i \mapsto i', j \mapsto j')^{(k,l)}\}$ is at least 6, and $\mu^{(k,l)}$ is at least as high as $\min(\overline{s}_k, \overline{s}_l)$, then $\mathcal{F}_l$ is reported as a matching image of $\mathcal{F}_k$ with a matching score of $\mu^{(k,l)}$.

There are several intrinsic adaptive properties of this algorithm. Equation 3.14 takes care of non-linear deformations in the region lying in between and around the minutiae pairs. The condition $\frac{1}{2}(s_{i'} + s_{j'}) \geq \overline{s_l}$ ensures that, for registration of $\mathcal{F}_l$ with $\mathcal{F}_k$, a minutia pair having a high degree of authenticity in $\mathcal{F}_l$ is considered, thereby encouraging a real match and discouraging a faked one. In STEP 3, since there cannot lie two minutiae p_u and $p_{u'}$ within a distance of $\min(\lambda_u, \lambda_{u'})$ (if so, they have been already eliminated as spur or bridge endpoints in the minutiae detection stage), there cannot be more than one minutia in the database set that lies within the range query box. As a result, if there is (at most) one minutia $p_u \in \mathcal{S}_k$ lying within the query box, then p_u is considered as a valid match with the corresponding query $\tilde{q}_v$, provided they are of identical types, and their directions differ by at most $22\frac{1}{2}^0$. In Eqn. 3.16, the parameter ω represents the weightage attached to the difference of scores of two matching minutiae in their way of participating in the estimate of $\mu^{(k,l)}$. As explained in Sec. 3.3, the value of ω should lie between 0 and $\frac{101}{198}$ so that $\xi(s_u, \tilde{s}_v)$ is never negative. In our experiments, we have considered $\omega = \frac{1}{2}$.

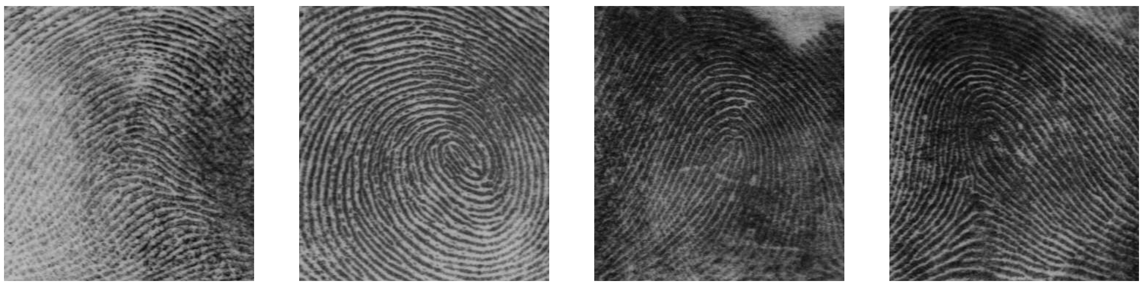
3.7 Experimental Results

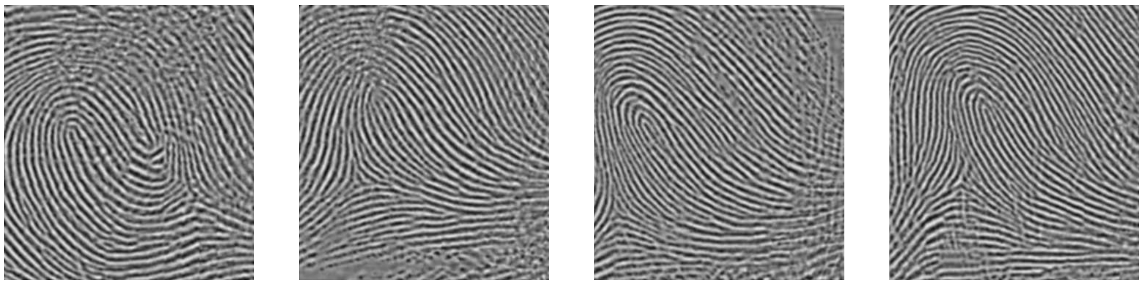
3.7.1 Results of Score Evaluation

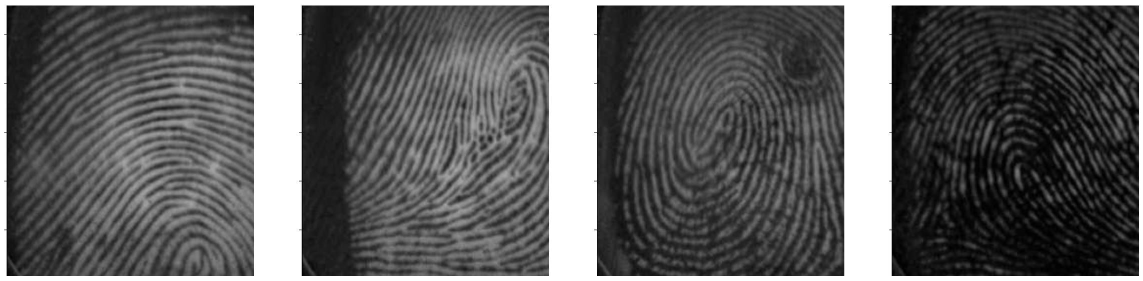
We used the fingerprint images from (i) NIST Special Database 4 [Watson and Wilson (1992)], (ii) NIST Special Database 14 [Candela *et al.* (1995)], (iii) Database B1 of FVC2000 [FVC 2000 (2000)], and (iv) Database B2 of FVC2000 [FVC 2000 (2000)]. Our experiments for evaluation of scores have been performed on 50 images of set (i), 124 images of set (ii), 80 images of set (iii), and 80 images of set (iv). Each image in these four sets is an 8-bit gray-scale image. The images of set (i) are used after applying Wavelet Scalar Quantization implemented in PCASYS [Candela *et al.* (1995)].

First, the input image is transformed to a skeletonized ternary image consisting of ridges, valleys, and backgrounds. In order to get the skeletonized ternary image from a gray-scale image, we have used the tool RiVEx developed by us [Project Group: ISI-Intel (2002b)]. For the detection of minutiae from a skeletonized binary image, consisting of ridges(1) against background(0), or valleys(1) against background(0), we have used another tool named as MinuBin [Project Group: ISI-Intel (2002a)]. Four gray-scale images — one from each of set (i), set (ii), set (iii), and set (iv) — are shown in Fig. 3.13. The images of set (i) and set (ii) are of size 480 × 512 each, those of set (iii) are of size 300 × 300, and those of set (iv) of size 364 × 256. All the images in these four sets are recorded at 500 dpi.

The results obtained over the four sets of images are presented in Table 3.1. In Ta-


NIST sdb-4.


NIST sdb-14.


FVC-2000 sdb-B1.

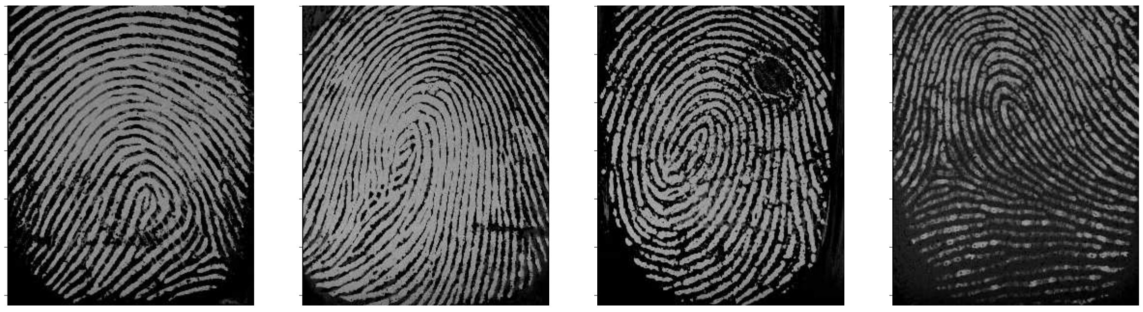

FVC-2000 sdb-B2.

Figure 3.13: Sample gray-scale images from 4 databases.

Table 3.1: Results for 4 sets of fingerprint images

Set	Database	No. of Imgs.	Image Size	Avg.no.of Minutiae	Mean Score	Std.dev. Score	Avg.time in sec.
i	NIST sdb-4	50	480×512	33	33.55	22.38	0.121
ii	NIST sdb-14	124	480×512	59	48.92	21.84	0.125
iii	FVC-2000 set-B db-1	80	300×300	16	32.46	19.13	0.045
iv	FVC-2000 set-B db-2	80	364×256	23	36.20	20.51	0.053

ble 3.1, column 3 indicates the number of images considered for generating the results shown. Column 5 indicates the average number of minutiae with positive scores, n_k, for kth set, vide Eqn. 3.18. Column 6 gives the *overall mean score*, μ_k, for the corresponding set of images, estimated from the *individual mean scores* of all the images in the set, μ_{kj}, in accordance with the Eqn. 3.19.

$$n_k = \frac{1}{m_k}\sum_{j=1}^{m_k} n_{kj} \tag{3.18}$$

$$\mu_k = \frac{\sum_{j=1}^{m_k} n_{kj}\,\mu_{kj}}{\sum_{j=1}^{m_k} n_{kj}} \tag{3.19}$$

where, m_k = total number of images in kth set, n_{kj} = number of minutiae with positive scores, and μ_{kj} is the mean score over all minutiae with positive scores, in jth image of kth set, $k \in \{i, ii, iii, iv\}$, as shown in the Eqn. 3.20.

$$\mu_{kj} = \frac{1}{n_{kj}}\sum_{i=1}^{n_{kj}} s_{kji} \tag{3.20}$$

where, s_{kji} is the (positive) score of ith minutiae in the jth image of kth set.

In table 3.1, column 7 displays the *overall standard deviation*, σ_k, for the corresponding set of images, vide Eqn. 3.21, estimated from the *individual standard deviations*, σ_{kj}, of all the images in the set, in accordance with the Eqn. 3.22. Average time for evaluating the scores of all minutiae per image of a database is given in column 8.

$$\sigma_k = \left[\frac{\sum_{j=1}^{m_k}(n_{kj}\,\sigma_{kj}^2 + n_{kj}\,\mu_{kj}^2)}{\sum_{j=1}^{m_k} n_{kj}} - \mu_k^2\right]^{\frac{1}{2}} \tag{3.21}$$

Table 3.2: Variation of score with different parameters

Image File	N	ν	γ	n_i	n_f	$\Delta n(\%)$	mean score
fpi.001.01	1.0	2.00	0.10	77	51	33.77	60.25
fpi.001.02	1.0	2.00	0.20	77	50	35.06	54.12
fpi.001.03	1.0	1.00	0.10	77	51	33.77	54.06
fpi.001.04	1.0	1.00	0.20	77	50	35.06	39.22
fpi.001.05	2.0	3.00	0.10	77	51	33.77	65.35
fpi.001.06	2.0	3.00	0.20	77	51	33.77	65.35
fpi.001.07	2.0	2.00	0.10	77	46	40.26	65.91
fpi.001.08	2.0	2.00	0.20	77	45	41.56	66.89
fpi.001.09	2.0	1.00	0.10	77	41	46.75	48.24
fpi.001.10	2.0	0.50	0.10	77	41	46.75	38.54
fpi.001.11	2.0	0.50	0.20	77	15	80.52	57.07
fpi.001.12	3.0	1.00	0.10	77	32	58.44	66.72
fpi.001.13	3.0	0.75	0.10	77	19	75.32	58.32
fpi.001.14	3.0	0.50	0.10	77	17	77.92	45.82

$$\sigma_{kj} = \left[\frac{1}{n_{kj}} \sum_{i=1}^{n_{kj}} (s_{kji} - \mu_{kj})^2 \right]^{\frac{1}{2}} \tag{3.22}$$

In the estimation of noise-based score, a number of parameters are involved. N decides the area covered by the window W. A higher value of N includes the distant contributors responsible for noise, whereas, a lower value often fails to incorporate the real noise contributors. Optimization of N is, therefore, a crucial factor. ν decides the noise threshold τ_{noise} that plays a vital role in deciding the noise level of the window W. The controlling parameter γ decides the influence of noise on the score. A higher value of γ will enforce a higher impact of noise in the score. Table 3.2 enumerates the roles played by the noise detection parameters. Different sets of values of these parameters have been chosen carefully to demonstrate their effects on the scores of minutiae for a ternary skeleton image fpi.001 exhibited in Fig. 3.14. In Table 3.2, the column with heading n_i indicates the initial number of minutiae that are present in the image fpi.001 (Fig. 3.14) prior to score evaluation. For each set of parameters, the corresponding final number of minutiae with positive scores is shown in column with heading n_f. The column with heading $\Delta n(\%)$ shows the percentage of difference of n_f from n_i. The images with (positively) scored minutiae for different sets of parameters are shown in Figs. 3.15(a)–(n). In all of these ternary images shown, the darker lines represent the ridges and the faint lines the valleys.

The score distribution for different images in the 4 sets of images is shown in Table 3.3. To show our experimental results on these sets, we have chosen $N = 2.00, \nu = 1.25, \gamma =$

Table 3.3: Score distribution for different images

Sl. no.	Image	Minutiae before score	Minutiae with (+)ve scores	Mean Score	Std.Dev. of Scores
1	nist.14.1	77	37	59.70	31.59
2	nist.14.2	79	40	58.67	32.25
3	nist.14.3	82	34	60.71	28.66
4	nist.14.4	111	66	62.12	28.05
5	nist.4.1	97	12	44.42	29.26
6	nist.4.2	95	28	35.50	24.43
7	nist.4.3	144	18	28.83	26.61
8	nist.4.4	111	24	33.88	25.25
9	fvc.b1.1	20	11	51.55	17.25
10	fvc.b1.2	34	19	38.74	21.87
11	fvc.b1.3	39	10	44.10	19.06
12	fvc.b1.4	45	11	24.36	28.33
13	fvc.b2.1	20	16	54.62	26.41
14	fvc.b2.2	13	11	51.68	28.01
15	fvc.b2.3	26	22	55.73	35.07
16	fvc.b2.4	25	14	62.64	28.93

0.10. The corresponding image for our chosen set of parameters with some score values written beside the corresponding minutiae is shown in Fig. 3.16, the darkness of a minutia being proportional to its score. Table 3.4 includes the scores (positive values only) of the bifurcation minutiae (BM), followed by those of the termination minutiae (TM), arranged in ascending orders.

Four typical minutiae out of the 77 minutiae of fpi.001 (Fig. 3.14) are selected along with their neighborhood regions of size 70 × 70 pixels, shown in Fig. 3.17, to clarify the three score components, $s_\mathrm{r}, s_\mathrm{v}$ and s_n, of these minutiae. Each of these minutiae is located in the center of its corresponding region in Fig. 3.17.

In the minutiae p_1 (type: bifurcation, x: 345, y: 414), each of the three score components is 0. This is because each of the three ridges incident at p_1 is less than λ in length, thereby failing to establish a positive ridge score for p_1, vide Sec. 3.4.1. Furthermore, the three ridges incident at p_1 may be very short and hence, it may not be possible to ascertain which one of them appears before bifurcation and which two are the bifurcated ridges. Hence, it is hard to calculate the local ridge angle at p_1 that is required as a guiding direction to find the valley termination minutia expected to be lying between the two bifurcated ridges. This makes the second score component, s_v, zero at p_1. Since $s_\mathrm{r} + s_\mathrm{v} = 0$, it is not meaningful to find the negative score imparted by noise, and therefore, by default,

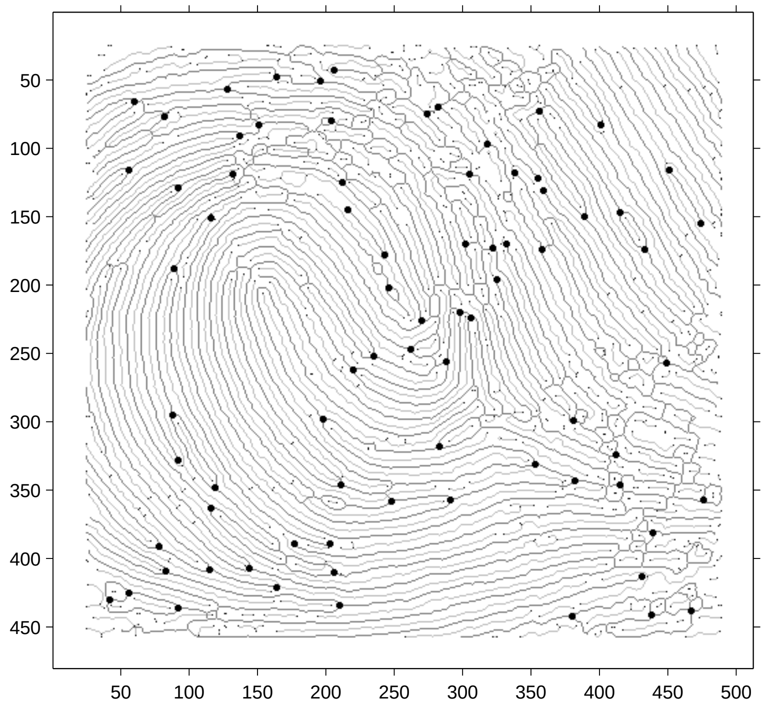
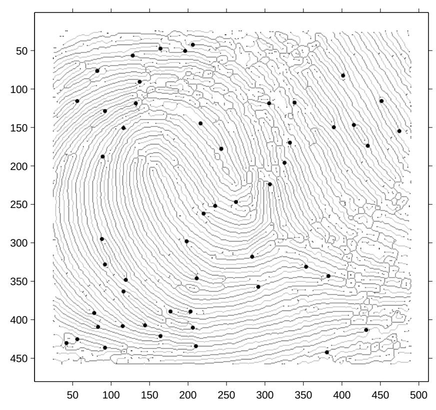
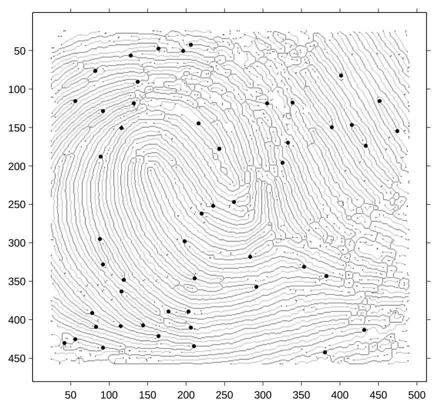


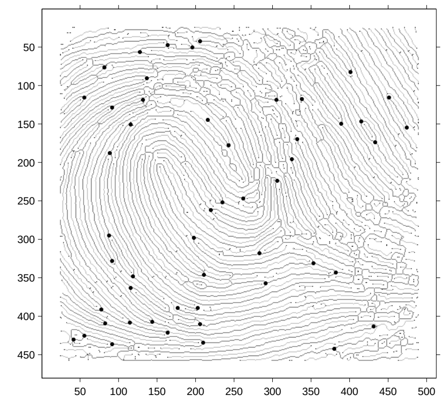
Figure 3.14: fpi.001 with 77 minutiae without scores.

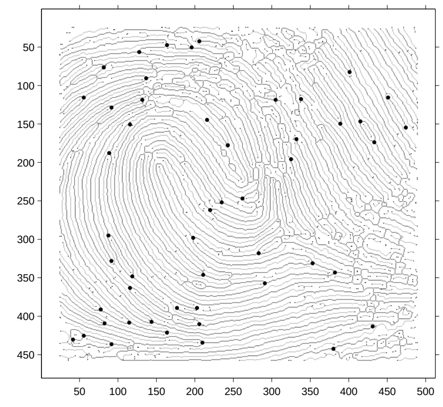


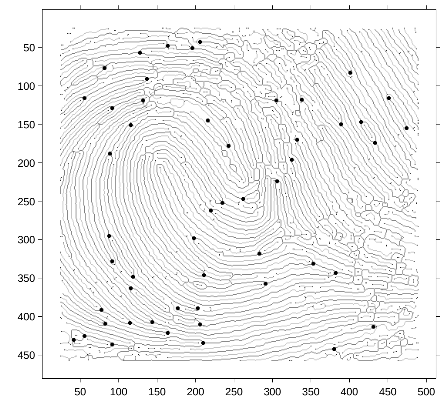
(a) fpi.001.01 with 51 minutiae

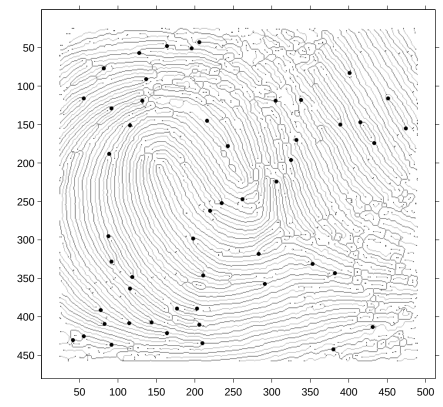


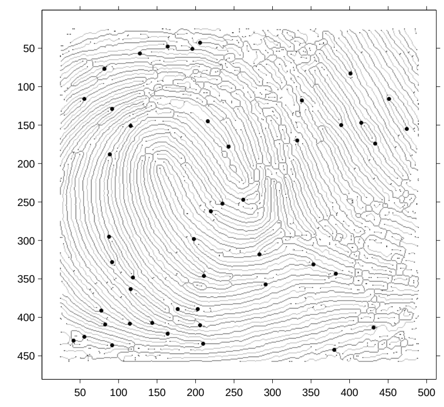
(b) fpi.001.02 with 50 minutiae

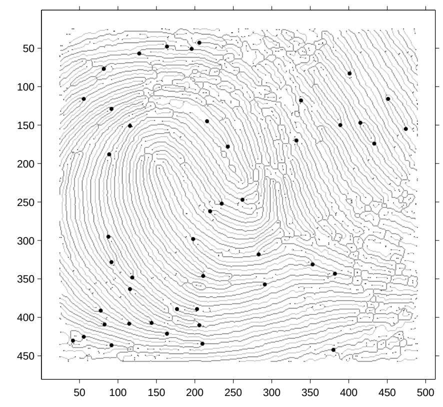

(c) fpi.001.03 with 51 minutiae

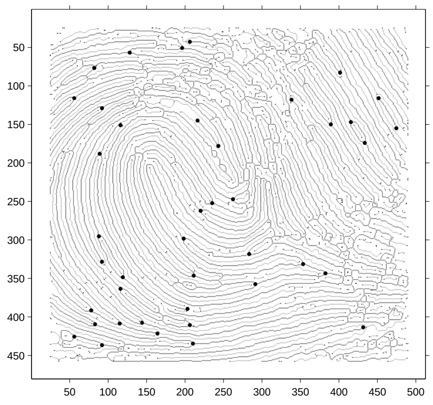

(d) fpi.001.04 with 50 minutiae

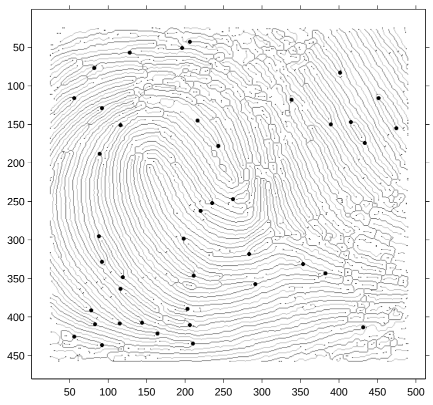

(e) fpi.001.05 with 51 minutiae

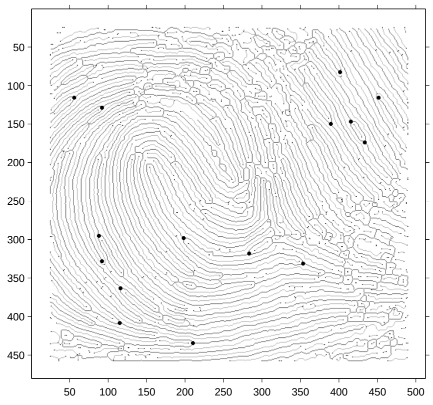

(f) fpi.001.06 with 51 minutiae

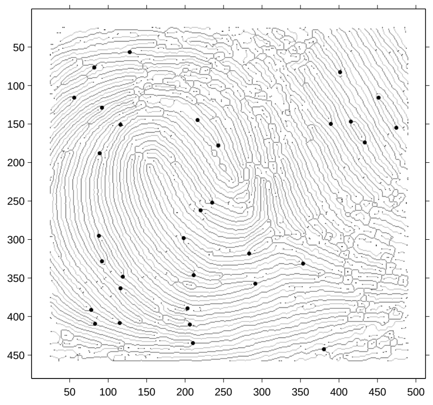

(g) fpi.001.07 with 46 minutiae

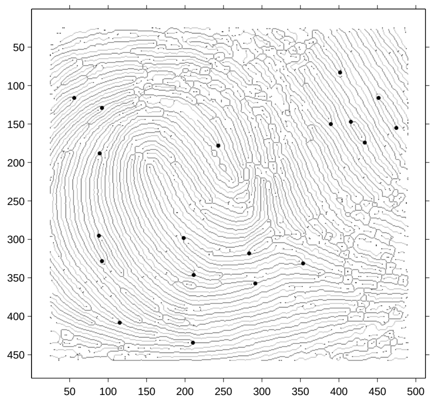

(h) fpi.001.08 with 45 minutiae


(i) fpi.001.09 with 41 minutiae


(j) fpi.001.10 with 41 minutiae


(k) fpi.001.11 with 15 minutiae


(l) fpi.001.12 with 32 minutiae


(m) fpi.001.13 with 19 minutiae

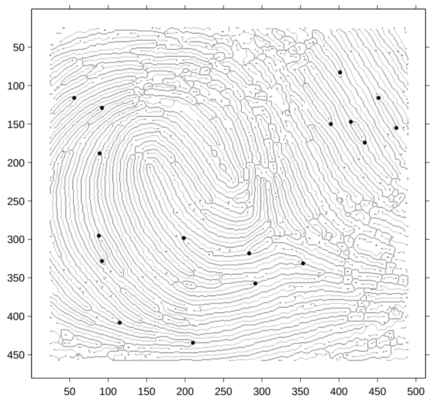

(n) fpi.001.14 with 17 minutiae

Figure 3.15: Minutiae with scores for different set of noise parameters (given in Table 3.2).

s_n becomes zero. Thus, the total score of minutia p_1 becomes 0, and it is not shown in Figs. 3.15(a)–3.15(n) and Fig. 3.16.

In the next minutiae p_2 (type: bifurcation, $x : 345, y : 210$) of Fig. 3.17, the first component of score, s_r, due to ridge topographical structure in and around p_2, is 34. But the valley termination minutia, supposed to be present near p_3 and within the two bifurcated ridges, is not present. In stead, the said valley termination minutia is lying outside the two bifurcated ridges. To measure the distances $d_{pp'}$, $d_{p''r_1}$, $d_{p''r_2}$, $d_{p''v_1}$, and $d_{p''v_2}$ in the set $\{d_{\text{BM},va}\}$, the said valley termination minutia is the basic prerequisite. So the score component due to valley topography, s_v, for p_2 is as low as 14. The noise level, η, in the window centered at p_2, falls short of the threshold noise, τ_{noise}, set for our chosen set of parameters, and therefore, s_n for p_2 turns to be zero. The total score of p_2 thus amounts to be $34 + 14 - 0 = 48$.

In the minutiae p_3 (type: bifurcation, x: 424, y: 55), s_r is found to be 35. This minutia has narrowly escaped to be marked as a loop minutia, since we have set that each of the edge lengths of a loop should be less than 2λ. The valley termination minutia is found to be present near p_3 between the two bifurcated ridges. Distances in set $\{d_{\text{BM},va}\}$ are measured, and s_v is found to be 39. However, it is evident from the figure that this minutia is located in a highly noisy area, and the score component s_n is found to be -61 as expected. The total score of p_3 thus becomes $35 + 39 - 61 = 13$.

In the fourth minutia p_4 (type: bifurcation, x: 297, y: 197), s_r and s_v are estimated to be 37 and 48 respectively, whereas, s_n is zero. Total score of p_4 thereby amounts to $37 + 48 - 0 = 85$. The topographic orderliness of the local neighborhood of p_4 clearly supports the authenticity of the score yielded by the procedure followed.

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 total CPU time for the evaluation of scores of all minutiae in a ternary skeletonized fingerprint image was found to be around 0.03 to 0.07 sec.

3.7.2 Results of Fingerprint Matching

We used fingerprint images from (i) NIST Special Database 4 [Watson and Wilson (1992)] and (ii) Database dB1a of FVC2000 [Maltoni *et al.* (2003)], whose details have been given in Sec. 3.7.1.

The matching algorithm is implemented in C on a Sun_Ultra 5_10, Sparc, 233 MHz, the OS being the SunOS Release 5.7 Generic. The average execution times for both the sets are given in Table 3.5. The times shown in braces are required when minutiae scores are not considered, which are substantially higher than those, shown outside braces, when scores are included. Fig. 3.18 shows the different Receiver Operating Characteristic (ROC) curves by plotting the Authentic Acceptance Rate (AAR) vs. the False Acceptance Rate (FAR), for the two databases. The firm lines (type 1) represent the ROC when the individual scores of both the database image and the pattern image are taken into consideration, whereas, the dotted lines (type 2) the ROC when the conventional method is followed without considering the local topological properties (scores) of the minutiae. Type 1 curves are obtained by choosing different threshold values on number of matching minutiae pairs (default value is 6 in our experiments) and on matching score (default $\min(\overline{s}_k, \overline{s}_l)$), and type 2 curves by different threshold values only on number of matching minutiae pairs.

3.8 Conclusion

This chapter describes a mechanism for assigning a score value to each of the extracted minutiae, based on its topographical properties. Further improvements on score evaluation might be achieved by designing an appropriate strategy for crosschecking the prominence of a minutia in the original gray-scale image and the smoothness of curvature of the ridgelines and valley lines in its local neighborhood in the ternary skeletonized image.

The score values obtained for a set of minutiae have been subsequently used to expedite a fingerprint matching process by a hierarchical arrangement of the minutiae based on the minutiae scores. Searching for a fingerprint match in a fingerprint database containing millions of records may take an excessive amount of time, which is reduced significantly by adopting a suitable cascading on the minutiae scores. Fingerprint indexing by Bhanu and Tan (2003) strengthens our assertion on minutiae scores for faster fingerprint recognition.

Improvements on execution time, AAR and FAR may be possible by further experiments on shape of the query box/region like the one in angular tree (Chapter 2), optimization on area covered under the query box, etc. Furthermore, research provision also lies in automatic thresholding of parameters, which play a crucial role in the score finding and in the matching process.

Table 3.4: Score values (s) of the minutiae detected for the fingerprint image in Fig. 3.16.

bifurcation minutiae (BM)					bifurcation minutiae (BM)					termination minutiae (TM)				
Sl. no.	x	y	θ	s	Sl. no.	x	y	θ	s	Sl. no.	x	y	θ	s
1	76	81	315	5	16	408	82	246	74	1	117	337	222	6
2	342	381	267	13	17	362	115	40	75	2	420	163	75	9
3	424	55	267	13	18	406	143	252	75	3	412	430	114	22
4	261	219	41	21	19	294	87	14	76	4	42	205	270	33
5	50	195	258	24	20	173	432	215	77	5	150	115	116	43
6	246	261	71	25	21	347	118	229	78	6	82	400	41	47
7	251	234	248	32	22	433	209	270	78	7	144	215	223	49
8	441	379	93	38	23	390	77	235	78	8	177	242	34	49
9	409	205	252	47	24	435	91	267	79	9	154	473	15	50
10	345	210	250	48	25	146	414	23	79	10	388	202	90	60
11	56	127	91	65	26	149	388	217	80	11	327	91	22	82
12	187	88	328	65	27	317	282	108	82	12	115	450	216	86
13	128	91	304	66	28	356	290	286	83					
14	407	114	251	67	29	330	352	254	84					
15	115	55	305	72	30	297	197	39	85					

Table 3.5: Matching results for 2 sets of fingerprint images

	NIST 4	FVC dB1
% of invalid images (< 6 minutiae)	6.00	4.36
Avg. time (secs.) for acceptance	0.10 (0.79)	0.06 (0.51)
Avg. time (secs.) for rejection	0.07 (1.02)	0.04 (0.64)

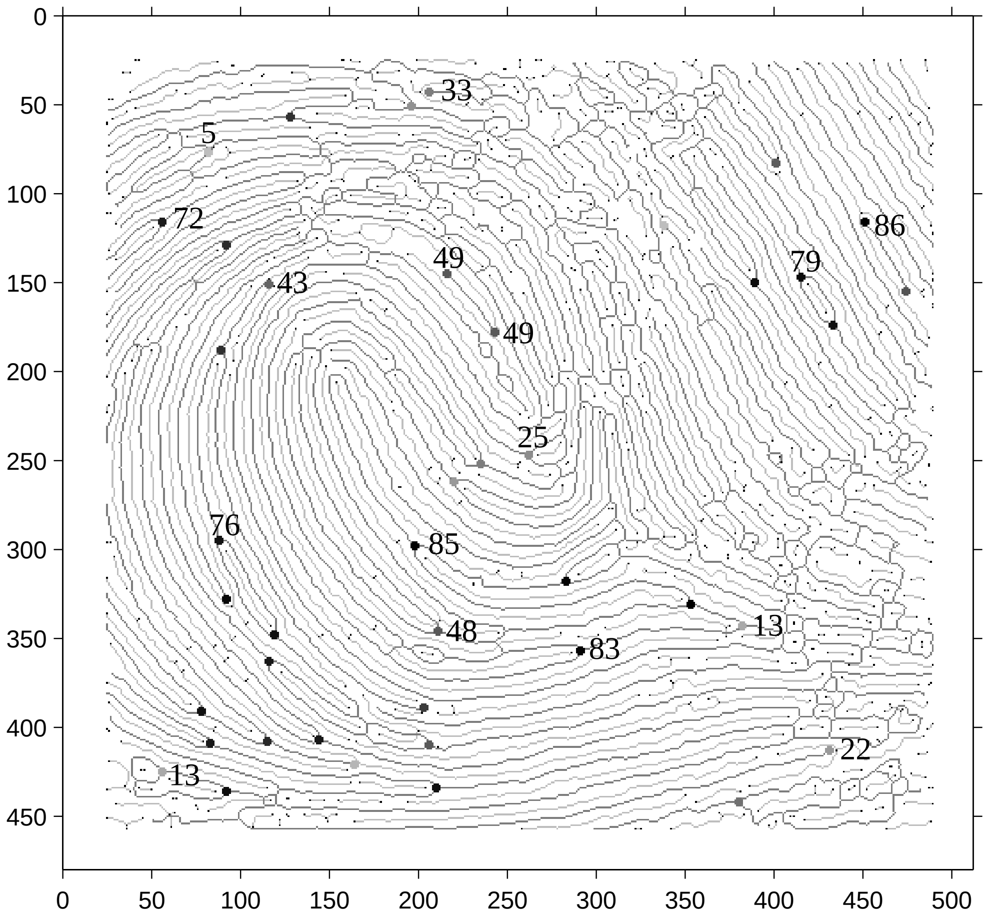
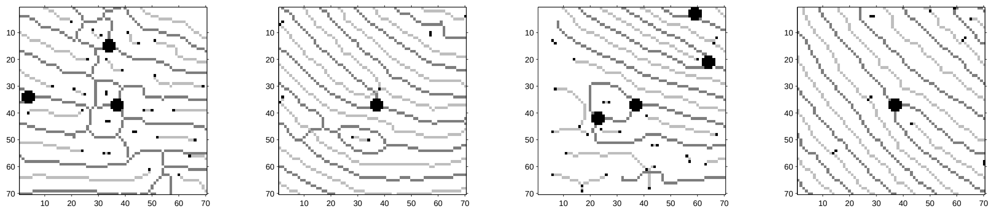


Figure 3.16: Minutiae shown for image fpi.001 with darkness proportional to scores, Scores of some typical minutiae have been written on the image for better understanding.



p_1 (0) p_2 (48) p_3 (13) p_4 (85)

Figure 3.17: Four sample minutiae, namely p_1, p_2, p_3, p_4, of image fpi.001 (Fig. 3.16) in their local neighborhoods.

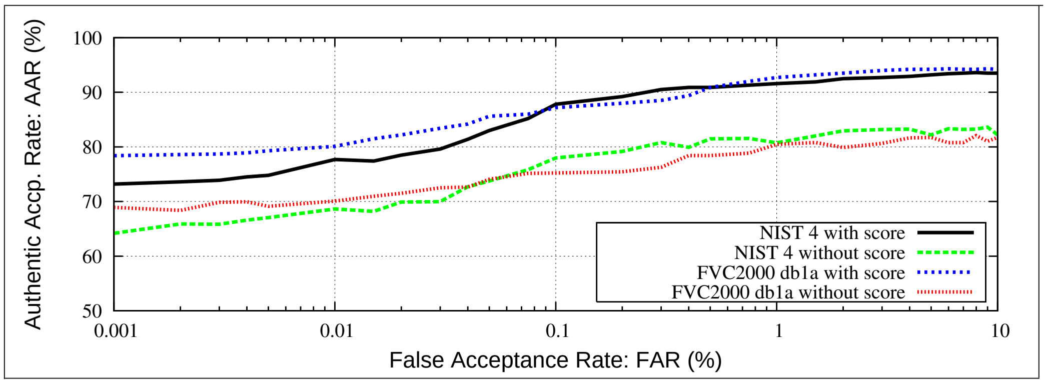


Figure 3.18: ROC curves of fingerprint matching for NIST 4 and FVC2000 dB1 images.

Chapter 4

Points Representing Corners in a General Digital Image

How can it be that mathematics, being after all a product of human thought, which is independent of experience, is so admirably appropriate to the objects of reality? Is human reason, then, without experience, merely by taking thought, able to fathom the properties of real things?

Albert Einstein
Sidelights on Relativity

4.1 Introduction

Corners are important geometric features of a digital image. Being sparse features, mere presence of them is considered sufficiently informative. Hence, much of the work on two-dimensional features of an image is focused on detection of corners. However, corners of an object, by themselves, just represent an ordinary set of points on a two-dimensional plane, which hardly begets any idea about the underlying object. The object becomes somewhat apparent once the directions of the incident edges are provided with each corner. In Fig. 4.1(a), a set of corners for a simple object is shown for illustration, which fails to yield a definite and conclusive impression about the related object. In Fig. 4.1(b), the set of corners along with the directions of the incident edges is shown, which produces a precise conception of the object, shown in Fig. 4.1(c). Hence, an application using corners will be more efficient if directions of edges incident at each corner are used to fortify the set of corners.

Detection of corners is often required in numerous applications in pattern recognition, computer vision, and image processing, some of which are as follows:

(i) shape analysis [Antoine *et al.* (1996), Biswas *et al.* (2005b), Buvaneswari and Naidu (1998), Pavlidis *et al.* (1997)]

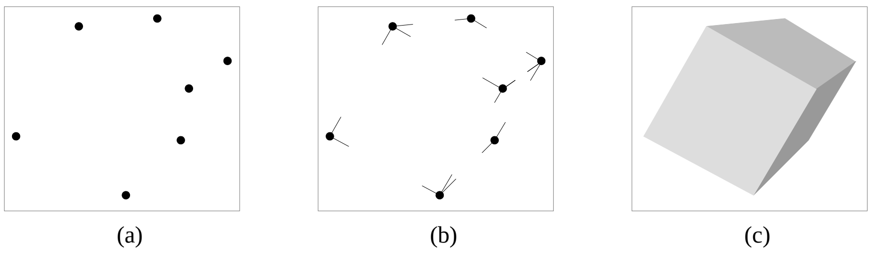


Figure 4.1: Only corners (in (a)) fail to suggest the nature of the underlying object, whereas corners along with the directions of the incident edges (in (b)) produce an impression of the actual object (in (c)).

(ii) automatic identification of events of interest, e.g. tracking and classification of moving vehicles [McCane *et al.* (2002), Mohanna and Mokhtarian (2003), Smith and Brady (1995), Zang and Klette (2003)]

(iii) optical flow computation [Barnard and Thomson (1980), Nagel (1987), Park and Han (1997), Smith and Brady (1995)]

(iv) 3D scene analysis and reconstruction from stereo image pairs [Burden and Bell (1997), Cecelja *et al.* (2003)]

(v) real-time automatic face tracking, face recognition, and motion correspondence [Gao (2004), Liang *et al.* (2003), Manjunath *et al.* (1992)]

(vi) determination of robot locations/trajectories using object/environment corners [Argyros *et al.* (2001), Blisset (1990), S.-Yuan and W.-Hsiang (1991), Tsay *et al.* (2003)]

(vii) retrieval of images and videos based on object corners [Ducksbury and Varga (1997), Gu and Tjahjadi (1999), Mokhtarian and Mohanna (2002), Wolf *et al.* (2000)]

Perception of corners by human visual system involves different kinds of information associated with an image region, such as brightness, contrast, clarity, rate of change of contour curvature, degree of straightness of the digital contour, etc. Now, in a gray-level image, corners are formed at boundaries between two or more significantly dissimilar image brightness regions, where the boundary curvature is sufficiently high. Using only boundary analysis based on the pattern of the digital curve, therefore, is very much preconditioned by the preceding segmentation. Use of only gray-level analysis based on derivatives often leads to acceptance of false corners and rejection of valid corners, as this method is susceptible to noise. Furthermore, in gray-level based approaches, the relative brightness parameter plays a critical role. An appreciable change of this parameter produces abrupt change in the set of detected corners. Hence, a good corner detection algorithm should satisfy a number of important criteria, which are as follows:

(i) Each true corner should be detected, i.e., no false negative should occur.
(ii) No false corner should be detected, i.e., no false positive should creep in.
(iii) Corner points should be well localized, and the estimated directions of incident edges at a corner point should lie within a reasonable tolerance.
(iv) Corner detector should be robust with respect to noise, i.e., number of false positives as well as number of false negatives should be minimal.
(v) Corner detector should be stable and efficient, i.e., the set of detected corners should not vary much with change in any control parameter, e.g. brightness threshold.

Corner detection can be broadly categorized as contour-based method or gray-level based method. In the contour-based method [Beus and Tiu (1987), Freeman and Davis (1977), Koplowitz and Plante (1995), Lee *et al.* (1995), Liu and Srinath (1990)], a segmented boundary is followed, and the rate of change of the contour angle is watched to detect and locate corners. Thus, this method is quite susceptible to the adopted segmentation procedures. As an improvement of this method on corner detection, CSS (curvature scale-space) method [Mokhtarian and Suomela (1998)] is proposed by Mokhtarian and Suomela, where corners are detected, tracked, and localized through the curvature analyses based on multiple scales. On the contrary, the gray-level based method has been suggested to detect corners directly from the gray-scale images without any prior segmentation. The use of angle-based templates is proposed by Rangarajan et al. [Rangarajan *et al.* (1989)]. There also exist gradient-based methods [Harris and Stephens (1988), Kitchen and Rosenfeld (1982), Lindeberg (1994)] for corner detection by identifying curvature changes by differential analysis without the need for prior segmentation. A literature survey by Zheng et al. [Zheng *et al.* (1999)] summarizes the existing gray-level corner detection methods. Discussions on the generic methodology for assessing the performance of corner detection algorithms and several case studies can be found in the existing literature [Rockett (2003)].

In recent times, several corner detection techniques have been proposed [Alkaabi and Deravi (2004), Banerjee *et al.* (2004), Elias and Laganiére (2002), Etou *et al.* (2002), Lüdtke *et al.* (2002), Mohanna and Mokhtarian (2002), Urdiales *et al.* (2003)]. One of the widely referred corner detection algorithms is SUSAN [Smith and Brady (1997)], where a circular mask of fixed diameter (7 pixels) is used to extract the local structural information based on the USAN area in order to judge the candidature of the mask's nucleus as a corner. The algorithm proposed here differs from SUSAN in several aspects, viz., adaptively augmenting the annular window depending on a number of conditions, design of an adaptive annular filtering scheme, adaptive thresholding of brightness parameter, and subpixel-precision evaluation of incident edge directions at each corner. A preliminary version of the algorithm has been discussed by Bhowmick and Bhattacharya (2004a) and by Bhowmick and Bhattacharya (2005b).

4.2 Proposed Algorithm

Let $I(i,j)$ be the gray-scale intensity of any point $p(i,j)$ in an eight-bit gray-scale digital image I with m rows and n columns. Then the gradient of intensity at the point $p(i,j)$ is given by two components in the right-hand side of Eqn. 4.1.

$$\nabla I(i,j) = \left\langle \frac{\partial I}{\partial x}(i,j), \frac{\partial I}{\partial y}(i,j) \right\rangle \tag{4.1}$$

The magnitude of $\nabla I(i,j)$ is often used as a measure of the strength of edge, if any, passing through (i,j), and can be expressed in several suitable forms, depending on the application, one of which is as follows (L_1 norm [Klette and Rosenfeld (2004a)]):

$$|\nabla I(i,j)| = \left|\frac{\partial I}{\partial x}(i,j)\right| + \left|\frac{\partial I}{\partial y}(i,j)\right| \tag{4.2}$$

The corresponding second order difference $\nabla^2 I(i,j)$ is used to find the zero-crossing across the edge, guided by the direction ϕ as given in Eqn. 4.3.

$$\phi = tan^{-1} \frac{\left|\frac{\partial I}{\partial x}(i,j)\right|}{\left|\frac{\partial I}{\partial y}(i,j)\right|} \tag{4.3}$$

Usage of above directional differences (first order and second order) is a traditional and popular approach to find the gray level topological features (viz. edges, corners, etc.) of an image, but this has some severe problems, some of which are as follows:

(i) *Inappropriate discretization of* $\nabla I(i,j)$ *and* $\nabla^2 I(i,j)$: In the discrete domain, evaluation of $\nabla I(i,j)$ and $\nabla^2 I(i,j)$ merely considers the neighboring pixels (in 4-neighborhood or 8-neighborhood) of (i,j), thereby weakening the relevance of their definitions in the real domain, especially for the case where the gray level transition in and around the concerned point (i,j) is spread over a larger region, which often leads to improper location and erratic detection of desired features.

(ii) *Zero crossing problem*: For the gray level transition associated with a ramp edge [Wang *et al.* (1996)], which is the edge commonly found in a natural gray-scale

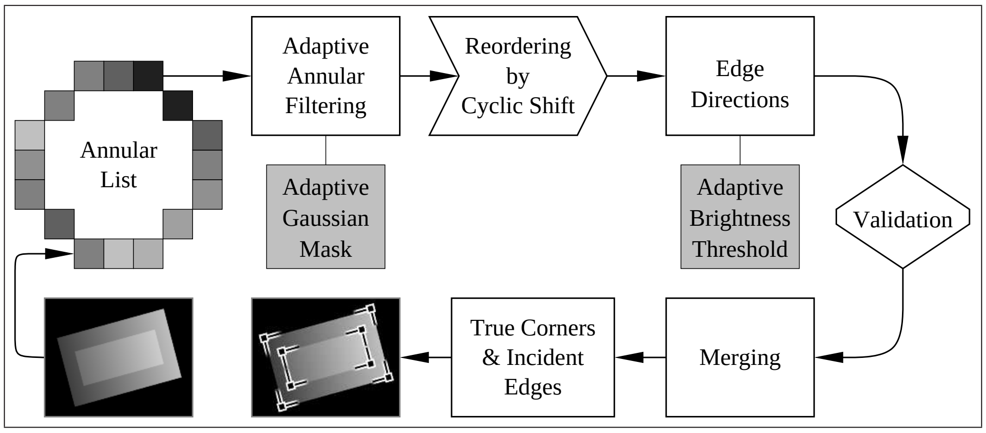


Figure 4.2: A schematic layout of the proposed algorithm CODE.

image, the zero crossing procedure at a point sometimes yields multiple solutions (when $|\nabla I(i,j)|$ is same for three or more consecutive pixels along the same "ramp") or a staggered edge point (when the zero crossing(s) lies away from the middle of the same "ramp"), posing further analysis in extracting the true edge point and the proper edge direction at the concerned point.

(iii) *Noise*: The presence of noise is a very common characteristic in a digital image, which, without any noise cleaning, when convoluted with the gradient operators, $\nabla I(i,j)$ and $\nabla^2 I(i,j)$, often produces impure results. The refinement process (e.g. mean filtering, Gaussian filtering, median filtering, etc.), in order to get rid of the noise from the image, in turn, blurs the image in general, and the edges in particular, thereby worsening the situation of detecting desired image features.

In order to circumvent the aforesaid problems, in this work, therefore, we have not used the directional difference operators for detecting the corners. We have not even applied any standard filtering on the input image, since any filtering mask of some defined size, say $w \times w$, (viz. 3×3, or, 5×5 Gaussian mask) may include, in worst case, the gray values of $(w^2 - w)$ non-edge pixels for a true edge pixel, which will affect our feature detection procedure. Instead, we have designed an adaptive annular filtering scheme as discussed in Sec. 4.2.1. In Sec. 4.2.2, the why and how of adaptive thresholding of gray value, necessary for accuracy and stability of the proposed algorithm, have been addressed. Sec. 4.2.3 elucidates the procedural details, raised to the finest level of precision, for finding out the directions of the incident edges at each corner point, which are put together in the form of a concise algorithm in Fig. 4.8. The experimental results and findings are

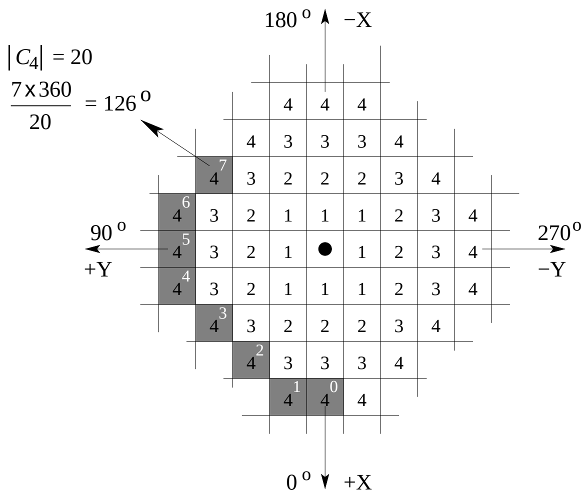


Figure 4.3: Annular lists $\mathcal{C}_1^{(x,y)}$, $\mathcal{C}_2^{(x,y)}$, …, where the pixels labeled by r are in the list $\mathcal{C}_r^{(x,y)}$ and $p(x, y)$ is shown by '•'.

given in Sec. 7.3, and finally, in Sec. 4.4, we have indicated the promises and prospects of using the corners along with their incident edge directions for further applications. In Fig. 4.2, a schematic diagram of CODE, the proposed algorithm for detection of COrners and Directions of incident Edges, is shown for an overall glimpse.

4.2.1 Adaptive Annular Filtering

Let $p(x, y)$ be any point in the image I, and $\mathcal{C}_r^{(x,y)}$ be the ordered list of pixels of the (digital) circle of radius r centered at p, enumerated in clockwise direction starting from $(x + r, y)$, as shown in Fig. 4.3. Let $|\mathcal{C}_r|$ be the number of pixels in $\mathcal{C}_r^{(x,y)}$, which will be independent of (x, y) and constant for a given value of r [Foley *et al.* (1993)]. It may be noted that, with increase in r, the digital perimeter $|\mathcal{C}_r|$ of the corresponding circle also increases monotonically, thereby improving its digital angular resolution, $360^0/|\mathcal{C}_r|$.

Now we define $\mathcal{A}_r^{(x,y)}$ as the ordered list (of size $|\mathcal{C}_r|$) of image gray values of the points in $\mathcal{C}_r^{(x,y)}$, such that for each k, $0 \leq k \leq |\mathcal{C}_r| - 1$, the kth entry in $\mathcal{A}_r^{(x,y)}$ is the gray value of the kth point in $\mathcal{C}_r^{(x,y)}$. In the implementation of our algorithm CODE, $|\mathcal{C}_r|$, and the angular tolerance in degrees, $(\pm)\alpha_r$, in accordance with Eqn. 4.4, are obtained from Look Up Table LUT-1, shown in Table 4.1, in order to reduce the execution time. It should be mentioned here that, structure of LUT-1 is independent of any of the image properties, and therefore, prepared once for all in the algorithm CODE.

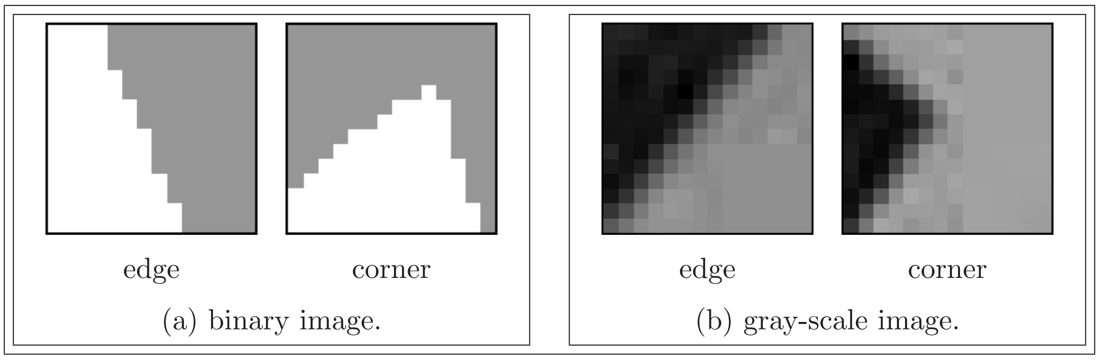


Figure 4.4: Typical examples of edges and corners in binary and gray-scale images, shown with enlarged pixels.

$$\alpha_r = \left\lfloor \frac{180}{|\mathcal{C}_r|} + \frac{1}{2} \right\rfloor \tag{4.4}$$

r	1	2	3	4	5
$\lvert\mathcal{C}_r\rvert$	8	12	16	20	28
α_r	23	15	11	9	6

Table 4.1: LUT-1 for $|\mathcal{C}_r|$ and α_r.

It may be observed that, if p lies on some edge or corner in a gray-scale image (Fig. 4.4(b)), or in a binary image (Fig. 4.4(a)), then the gray level transition across the respective edges would be reflected in the pattern of gray level values in $\mathcal{A}_r^{(x,y)}$, which can be exploited to extract the location and direction of the edge. Strengthening and appropriating the procedure of finding the concerned edge directions requires the reduction of noise present in $\mathcal{A}_r^{(x,y)}$, which is done by convoluting each element in $\mathcal{A}_r^{(x,y)}$ with a mask $\mathcal{W}_r = \boxed{\omega_1}\boxed{\omega_2}\boxed{\ldots}\boxed{\omega_{\lambda_r}}\boxed{\ldots}\boxed{\omega_2}\boxed{\omega_1}$ to create a filtered list of gray values $\mathcal{F}_r^{(x,y)} = \mathcal{W}_r * \mathcal{A}_r^{(x,y)}$, in accordance with the following equation.

$$\mathcal{F}_r^{(x,y)}(k) = \frac{\sum_{i=-\lambda_r}^{\lambda_r} \mathcal{W}_r(\lambda_r + i) \cdot \mathcal{A}_r^{(x,y)}(k+i)}{\sum_{i=-\lambda_r}^{\lambda_r} \mathcal{W}_r(\lambda_r + i)} \tag{4.5}$$

where, $\mathcal{A}_r^{(x,y)}(k+i)$ denotes the $((k+i) \texttt{ mod } |\mathcal{C}_r|)$-th element in $\mathcal{A}_r^{(x,y)}$, for $k = 0, 1, \ldots, |\mathcal{C}_r| - 1$. Since the first order difference of the gray value transition along the maximum

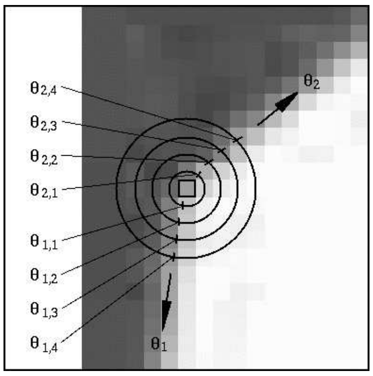


Figure 4.5: An example of finding the directions (θ_1, θ_2) of incident edges at a point $p(x, y)$ (shown by $\square$) using the angles $\{\theta_{1,1}, \theta_{1,2}, \ldots\}$ and $\{\theta_{2,1}, \theta_{2,2}, \ldots\}$ extracted from filtered annular lists $\mathcal{F}_1^{(x,y)}$, $\mathcal{F}_2^{(x,y)}$, ...

gray value gradient for a ramp edge resembles the one-dimensional Gaussian function, we resort to the Gaussian mask $\mathcal{W}_r$ of size $2\lambda_r + 1$. The speciality of the mask $\mathcal{W}_r$ adopted in our algorithm is that, with increase in the value of r, i.e., higher augmentation of $\mathcal{C}_r^{(x,y)}$, the size of $\mathcal{W}_r = 2\lambda_r + 1$ is also increased. This is to suit the slant of the ramp edge, since a lower value of r encompasses a part of the gray value transition, whereas a sufficiently high value of r ensures the inclusion of the entire gray value transition of the edge in $\mathcal{C}_r^{(x,y)}$. A sample image is cropped and magnified as an example in Fig. 4.5 that demonstrates the increase in the length of the ramp edge, where, there exist two ramp edges, whose ramp lengths are given in Table 4.2.

		ramp length					gray value trans.	
		edge-1		edge-2			edge-1	edge-2
r	$\|\mathcal{C}_r\|$	$\mathcal{A}_r^{(x,y)}$	$\mathcal{F}_r^{(x,y)}$	$\mathcal{A}_r^{(x,y)}$	$\mathcal{F}_r^{(x,y)}$	λ_r	$\mathcal{F}_r^{(x,y)}$	$\mathcal{F}_r^{(x,y)}$
1	8	4	4	4	4	0	92	103
2	12	6	7	5	7	1	141	133
3	16	6	7	6	6	1	165	172
4	20	5	10	5	6	2	177	175
5	28	5	9	5	6	2	177	174

Table 4.2: Gray value ramp lengths and gray value transitions for different values of r in Fig. 4.5.

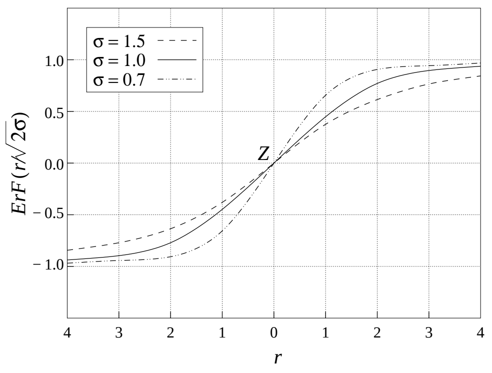


Figure 4.6: Change of $ErF(r/\sqrt{2}\sigma)$ versus r, which corresponds to change of gray level for a ramp edge with annular radius r.

4.2.2 Adaptive Brightness Thresholding

In a natural image, the gray-level value around an edge point or a corner point along the maximum gradient direction changes gradually. Thus, larger the augmenting window radius r centered at the corresponding point is, higher is the change of gray value across an edge, within some suitable range of r. Fig. 4.6 justifies the need for adaptively changing the brightness threshold γ_r with r. If Z represents the zero-crossing in a ramp edge as shown in Fig. 4.6, then γ_r is practically divided into two equal parts, one above the abscissa line through Z and the other below it. As a result, the value of $\frac{1}{2}\gamma_r$ (γ_r, there of) increases with the window radius r, so that γ_r is minimum for $r = 1$ and has no appreciable change when r exceeds 3 or 4. Hence the brightness threshold has to be adjusted appropriately in order to judge the candidature of a point as a corner.

Now, an ideal ramp edge is the integration of one-dimensional Gaussian function [Chanda and Dutta Majumder (1999)]. It may be noted that, integrals of the form $\int x^n e^{-kx^2} dx$, $n = 0, 2, 4, \ldots$, are referred to as "Gaussian" integrals (or integrals of Gaussian functions). It can be shown that the prototype of all such Gaussian integrals is $\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$, which can be derived using the fact that $\int_0^{\infty} e^{-u} du = 1$. It may be also noted that, an Error Function, $ErF(x)$, is defined in integral calculus by the *finite integral*, given in Eqn. 4.6, such that $ErF(\infty) = 1$.[1]

$$ErF(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-u^2} du \tag{4.6}$$

[1] Other integrals can be written in terms of this basic integral. For example, $\frac{2}{\sqrt{\pi}} \int_0^x e^{-ku^2} du = \frac{1}{\sqrt{k}} ErF(x\sqrt{k})$.

Further, the finite integral function $ErF(x)$ cannot be evaluated in terms of elementary functions, defined in the real domain, and therefore, proper approximating techniques must be used. One such technique is Maclaurin series expansion [Faires and Burden (1998)], from where we approximate the integral form of $ErF(x)$, given in Eqn. 4.6, to an well-approximated form given in Eqn. 4.7.

$$\begin{aligned} ErF(x) &= \tfrac{2}{\sqrt{\pi}} \sum_{k=0}^{\infty} \tfrac{(-1)^k x^{2k+1}}{(2k+1)k!} \\ &= \tfrac{2}{\sqrt{\pi}} \left(x - \tfrac{1}{3}x^3 + \tfrac{1}{10}x^5 - \tfrac{1}{42}x^7 + \tfrac{1}{216}x^9 - \tfrac{1}{1320}x^{11} + \ldots \right) \end{aligned} \tag{4.7}$$

The value of the Error Function, obtained using Eqn. 4.7, correct up to three decimal places, is considered for determining the pattern of the gray value transition associated with a ramp edge. Now, on substituting x by $r/\sqrt{2\sigma}$ in Eqn. 4.7, we get the ideal form of ramp edge whose first derivative obeys the Gaussian form. Fig. 4.6 shows the plot of $ErF(r/\sqrt{2\sigma})$ versus r, for different values of σ, where r signifies the distance of the concerned point from the zero-crossing ($r = 0$).

It is evident from Fig. 4.6 that, with a low value of σ, the corresponding gray-value transition associated with a ramp edge is gentle and imperceptible, whereas, with a high value of σ, the transition becomes very pronounced. A judicious selection of σ is, therefore, mandatory in order to detect the true edge transitions.

Now, the gray-level change in an annular window corresponds to a valid edge transition, if the concerned gray-level change exceeds the gray-level threshold associated with the window radius. Higher the window radius, higher would be the gray-level threshold. In line with the nature in change of gray-level value associated with a ramp edge, as shown in Fig. 4.6, therefore, we have fixed the gray-level threshold adaptively with the window radius, such that the gray-level threshold never exceeds the maximum gray-level threshold, γ_∞, which is the sole parameter supplied by the user.

Based on the observation that an entire edge transition along the maximum gradient direction (i.e. ramp length as shown in Table 4.2) gets captured in the annular list $\mathcal{A}_r$ for $r = 3$ or 4, we have considered $\sigma = 1$ in our experiments. It may be noted that implementation of Eqn. 4.7 is realized by a look up table LUT-2, which is prepared only once for an image for a given value of brightness threshold, γ_∞, as shown in Table 4.3.

r	1	2	3	4	5
γ_r	19	32	36	38	38

Table 4.3: An instance of LUT-2 for $\gamma_\infty = 40$ and $\sigma = 1$.

4.2.3 Estimation of Edge Directions

Another important feature of the proposed work is the estimation of edge direction by mean weighted difference. This method of finding edge direction works superbly for both natural and synthetic images, which usually possess ramp and step edge transitions respectively. For step edges, usual directional derivatives $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ provide the necessary edge directions. However, for ramp edges with low $|\nabla I|$, the derivatives produce erratic edge directions, since calculation of $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ deals with only 8×8 neighborhood of the concerned point and the entire transition pattern along the maximum gray value gradient is not considered while estimating the edge direction. In this approach, we have taken into consideration the entire gray-level transition corresponding to an edge and, therefore, the estimated edge direction precisely matches the true edge transition.

We define a function φ to reorder the annular list $\mathfrak{F}_r^{(x,y)}$ by a cyclic forward shift s to produce a new list $\widetilde{\mathfrak{F}}_r^{(x,y)}$ in order to expedite the edge extraction procedure as follows.

$$\varphi : \{k\}_{k=0}^{|\mathcal{C}_r|-1} \mapsto \{(k+s) \text{ mod } |\mathcal{C}_r|\}_{k=0}^{|\mathcal{C}_r|-1} \tag{4.8}$$

such that the following two conditions, (**c1**) and (**c2**), are satisfied simultaneously.

(**c1**) $\langle \mathfrak{F}_r^{(x,y)}(k)\rangle_0^{|\mathcal{C}_r|-1} = \langle \widetilde{\mathfrak{F}}_r^{(x,y)}(k+s) \text{ mod } |\mathcal{C}_r|\rangle_{k=0}^{|\mathcal{C}_r|-1}$,
where, $0 \le s \le |\mathcal{C}_r| - 1$.

(**c2**) either $\left(\delta(\widetilde{\mathfrak{F}}_r^{(x,y)}, 0, 1) \textit{ and } \delta(\widetilde{\mathfrak{F}}_r^{(x,y)}, |\mathcal{C}_r|-1, 0)\right) = 0$,
or $\texttt{sign}\left(\delta(\widetilde{\mathfrak{F}}_r^{(x,y)}, 0, 1)\right) \neq \texttt{sign}\left(\delta(\widetilde{\mathfrak{F}}_r^{(x,y)}, |\mathcal{C}_r|-1, 0)\right)$,
where, $\delta(\widetilde{\mathfrak{F}}_r^{(x,y)}, p, q) = \widetilde{\mathfrak{F}}_r^{(x,y)}(p) - \widetilde{\mathfrak{F}}_r^{(x,y)}(q)$,

$$\text{and } \texttt{sign}\,(a) = \begin{cases} -1 & \text{if } a < 0; \\ 1 & \text{if } a > 0; \\ 0 & \text{if } a = 0. \end{cases}$$

Let $\langle \widetilde{\mathfrak{F}}_r^{(x,y)}(p_{u,t}) : 1 \le t \le e_u\rangle$ be the uth monotonically ascending (or, monotonically descending) subsequence of length e_u in $\widetilde{\mathfrak{F}}_r^{(x,y)}$. Then $\langle \widetilde{\mathfrak{F}}_r^{(x,y)}(p_{u,t}) : 1 \le t \le e_u\rangle$ corresponds to the gray value transition for a valid edge if both the following conditions (**c3**) and (**c4**) are true.

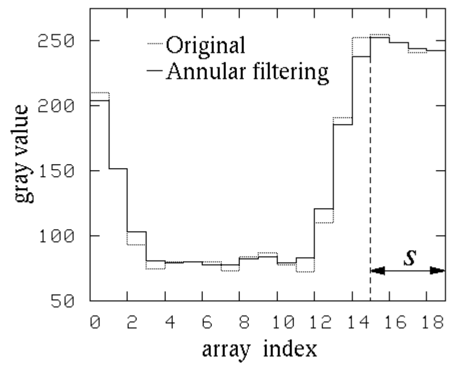
(**c3**) In $\widetilde{\mathfrak{F}}_r^{(x,y)}$, there exists no monotonically ascending subsequence of length exceeding e_u that contains $\langle \widetilde{\mathfrak{F}}_r^{(x,y)}(p_{u,t}) : 1 \leq t \leq e_u \rangle$.

(**c4**) $\left| \delta(\widetilde{\mathfrak{F}}_r^{(x,y)}, p_{u,e_u}, p_{u,1}) \right| \geq \gamma_r$.

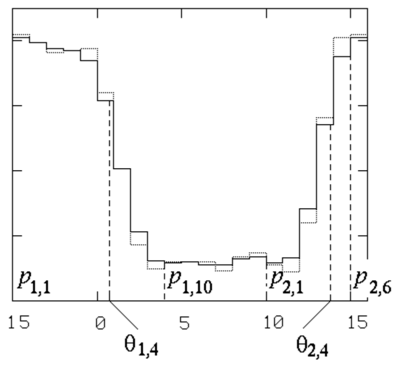
As there are $|\mathcal{C}_r|$ pixels in the annular list $\widetilde{\mathfrak{F}}_r^{(x,y)}$, the angular resolution at the center $p(x,y)$ of the digital circle $\mathcal{C}_r^{(x,y)}$ is $\frac{360^0}{|\mathcal{C}_r|}$. So, the angle made by the radius vector joining the point $\mathcal{C}_r^{(x,y)}(t)$ (please see Fig. 4.3) with the +ve x-axis measured in clockwise direction is given by $\frac{t \cdot 360^0}{|\mathcal{C}_r|}$. Further, it may be observed that there are e_u discrete gray-values in the subsequence $\langle \widetilde{\mathfrak{F}}_r^{(x,y)}(p_{u,t}) : 1 \leq t \leq e_u$, i.e. $(e_u - 1)$ gray value differences, which have a cumulative effect on the value $\delta(\widetilde{\mathfrak{F}}_r^{(x,y)}, p_{u,e_u}, p_{u,1})$ of overall gray-level transition corresponding to uth edge incident at p. The dominant ones out of these $(e_u - 1)$ differences will outweigh the others in their way of deciding the resultant edge direction. Hence, the edge direction corresponding to the subsequence $\langle \widetilde{\mathfrak{F}}_r^{(x,y)}(p_{u,t}) : 1 \leq t \leq e_u \rangle$ is given by

$$\theta_{u,r} = \left\lfloor \frac{360^0}{|\mathcal{C}_r|} \left(\sum_{t=1}^{e_u - 1} \frac{q_{u,r}^{(t)}}{\delta(\widetilde{\mathfrak{F}}_r^{(x,y)}, p_{u,e_u}, p_{u,1})} - s \right) + \frac{1}{2} \right\rfloor \tag{4.9}$$

where, $q_{u,r}^{(t)} = \frac{1}{2}(p_{u,t+1} + p_{u,t}) \cdot \delta(\widetilde{\mathfrak{F}}_r^{(x,y)}, p_{u,t+1}, p_{u,t})$.



(a) $\mathcal{A}_4^{(x,y)}$ and $\mathfrak{F}_4^{(x,y)}$



(b) $\widetilde{\mathfrak{F}}_4^{(x,y)}$

Figure 4.7: An example of finding the directions $(\theta_{1,4}, \theta_{2,4})$ of incident edges at $p(x,y)$ in Fig. 4.5 using the filtered annular list $\widetilde{\mathfrak{F}}_4^{(x,y)}$.

In Fig. 4.7(a), an instance for $r = 4$ is shown for the sample given in Fig. 4.5. The minimum cyclic shift s, shown in Fig. 4.7(a), is required to produce the new (cyclic shifted) annular list $\widetilde{\mathcal{F}}_4^{(x,y)}$ shown in Fig. 4.7(b). The list $\widetilde{\mathcal{F}}_4^{(x,y)}$ contains one descending subsequence $\{15, 16, \ldots, 19, 0, 1, \ldots, 4\}$ and one ascending subsequence $\{10, 11, \ldots, 15\}$ (considering the indices in the original list $\mathcal{F}_4^{(x,y)}$) that satisfy conditions (**c3**) and (**c4**). Hence these two subsequences correspond to the angles $\theta_{1,4}$ and $\theta_{2,4}$, shown by their respective index values in Fig. 4.7(b), and actual angles overlaid in Fig. 4.5, evaluated from Eqn. 4.9. It may be noted that, the last entry of the last valid subsequence (conforming to conditions (**c3**) and (**c4**)) may be the first element in $\widetilde{\mathcal{F}}_4^{(x,y)}$, which should be checked exclusively, which occurs when the start pixel of the first edge transition coincides with the end pixel of the last edge transition (e.g., in Fig. 4.7(b)).

4.2.4 Validation of Corners

If a point $p(x, y)$ is a true corner, then it should possess at least two edges incident upon it. Further, if there are exactly two edges incident upon p, then their directions should differ by an amount that falls "sufficiently" short of 180^o; because a difference of 180^o or so indicates that p is not a corner but an ordinary edge point. In addition, for each edge incident at p, the corresponding direction estimated for a particular window radius r should match with that estimated for each other window radius r'; to be precise, the condition $|\theta_{u,r+1} - \theta_{u,r}| \leq \alpha_r$ should hold true for $r = 1, 2, 3, \ldots$, in order that p is a valid corner. It may be noted that α_r decides the allowance that determines whether two (digital) angles match each other, which has been pre-calculated once for all and stored in LUT-1 (vide Table 4.1).

Considering the above facts, the necessary and sufficient conditions that validate p as a true corner are as follows:

(**c5**) $n_{r+1} = n_r, \forall r = 1, 2, \ldots, r_{\max} - 1;$

(**c6**) $|\theta_{u,r+1} - \theta_{u,r}| \leq \alpha_r, \forall u = 1, 2, \ldots, n_r, \forall r = 1, 2, \ldots, r_{\max} - 1;$

where n_r denotes the number of distinct edges incident at p, as detected from $\widetilde{\mathcal{F}}_r^{(x,y)}$, and $r_{\max}$ the maximum radius of the annular window. For testing the algorithm, we have considered $r_{\max} = 4$ for all images.

Now, if p is validated as a corner, then the direction of each edge incident at p is assigned as the average of the estimated angles over all window radii.

An example of a valid corner is shown previously, in Fig. 4.5, in which the point p

gets validated as a true corner. The ordered set of estimated angles for the first edge is $\langle\theta_{1,r}\rangle_{r=1}^{r=4} = \langle 224, 225, 225, 227\rangle$, and that for the second edge is $\langle\theta_{2,r}\rangle_{r=1}^{r=4} = \langle 12, 13, 13, 12\rangle$. Since for each of the two edges, the estimated angles for consecutive radii are well within the respective angular tolerance (α_r), p becomes a valid corner, and the directions of its two edges are given by 225^o and 13^o respectively.

```
Algorithm CODE($I$)
Steps:
 1. initialize: $\alpha_r; \gamma_r, \forall\, r = 1, 2, \ldots, r_{\max}$; ▷ from LUT-1, LUT-2
            $P \leftarrow \emptyset$ ▷ set of corners
 2. for each point $p(x, y) \in I$
 3.     for $r \leftarrow 1, 2, \ldots, r_{\max}$
 4.         construct $\mathcal{A}_r^{(x,y)}$ ▷ please see Sec. 4.2.1
 5.         $\mathfrak{F}_r^{(x,y)} \leftarrow \mathcal{W}_r * \mathcal{A}_r^{(x,y)}$ ▷ using Eqn. 4.5
 6.         construct $\widetilde{\mathfrak{F}}_r^{(x,y)}$ from $\mathfrak{F}_r^{(x,y)}$ ▷ conforming to c1 & c2
 7.         construct $\{\theta_{u,r}\}_{u=1}^{n_r}$ from $\widetilde{\mathfrak{F}}_r^{(x,y)}$ ▷ conforming to c3 & c4, using Eqn. 4.9
 8.         if $r = 1$
 9.             if $(n_1 < 2)$ or $(n_1 = 2$ and acute$(\theta_{1,1}, \theta_{2,1}) \geq 180^0 - \alpha_1)$
10.                 then goto step 2 ▷ $p$ is not a corner
11.         else
12.             if $n_r \neq n_1$
13.                 then goto step 2
14.             if $|\,\theta_{u,r} - \theta_{u,r-1}| > \alpha_r$ for any $u, 1 \leq u \leq n_r$,
15.                 then goto step 2
16.     $\theta_u \leftarrow \frac{1}{r_{\max}} \sum_{r=1}^{r_{\max}} \theta_{u,r}, \forall\, u = 1, 2, \ldots, n_1$
17.     $P \leftarrow P \cup \{p\}$
18. apply merging for each cluster of corners
```

Figure 4.8: Algorithm CODE to detect the corners in the input gray-scale image I.

4.2.5 Merging of Corners

For each cluster of corners accumulated in and around a true corner, selection of the most appropriate corner and rejection of the rest is done in the final stage to derive the proper set of corners for a given image. We adopt a simple and fast procedure to perform this task, which is as follows.

At first, the closely located corners are clustered together using their estimated edge directions. The strategy for doing this is that, a corner p is put to the cluster $\mathcal{C}$, provided
(i) the number of edges incident at p is same as that of each corner in $\mathcal{C}$,
(ii) p lies in the 8-neighborhood of some other corner Q in $\mathcal{C}$, and
(iii) the direction of each edge of p matches with that of the corresponding edge of Q (considering a tolerance of $\alpha_1 = 23^o$).

Next, for each cluster the best representative corner is chosen as the one having minimum variance of edge directions for $1 \leq r \leq r_{\max}$. For a faster implementation, here we have not used the statistical variance; instead, we have used the sum of differences of the estimated edge directions (angles) over all pairs of annular windows with radius r, $1 \leq r \leq r_{\max}$, for all edges incident at p, which is given by

$$\Sigma d\theta(p) = \sum_{u=1}^{n} \sum_{r_i=1}^{r_{\max}-1} \sum_{r_j=r_i+1}^{r_{\max}} \left|\theta_{u,r_i} - \theta_{u,r_j}\right| \tag{4.10}$$

where n is the number of edges incident at p. The best corner in a cluster is one that has minimum value of $\Sigma d\theta$ in the corresponding cluster. In case of ties, the corner having minimum value of $\Sigma d\theta$ and located closest to the geometric center of the cluster is chosen, and if any further tie occurs at this level (which is very rare in practice), then it is resolved arbitrarily.

The various stages of the algorithm CODE being explained above, the major steps of the algorithm are presented now in Fig. 4.8.

4.3 Experimental Results

We have implemented our algorithm in C in SunOS Release 5.7 Generic of Sun_Ultra 5_10, Sparc, 233 MHz, and compared its performance with SUSAN [Smith and Brady (1997)], the mostly referred algorithm in recent times. Fig. 4.9 (top row) shows the image ("test-1") of a rendered 3D object, which does not have any noise. In Fig. 4.9 (second row), the corners detected by SUSAN have been shown, which are same as those detected by the proposed algorithm (CODE). It also shows the estimated directions of the incident edges for all the corners detected, which are found to have very good accuracy with respect to the actual edge directions.

In order to demonstrate the robustness of the proposed algorithm with respect to noise, we have also shown here the results on a sequence of noise-affected images for the object "test-1". Random Gaussian noise with SNR (signal-to-noise ratio), varying from 0.01 to 0.50, has been applied on "test-1", and the corresponding set of detected corners has been compared with the ground-truth set, shown in Fig. 4.9. A few sets for some selected values of SNR have been presented in Fig. 4.9. From the sequence of these images, it may be observed that, with increase in SNR, the plentitude of false corners detected by SUSAN goes on rising abnormally compared to CODE. Further, in SUSAN, the set of false corners is mostly contributed by false positives, which would have a negative effect on subsequent applications using corners. On the contrary, in CODE, false positives as well as false negatives are much less; this indicates the robustness of its inherent methodologies. Fig. 4.10(a) shows the plot on number of false corners (false negatives, false positives, and total), detected by SUSAN, versus noise level (SNR) for "test-1" image, whereas, Fig. 4.10(b) shows the same for CODE, which clearly demonstrates the robustness of CODE to the presence of noise in an image.

Fig. 4.11(a) shows the detected corners on a synthetic image, "test-2", using SUSAN, which are identical with those detected by CODE, shown in Fig. 4.11(c), where the brightness (gray-value) threshold for both SUSAN and CODE (γ_∞) is fixed at 10. The corners by CODE, before merging, are shown in Fig. 4.11(b), to show the compactness of a cluster of the preliminary corners in and around a true corner. The directions of the incident edges on the corners detected by CODE, shown in Fig. 4.11(d), once again depicts the precision of the method for estimating the edge directions in the proposed algorithm.

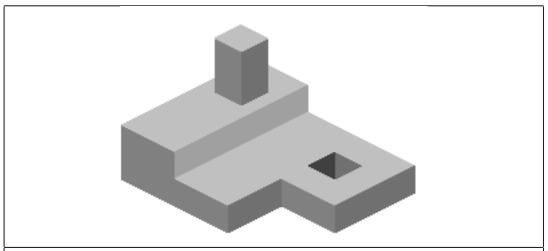

"test-1": rendered image of a 3D object

Fig. 4.9: continued to next page.

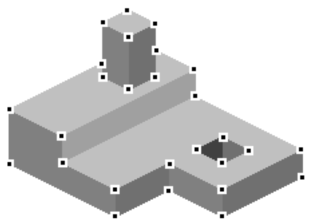
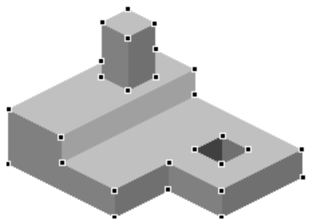
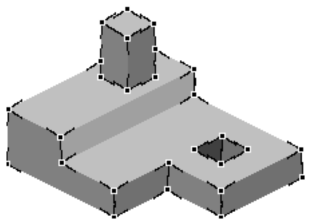
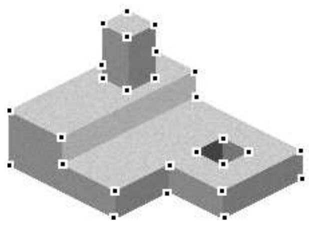
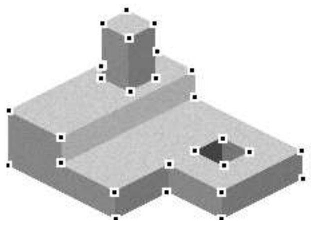
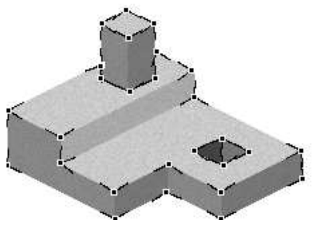
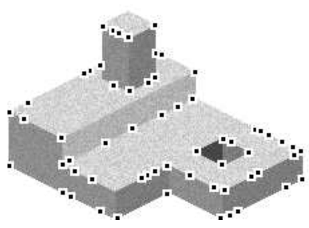
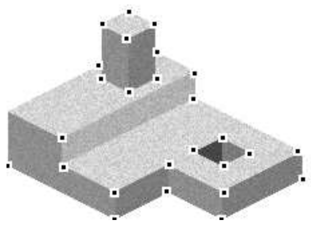
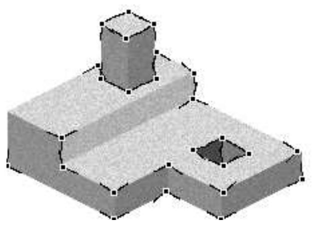
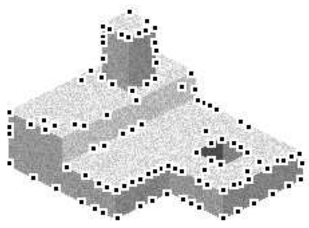
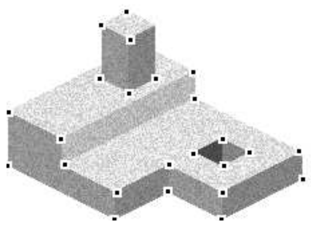
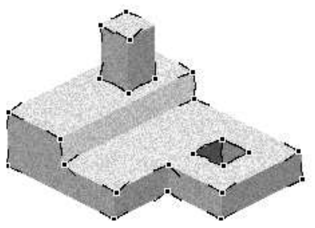
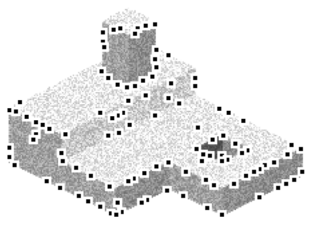
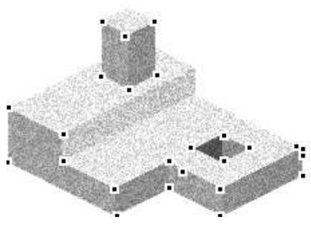
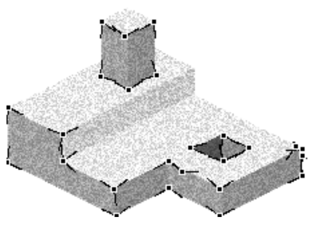
	SUSAN	CODE	
SNR	corners	corners	incident edges
0.00			
	Note: SNR= 0.00 implies ground-truth results.		
0.10			
0.20			
0.30			
0.50			

Figure 4.9: Outputs for “test-1” image for different values of SNR.

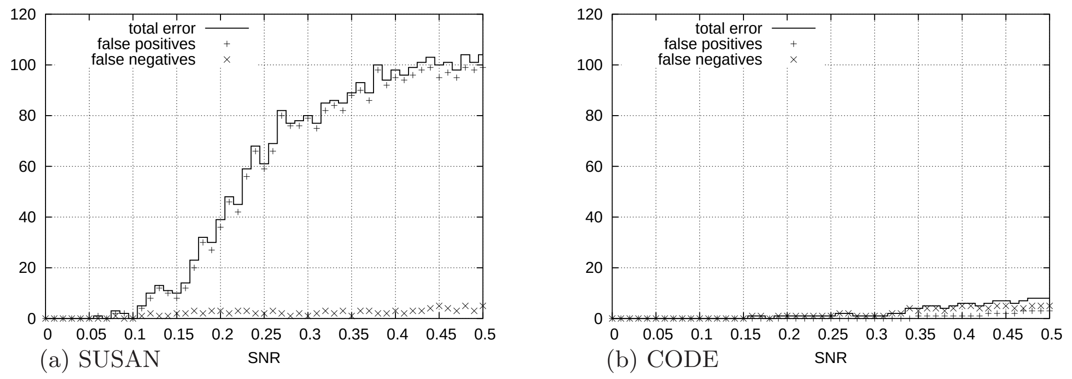


Figure 4.10: Plots showing number of false corners versus noise level (SNR) for "test-1" image shown in Fig. 4.9.

We have also shown here results on the natural image of "lab" for varying brightness thresholds, in order to establish the soundness and stability of the proposed corner detection algorithm. Output images for a few brightness thresholds are displayed in Fig. 4.12. It is evident from the output images that CODE is quite less sensitive to the brightness threshold compared to SUSAN.

Figs. 4.13 and 4.14 demonstrate the results obtained for two more gray-scale images, where the corners (and directions of the incident edges) are better detected by CODE than by SUSAN. In Fig. 4.15, the outputs for a binary logo image are given for both SUSAN and CODE, where all the corners along with the incident edge directions are properly and precisely detected by CODE, whereas, SUSAN fails to produce the desired result, particularly for the case when there is some jaggedness present in the object contour.

It may be noted that, apart from detection of corners, our algorithm also finds the directions of incident edges for each corner, and therefore, takes some additional time during execution. The details of experimental results for our algorithm are shown in Table 4.4.

4.4 Conclusion

Corners of an object serve as a very powerful geometric feature for different computer vision applications, as mentioned in Sec. 8.1. Coupled with (directions of) their incident edges, they provide a powerful abstraction of the underlying object. The nature of association of the corners and their incident edges with the corresponding object(s) has a striking distinctiveness, which, if exploited properly, would act as an effective higher-level

Image name	Image size	# corners A	# corners B	Time (secs.) C	Time (secs.) D	γ_∞
test1	262 × 196	185	27	0.32	0.13	10
test2	256 × 256	464	61	0.78	0.19	10
lab	512 × 512	956	314	2.13	1.07	30
blocks	256 × 256	162	56	0.66	0.39	30
house	256 × 256	105	60	0.38	0.25	30
logo	256 × 256	90	22	0.54	0.21	30

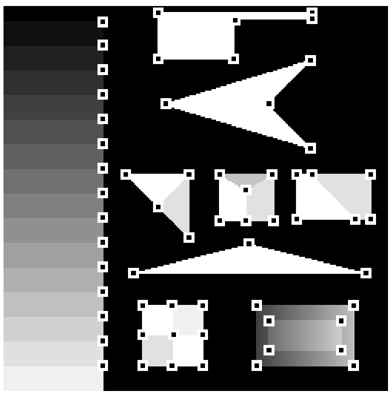
A = No. of corners before merging.
B = No. of corners after merging.
C = Time for detection of corners before merging.
D = Time for merging corners.

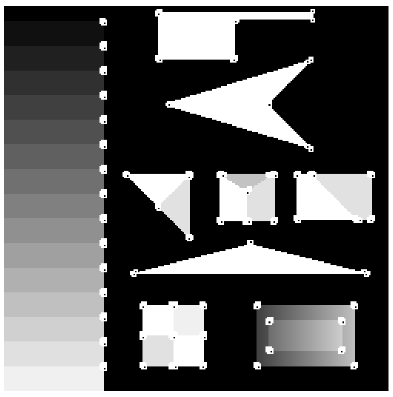
Table 4.4: Results for sample images.

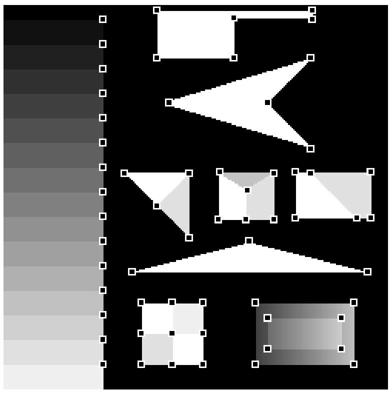
description of the object(s), and thus can be employed to solve many problems of computer vision, and image retrieval.

The proposed algorithm produces consistent results for both real images and synthetic images (binary as well as gray-scale), and is characterized by its adaptability with the gray-scale topology of an image. Though the computational complexity rises with the increase in radius of the annular list, but in terms of the accuracy and localization of detected corners and associated edge directions, the algorithm offers a high level of reliability and efficiency. Furthermore, the algorithm can be extended to devise a procedure of finding the most genuine corners in any image by augmenting the window to a suitable higher radius, having predetermined trade-off with computational complexities.

As an application, we can use the set of corners (along with the associated information such as directions of the incident edges at each corner) of an object to find its similarity with another object. In such an application, we can use the APSPM algorithm in the digital space [Chapter 2]. The problem of object matching/object tracking in a digital image/video, based on the APSPM algorithm using the set of corners and associated information, may be studied further in the future.


(a) SUSAN.


(b) CODE: corners before merging.


(c) CODE: corners after merging.

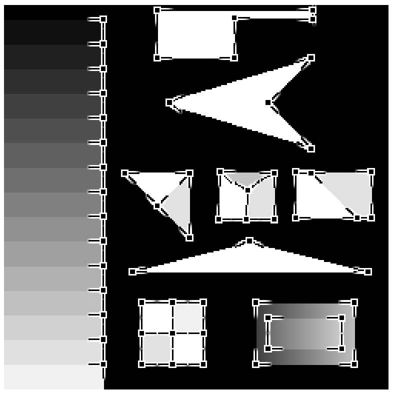

(d) CODE: incident edges.

Figure 4.11: Outputs for “test-2” image.

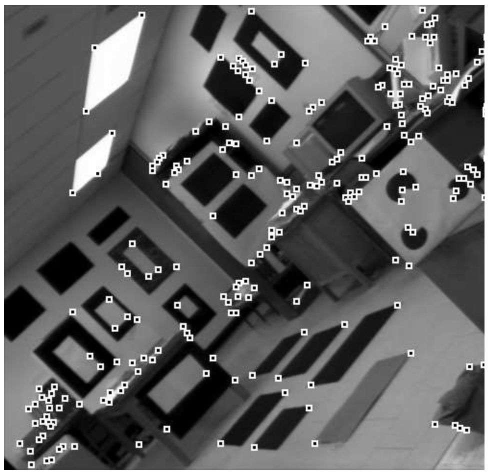
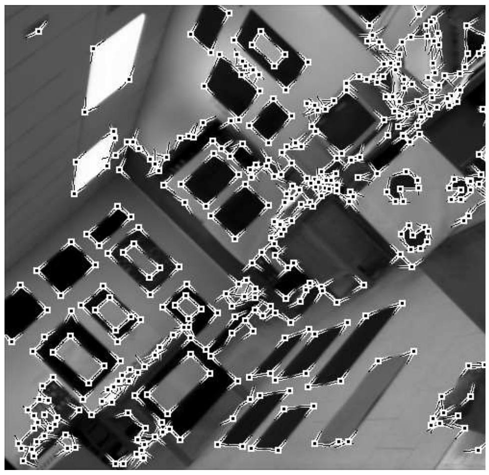
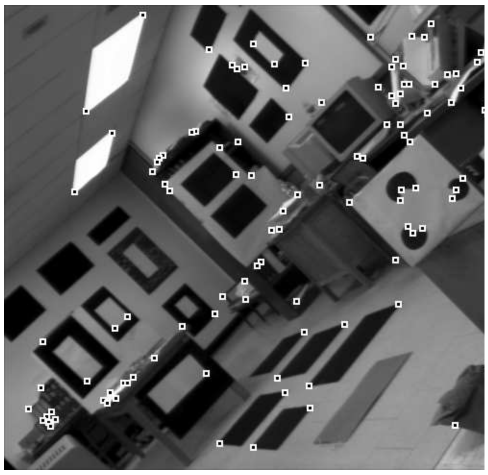
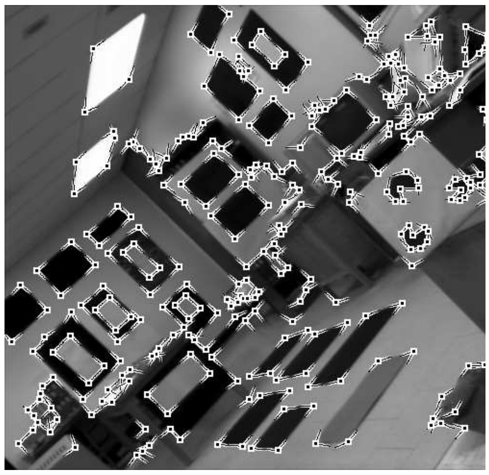
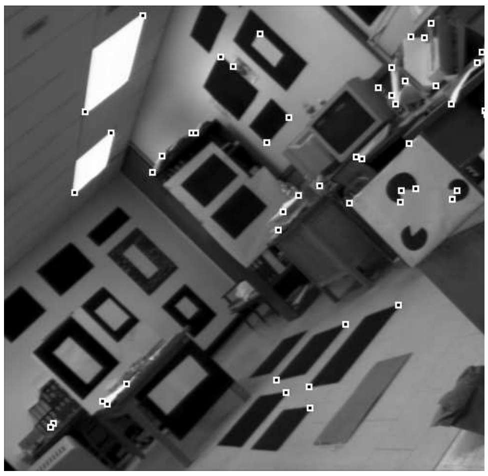
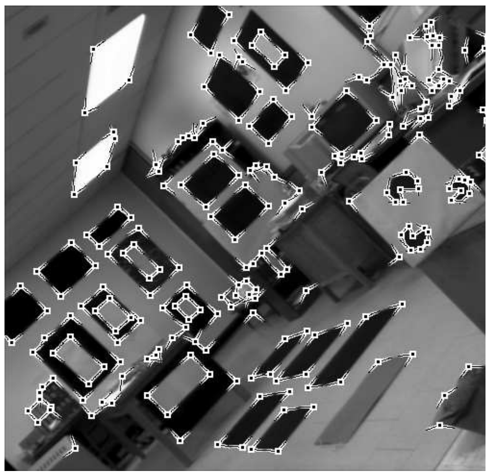
γ_∞	SUSAN	CODE (corners and incident edges)
20		
30		
40		

Figure 4.12: Outputs for "lab" image for different values of γ_∞ (brightness threshold).

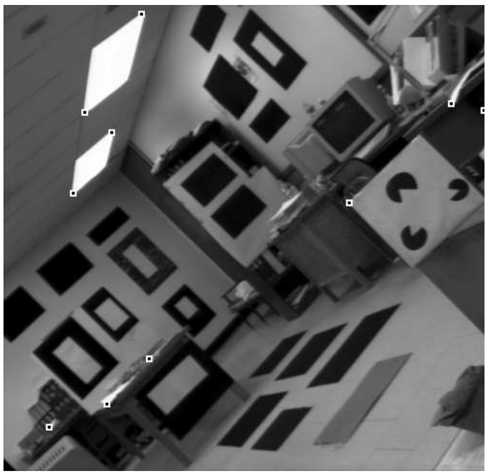
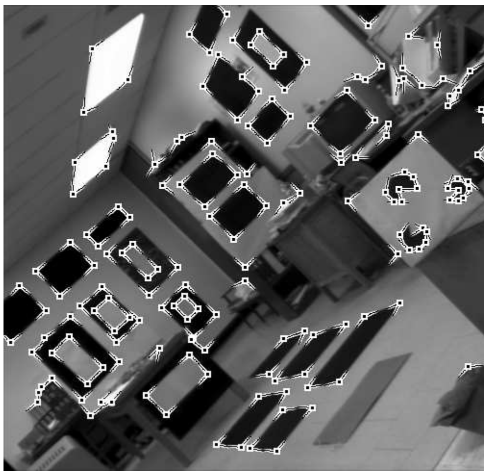
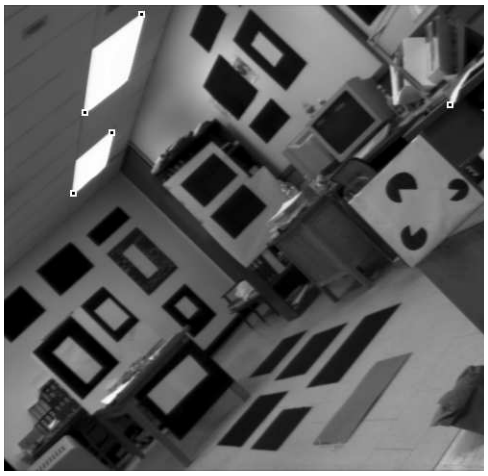
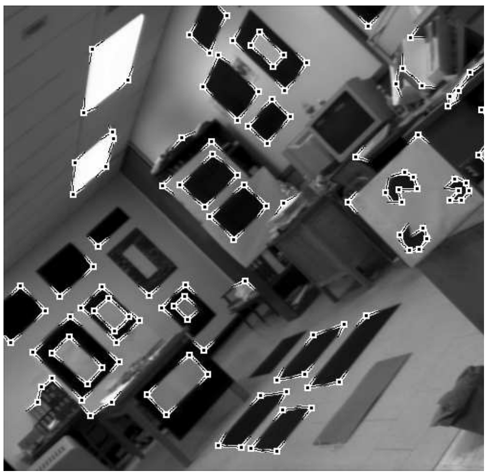
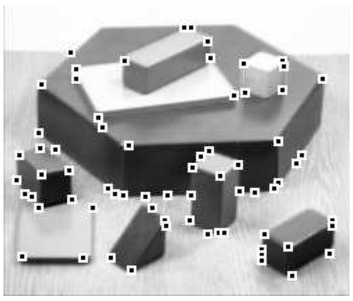
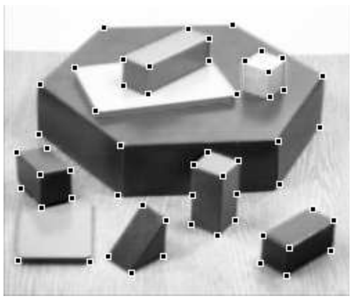
γ_∞	SUSAN	CODE (corners and incident edges)
60		
80		

Figure 4.12 (continued).


(a) SUSAN.


(b) CODE: corners.

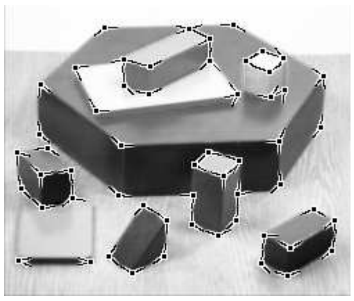
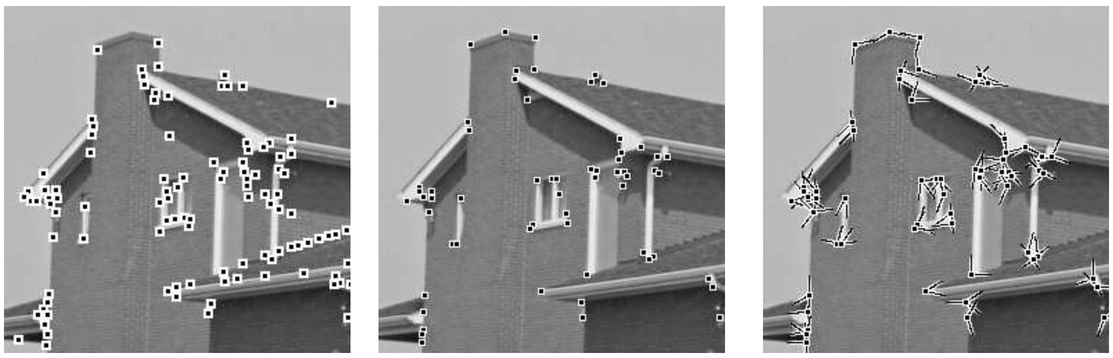
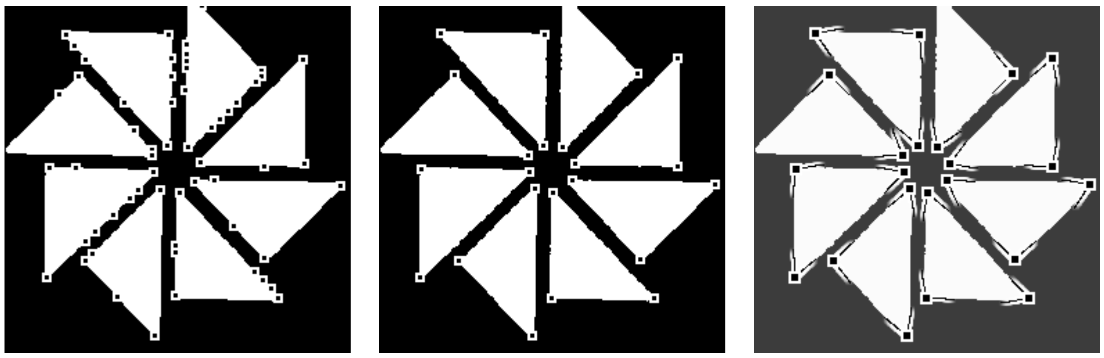

(c) CODE: incident edges.

Figure 4.13: Outputs for blocks image.



(a) SUSAN. (b) CODE: corners. (c) CODE: incident edges.

Figure 4.14: Outputs for house image.



(a) SUSAN. (b) CODE: corners. (c) CODE: incident edges.

Figure 4.15: Outputs for a sample logo image.

Chapter 5

Straight Lines in the Digital Plane and Polygonal Approximation of Digital Curves

The mathematician's patterns, like the painter's or the poet's must be beautiful; the ideas, like the colours or the words must fit together in a harmonious way. Beauty is the first test: there is no permanent place in this world for ugly mathematics.

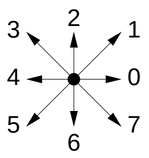
G. H. Hardy
A Mathematician's Apology (London, 1941)

5.1 Introduction

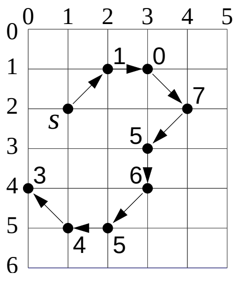
Efficient representation of lines and curves in the digital plane has been an active subject of research for nearly half a century [Klette and Rosenfeld (2004a,b), Rosenfeld and Klette (2001)]. In particular, digital straight line segments (DSS)[1] have drawn special attention for their challenging nature from the viewpoint of theoretical formulation, and also for their potential applications to image analysis and computer graphics. In a digital image containing one or more objects with fairly straight edges, the set of (exact or approximate) DSS captures a strong geometric property that can be used for shape abstraction of the underlying objects, as well as for finding the resemblance among several digital objects.

The necessary and sufficient conditions for a discrete/digital curve (DC) to be a DSS have been stated in the literature in various forms [Freeman (1961b), Klette and Rosenfeld (2004a), Rosenfeld (1974), Rosenfeld and Klette (2001)]. It has been shown by Rosenfeld in 1974 that a digital curve is the digitization of a straight line segment, if and only if it

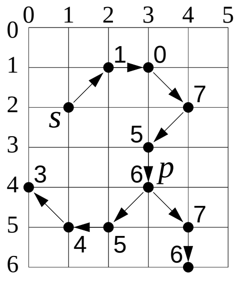
[1]The acronyms "DC", "DSL", "DSS", and "ADSS" have been used in this chapter in both singular and plural senses, depending on the context.



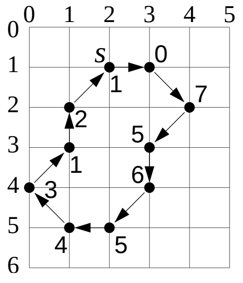
(a) chain codes in 8-N.



(b) since $(1, 2)$ is the start point, the chain code is $(1, 2)$`10756543`.



(c) since $(3, 4)$ is a branching point, the complete chain code is $(1, 2)$`10756543`$(3, 4)$`75`.



(d) with $(2, 1)$ as the start point, the chain code is $(2, 1)$`0756543121`.

Figure 5.1: Chain codes and their enumeration for defining digital curves.

has the *chord property.* A digital curve $\mathcal{C}$ has the chord property if, for every (p, q) in $\mathcal{C}$, the chord $\overline{pq}$ (the line segment, drawn in real plane, joining p and q) "lies near" $\mathcal{C}$, which, in turn, means that for any point (x, y) of $\overline{pq}$, there exists some point (i, j) of $\mathcal{C}$ such that $\max(|i - x|, |j - y|) < 1$. Few other definitions related to this work are given below.[1]

Chain Code: If $p\,(i, j)$ is a grid point, then the grid point (i', j') is a neighbor of p, provided $\max(|i - i'|, |j - j'|) = 1$. The chain code [Freeman (1961a,b)] of p w.r.t. its neighbor grid point in $\mathcal{C}$ can have a value in $\{0, 1, 2, \ldots, 7\}$, as shown in Fig. 5.1(a).

Digital Curve (DC): A digital curve $\mathcal{C}$ is an ordered sequence of grid points (representable by chain codes) such that each point (excepting the first one) in $\mathcal{C}$ is a neighbor of its predecessor in the sequence (see Fig. 5.1(b)–(d)).

Irreducible Digital Curve: An open-end digital curve $\mathcal{C}$ is said to be irreducible if and only if removal of any grid point (excepting the end points) in $\mathcal{C}$ makes $\mathcal{C}$ disconnected. If $\mathcal{C}$ is a closed digital curve (Fig. 5.1(d)), then $\mathcal{C}$ is irreducible if and only if removal of any of its grid points makes it open-end. All the DC shown in Fig. 5.1 are irreducible. A DSS is essentially an irreducible DC.

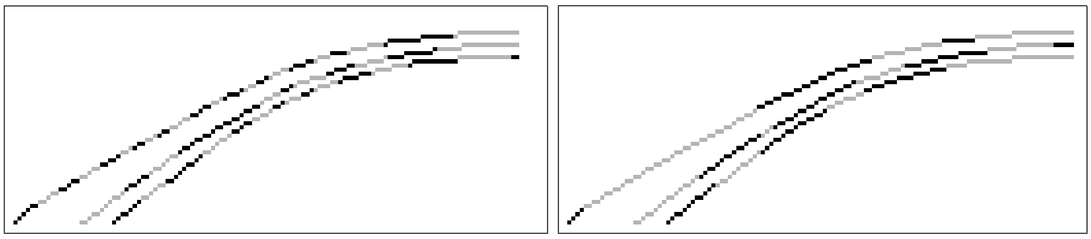
[1]The definitions and discussions in this chapter are with respect to 8-neighbor connectivity [Klette and Rosenfeld (2004a)] of the object, and are valid for 4-neighbor connectivity as well with certain modifications.

An example of an open-end DC is shown in Fig. 5.1(b). The traversal of an open-end DC using Depth First Search (DFS) [Cormen *et al.* (2000)], starts from one of its two end points, say s. Thus, the chain code of $\mathcal{C}$ is given by $(1,2)\texttt{10756543}$, considering $s = (1,2)$. If $\mathcal{C}$ has some branching (Fig. 5.1(c)), then the complete chain code of $\mathcal{C}$ can be written as $(1,2)\texttt{10756543}(3,4)\texttt{76}$. If $\mathcal{C}$ is a closed curve, then the DFS may start from any suitable grid point of $\mathcal{C}$ (Fig. 5.1(d)).

Several intriguing problems and properties related to DSS and DSL (digital straight line/ray) have been studied by various authors [Bresenham (1965), Povazan and Uher (1998), Rosenfeld and Klette (2001)]. Many attributes of DSS can be interpreted in terms of continued fractions [Koplowitz *et al.* (1990), Mignosi (1991), Voss (1991)]. The most fundamental problem, which is highly relevant to pattern recognition in general, and to curve approximation in particular, is to ascertain whether or not a given DC is a DSS. Many solutions to this problem have been reported in the literature [Creutzburg *et al.* (1982), Davis *et al.* (1976), Debled-Rennesson and Reveilles (1995), Kovalevsky (1990), Mieghem *et al.* (1995), Smeulders and Dorst (1991)]. Debled-Rennesson *et al.* (2003) has proposed the concept of fuzzy segments for a flexible segmentation of discrete curves, using an arithmetic approach of discrete straight lines.

The proposed work introduces a new concept of approximate digital straight line segments (ADSS), by preserving some of the most fundamental properties of a DSS, while relaxing or dropping a few others (see Fig. 5.4). A procedure is then described for extracting the set of ADSS required to cover a given DC assuming 8-neighborhood connectivity. The number of ADSS extracted from a set of DC in a real world scenario is likely to be appreciably fewer than that of DSS cover, since many visually straight segments may fail to satisfy all the stringent properties of an exact DSS, and thus are recognized as multiple DSS by the extraction algorithms. Some examples of DSS and ADSS present in digital curves have been shown in Fig. 5.2. It may be observed that the set of DSS in Fig. 5.2 contains forty-eight fragments each of which is "exactly straight", whereas, that of ADSS contains only twenty, which look "visually straight".

The concept of ADSS can also be used to construct polygonal approximation of a DC efficiently. Since the set of ADSS provides an elegant and compact representation of digital curves, it is very effective in producing approximate polygons (or, polychains) using a single parameter. The whole process consists of two stages — first, extraction of ADSS, and then the second stage of polygonal approximation. A glimpse of the algorithm has been shown in Fig. 5.3 for a preliminary introduction on a set of digital curves taken from the real-world image representing the contour of a "duck". Both the stages in the



(a) DSS (48 nos.) (b) ADSS (20 nos.)

Figure 5.2: Set of DSS and that of ADSS extracted from a small (cropped) set of digital curves, the segments being alternately colored in black and gray colors.

algorithm can be easily implemented, and have been found to work correctly and efficiently on different sets of digital curves of arbitrary shapes and complexities. The major features of the algorithm are as follows:

(i) The detection of ADSS in stage 1 is based on simple chain code properties; thus, only primitive integer operations, namely comparison, increment, shift, and addition (subtraction) are required.

(ii) The ADSS extraction algorithm does not use any recursion, and thus saves execution time.

(iii) To compute the polygonal approximation in stage 2, only the two end points of each ADSS are required as input data and a few integer multiplications as operations; thus the algorithm runs very fast.

(iv) The actual approximation of a DC never oversteps the worst-case approximation for a given value of a control parameter. That is, the maximum deviation (of an edge) of the resulting polygon from the original curve never exceeds the prescribed value of the approximation (control) parameter.

Several other methods [Asano and Kawamura (2000), Asano *et al.* (2003a), Chen and Chung (2001b), Climer and Bhatia (2003), Guru *et al.* (2004), Xie and Ji (2001)] have been proposed recently for (approximate) line detection. Most of the conventional parametric approaches are based on certain distance criteria, usage of masks, eigenvalue analysis, Hough transform, etc. In contrast, the proposed method relies on utilizing some of the basic properties of DSS for extraction of ADSS.

Many algorithms for approximating a given digital curve or contour are well known [Aken and Novak (1985), Attneave (1954), Imai and Iri (1986)]. Several variants of efficient but suboptimal algorithms had been proposed later [Bhowmick *et al.* (2005b, 2006),

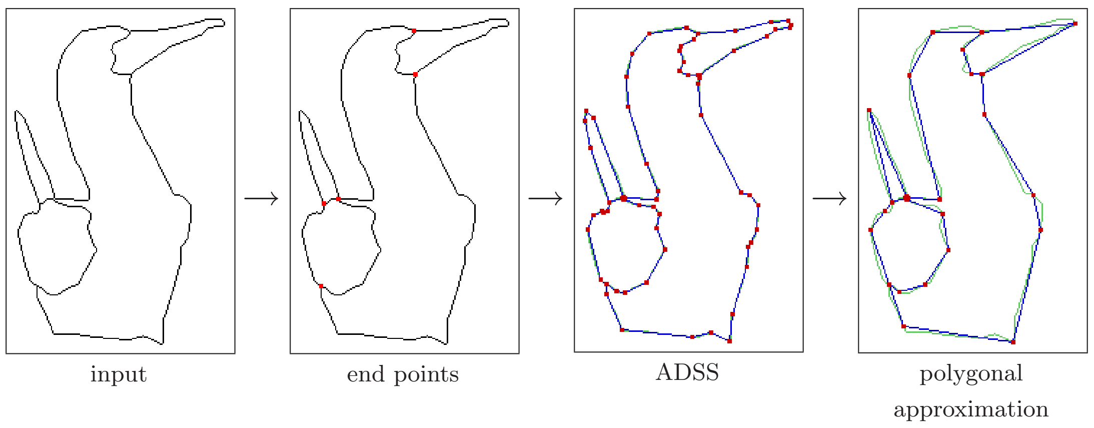


Figure 5.3: A brief demonstration of the proposed algorithm on the "duck" image.

Biswas *et al.* (2005c), Perez and Vidal (1994), Schröder and Laurent (1999), Schuster and Katsaggelos (1998)]. The class of polygonal approximation algorithms, in general, 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.

Most of the existing polygonal approximation algorithms, excepting a few, have superlinear time complexities; to cite a few, the complexities are $\mathcal{O}(N)$ in [Wall and Danielsson (1984)], $\mathcal{O}(MN^2)$ in [Perez and Vidal (1994)], $\mathcal{O}(N^2)$ in [Schröder and Laurent (1999), Schuster and Katsaggelos (1998)], $\mathcal{O}(N^3)$ in [Rocha and Bernardino (1998)], where M denotes the number of segments, and N the total number of points representing the input set of DC. A comparative study of these algorithms can be found in the paper by Yin (1998). Further, in order to analyze curvature, most of them require intensive floating-point operations [Anderson and Bezdek (1984), Fischler and Wolf (1994), Freeman and Davis (1977), Teh and Chin (1989), Wuescher and Boyer (1991)]. 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. The method developed by us uses integer operations only, and yields a suboptimal polygonal approximation with linear time complexity. A comparison of the proposed technique with some of the important approaches has been shown in Table 5.1.

A brief outline of the chapter is as follows. In Sec. 5.2, we start with a brief overview of some fundamental digital-geometric properties of a DSS, followed by the motivation and underlying principle for extraction of the approximate segments (ADSS) in Sec. 5.2.1.

Table 5.1: A comparative study of some existing algorithms with the proposed one.

	Algorithm and its features	Non-Eucli-dean	Flexi-bility[a]	Proce-dural comple-xity	Difficulty in imple-mentation	Error control	Non-recur-sive
1.	**curvature maxima** [Teh and Chin (1989)] – region of support – measure of significance[b] – non-maxima suppression	no	no	high[c]	medium	no	no
2.	**ant colony search** [Yin (2003)] – graph representation – node transition rule – pheromone updating rule	no	no	very high[d]	high	yes	no
3.	**area deviation** per unit length of approximating segment [Wall and Danielsson (1984)]	yes[e]	no	low	low	yes	yes
4.	**perceptual organization** [Hu and Yan (1997)] – link classification[f] – linking-merging – smoothing – application of knowledge or rule	yes	no	high[g]	high	yes	no
5.	**using ADSS** (proposed).	yes	yes	very low	low	yes	yes

[a]w.r.t. (self-)intersecting/branching curves when the input set of grid points constituting a curve is not ordered

[b]e.g., cosine curvature, k curvature, 1 curvature, etc.

[c]due to multiple iterations for non-maxima suppression (in 4 passes) and curvature finding

[d]due to complex calculations, e.g., exponentiation in selection problem of a node

[e]by replacing $\sqrt{x^2+y^2}$ with $(|x|+|y|)$ or $\max\{|x|,|y|\}$

[f]parallel links (class 1 & 2), intersection links, and single links.

[g]due to three stages, each using multiplications for the entire set of points on the curve

Sec. 5.2.2 presents the algorithm on extraction of ADSS along with its brief analysis, and in Sec. 5.2.3, the condition of erroneousness of a point in an ADSS (due to deviation from the DSS properties) has been derived. The proposed method on (suboptimal) polygonal approximation has been explained in Sec. 5.3, with the two possible approximation criteria briefed in Secs. 5.3.1.1 and 5.3.1.2, the algorithm stated in Sec. 5.3.2 along with the proof of its linear time complexity, and the subsequent quality of approximation formulated in Sec. 5.3.3. Sec. 5.4 exhibits some test results (including DSS extraction, ADSS extraction, polygonal approximation, and quality measures of approximation), endorsing the supremacy of ADSS in polygonal approximation. Finally, Sec. 5.5 summarizes the work.

5.2 Exact and Approximate Straight Line Segments

In this work, we use some regularity properties of DSS that can be successfully derived from the chord property. Before justifying the rationale of our algorithm, the DSS properties (defined w.r.t. chain codes [Freeman (1961a)]) [Freeman (1961b), Rosenfeld (1974)] are listed below.

(F1) at most two types of elements can be present, and these can differ only by unity, modulo eight;
(F2) one of the two element values always occurs singly;
(F3) successive occurrences of the element occurring singly are as uniformly spaced as possible.

The properties (F1–F3) were based on heuristic insights [Freeman (1961b)]. Further, the property (F3) is not precise enough for a formal proof, as stated by Pavlidis (1977). A formal characterization of DSS was provided later [Rosenfeld (1974)], stated as follows.

(R1) The runs have at most two directions, differing by 45^0, 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 instances have been given in Sec. 5.2.1 (see Fig. 5.4) to clarify the significance of (R1–R4) in characterizing a DSS.

5.2.1 Extraction of ADSS

In the proposed algorithm, Extract-ADSS, designed for extraction of ADSS from a DC, we have used (R1) along with certain modifications in (R2). However, we have dropped (R3) and (R4), since they impose very tight restrictions on a DC to be recognized as a DSS. Such a policy, with adoption of (R1), modification of (R2), and omission of (R3) and (R4), has been done in order to successfully extract the ADSS from a DC, and some of the major advantages of this scheme are as follows:

(i) avoiding tight DSS constraints, especially for the curves representing the gross pattern of a real-life image with certain digital aberrations/imperfections;
(ii) enabling extraction of ADSS from a DC, thereby straightening of a part of the DC when the concerned part is not exactly "digitally straight";
(iii) reducing the number of extracted segments, thereby decreasing storage requirement and run-time in subsequent applications;
(iv) reducing the CPU time of ADSS extraction;
(v) usage of integer operations only.[1]

Since the chain code of a DC is a one-dimensional list, $\mathcal{C}$, the ADSS may be characterized by the following sets of parameters:

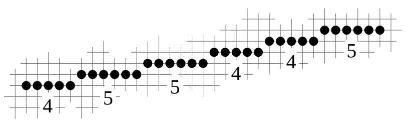
(i) *orientations parameters* given by $\mathtt{n}$ (non-singular element), $\mathtt{s}$ (singular element), l (length of leftmost run of $\mathtt{n}$), and r (length of rightmost run of $\mathtt{n}$), which play decisive roles on the orientation (and the digital composition, thereof) of the concerned ADSS. For example, in Fig. 5.4, the curve $\mathcal{C}_1$ has $\mathtt{n} = \mathtt{0}$, $\mathtt{s} = \mathtt{1}$, and chain code $\mathtt{0^4 1 0^5 1 0^5 1 0^4 1 0^4 1 0^5}$ having $l = 4$ and $r = 5$.
(ii) *run length interval parameters* given by p and q, where $[p, q]$ is the range of possible lengths (excepting l and r) of $\mathtt{n}$ in $\mathcal{C}$ that determines the level of approximation of the ADSS, subject to the following two conditions:

$$\text{(c1)}\quad q - p \leq d = \lfloor (p+1)/2 \rfloor. \tag{5.1}$$

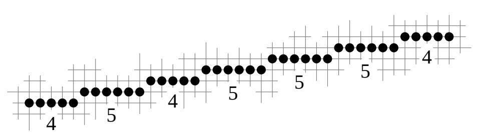
$$\text{(c2)}\quad (l-p), (r-p) \leq e = \lfloor (p+1)/2 \rfloor. \tag{5.2}$$

While implementing Extract-ADSS, we have strictly adhered to (R1), as it is directly related to the overall straightness of a DC. However, we have modified the stricture in (R2) by considering that the run lengths of $\mathtt{n}$ can vary by more than unity, depending on the

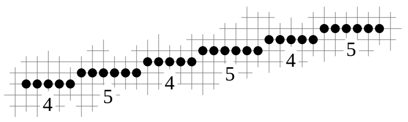
[1] No floating-point operations are required in Extract-ADSS. Only integer operations for addition, shift, and comparison are necessary. Even multiplications and divisions have been avoided, e.g., to compute $\lfloor (p+3)/4 \rfloor$, 3 is added with p, followed by two successive right shifts.



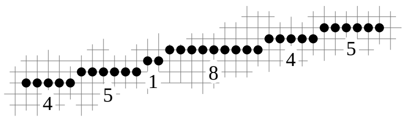
$\mathcal{C}_1$: Chain code $\mathtt{0^4 10^5 10^5 10^4 10^4 10^5}$ (from left to right) ($p = 4, q = 5, l = 4, r = 5$) does not satisfy (R3), since both the run lengths `4` and `5` have non-singular occurrences in the code of run lengths: `455445`. $\mathcal{C}_1$ is not a DSS but an ADSS.



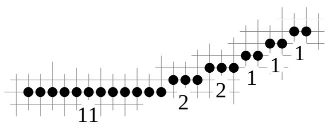
$\mathcal{C}_2$: Chain code $\mathtt{0^4 10^5 10^4 10^5 10^5 10^5 10^4}$ ($p = 4, q = 5, l = 4, r = 4$) does not satisfy (R4), since in the run length code `4545554`, the runs of `5` have lengths 1 and 3 that are not consecutive. $\mathcal{C}_2$ is not a DSS but an ADSS.



$\mathcal{C}_3$: Chain code $\mathtt{0^4 10^5 10^4 10^5 10^4 10^5}$ ($p = 4, q = 5, l = 4, r = 5$) satisfies (R1–R4) and (c1, c2). $\mathcal{C}_3$ is an ADSS as well as a DSS.



$\mathcal{C}_4$: Chain code $\mathtt{0^4 10^5 1010^8 10^4 10^5}$ ($p = 1, q = 8, l = 4, r = 5$) does not satisfy (R2) and conditions (c1, c2). $\mathcal{C}_4$ is neither a DSS nor an ADSS.



$\mathcal{C}_5$: Chain code $\mathtt{0^{11} 10^2 10^2 101010}$ ($p = 1, q = 2, l = 11, r = 1$) violates (R2) and so it is not a DSS. Further, although it satisfies (c1) (as $q - p\,(= 1) \leq d\,(= 1)$), but since $l - p\,(= 10) \not\leq e\,(= 1)$, it fails to satisfy (c2); hence, it is not even recognized as an ADSS.

Figure 5.4: Instances of digital curves showing the significance of properties and conditions related with DSS and ADSS recognition.

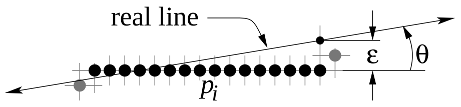


Figure 5.5: Maximum (isothetic) error ε corresponding to a run length p_i (of `0`'s) — with $p_i + 1$ number of grid points in the corresponding row — for an ADSS representing a real line segment with slope $m = \tan\theta$, $0 \leq m \leq 1$. Although the real line intersects the run, in some other instance it may not.

minimum run length of $\mathtt{n}$. The rationale of modifying (R2) to the condition (c1) lies in the fact that, in order to approximate the extracted line segments from the DC, an allowance of approximation (d) specified by (c1) may be permitted. Given a value of p, the amount d by which q is in excess of p indicates the deviation of the ADSS from the actual/real line, since ideally (for a DSS) q can exceed from p by at most unity; and the significance of d in characterizing an ADSS is as follows.

W.l.o.g., let **L** be an ADSS with slope in the interval $[0, 1]$. Let $p_0 = p, p_1 = p_0+1, p_2 = p_1 + 1, \ldots, p_d = q$ be the run lengths of `0`'s present in **L**, and let the frequency of the run length p_i in **L** be $n_i\,(\geq 0)$, for $i = 0, 1, \ldots, d$. Then the slope of the real line (joining the start point and the end point of **L**) is given by

$$m = \frac{\sum\limits_{i=0}^{d} n_i - 1}{\sum\limits_{i=0}^{d} n_i(p_i + 1)} = \frac{N - 1}{\sum\limits_{i=0}^{d} n_i(p_i + 1)}, \tag{5.3}$$

where, $N = \sum\limits_{i=0}^{d} n_i$. Hence, using the fact that $p \leq p_i \leq q$ for $i = 0, 1, \ldots, d$, we get

$$\begin{aligned} \frac{N-1}{\sum\limits_{i=0}^{d} n_i(q+1)} &\leq m \leq \frac{N-1}{\sum\limits_{i=0}^{d} n_i(p+1)}, \\ \text{or,} \qquad \frac{N-1}{N}\frac{1}{q+1} &\leq m \leq \frac{N-1}{N}\frac{1}{p+1}. \end{aligned} \tag{5.4}$$

Thus, when the real line intersects the concerned run (if the real line does not intersect the concerned run of **L**, then the error is greater), the isothetic error of a run length p_i in **L** (see Fig. 5.5) is given by

$$\varepsilon \leq m(p_i + 1) \leq \frac{N-1}{N}\frac{p_i+1}{p+1} \leq \frac{N-1}{N}\frac{q+1}{p+1} \leq \left(1 - \frac{1}{N}\right)\left(1 + \frac{d}{p+1}\right),$$

$$\text{or,} \qquad \varepsilon \leq 1 + \frac{d}{p+1}. \tag{5.5}$$

Hence, it is evident that the error incurred with an ADSS is controlled by d, and by Eqn. 5.5, lower the run length (p) of $\mathfrak{n}$, lower would be this allowance of approximation. Thus, we keep the provision for adaptively changing this allowance of approximation, so that elongation of an ADSS is made as much as possible till it does not lose its overall visual straightness.

Apart from d, the other parameter, namely e, is incorporated in (c2), which, along with (c1), ensures that the extracted ADSS is not badly approximated owing to some unexpected values of l and r. The DSS properties, (R1–R4) however, do not give any idea about the possible values of l and r (depending on $\mathfrak{n}$). Further, in the algorithm for DSS recognition [Creutzburg *et al.* (1982)], l and r are not taken into account for adjudging the DSS characteristics of a DC. However, we have imposed some bounds on the possible values of l and r, in order to ensure a reasonable amount of straightness at either end of an extracted ADSS. The values of d and e are heuristically chosen so that they become computable with integer operations only. Some other values, like $d = \lfloor (p+3)/4 \rfloor$ and $e = \lfloor (p+1)/2 \rfloor$, or so, may also be chosen provided the computation is realizable in integer domain and does not produce any undesirable ADSS. For example, in Fig. 5.4, the curve $\mathcal{C}_5$ has $p = 1, q = 2, l = 11, r = 1$. In our case (Eqns. 5.1 and 5.2), therefore, we get $d = 1$ and $e = 1$ resulting a violation of (c2) by l; thus $\mathcal{C}_5$ will not be accepted as an ADSS.

To justify the rationale of (c1) and (c2), we consider a few digital curves, $\mathcal{C}_1$–$\mathcal{C}_5$, as shown in Fig. 5.4. It is interesting to observe that, although each of $\mathcal{C}_1$ and $\mathcal{C}_2$ has the appearance of a digital line segment, they fail to hold all the four properties of DSS simultaneously, as shown in their respective figures. The curve $\mathcal{C}_1$ violates (R3), and the curve $\mathcal{C}_2$ violates (R4). However, they satisfy (R1), (c1), and (c2) and therefore, each of them is declared as an ADSS. Similarly, the curve $\mathcal{C}_3$ satisfies (R1–R4) and (c1, c2); it is both an ADSS and a DSS. However, none of the curves $\mathcal{C}_4$ and $\mathcal{C}_5$ can be announced as a DSS or an ADSS because of the violation of (R2), (c1), and (c2) [see Fig. 5.4].

5.2.2 Algorithm Extract-ADSS

Figure 5.6 describes the algorithm Extract-ADSS for extracting ADSS from the chain code of each DC, say $\mathcal{C}_k$, stored in the list $\mathcal{C}$. This requires n_k repetitions from Step 2 through Step 23, where n_k is the number of ADSS in $\mathcal{C}_k$. Let the ith repetition on $\mathcal{C}_k$

produce the ADSS $\mathbf{L}_i^{(k)}$. Recognition of $\mathbf{L}_i^{(k)}$ is prompted by finding its corresponding parameters $(\mathtt{n}, \mathtt{s}, l)$ using the FIND-PARAMS procedure in Step 2 of EXTRACT-ADSS. This is followed by checking/validation of

(i) property (R1): Step 4 and Step 10;
(ii) condition (c1): **while** loop check at Step 9;
(iii) condition (c2): on the leftmost run length l in Step 8 and Step 11, and on the rightmost run length r in Step 14.

Proof of correctness: For each ADSS, $\mathbf{L}_i^{(k)}$, we show that property (R1) and conditions (c1) and (c2) are simultaneously satisfied. We also show that $\mathbf{L}_i^{(k)}$ is maximal in length in $\mathcal{C}_k$ in the sense that inclusion of the character ($\mathtt{n}$ or $\mathtt{s}$ or any other in $\{\mathtt{0}, \mathtt{1}, \ldots, \mathtt{7}\}$) (or a substring of characters) that immediately precedes or follows the part of DC corresponding to $\mathbf{L}_i^{(k)}$ in $\mathcal{C}_k$ does not satisfy the ADSS property/conditions.

While checking (R1) in Step 4 or Step 10, if an expected $\mathtt{n}$ or $\mathtt{s}$ is not found at the desired place in $\mathcal{C}_k$, then the current ADSS, $\mathbf{L}_i^{(k)}$ ends with the previously checked valid characters. This is explicit in Step 4 and implicit in Step 10. Thus $\mathbf{L}_i^{(k)}$ satisfies (R1), and is maximal from its starting point and the finishing end, since either it is the first ADSS in $\mathcal{C}_k$ or the previous ADSS $\mathbf{L}_{i-1}^{(k)}$, was maximal.

Now, for each new run (of $\mathtt{n}$), (c1) is verified in Step 9 — excepting the leftmost run, l, which is not required since p (and q) does not exist for a single run — after appropriately updating p and q in Step 17 and Step 19 respectively, whenever necessary. In Step 9, if it is found that q is unacceptably large (i.e., $q \not\leq p + d$), then the **while** loop (steps 10–19) is not executed, and the current ADSS, $\mathbf{L}_i^{(k)}$, ends with the truncated part of that run (truncated maximally, i.e., up to length $p + e$, in Step 15 of the previous iteration) as its rightmost run, r.

For checking (c2), however, we have to be more careful. For the second run (i.e., the run immediately following l) of the current ADSS, (c2) is checked (with respect to l) in Step 8. It may be noted that, if $l - p > e$, then (c2) is not satisfied, and so the first two runs (l and its successor) trivially constitute an ADSS by Step 7; because for two runs, we get only l and r (and no p or q), and no relation is imposed between l and r to define an ADSS.

For the third and the subsequent run(s), if any, the corresponding run length is stored in k (Step 10). If some (small enough) k violates (c2), then that k is treated as r (steps 11–13), and the current ADSS ends with that run as the rightmost run (of run length k), whereby the maximality criterion of the ADSS is fulfilled. Otherwise, if k does not exceed

```
Algorithm EXTRACT-ADSS ($\mathcal{C}$)

Steps:
 1. $\mathcal{A} \leftarrow \{1\}, u \leftarrow 1$
 2. FIND-PARAMS ($\mathcal{C}, u$)
 3. $c \leftarrow l$
 4. if $s - n \pmod 8 \neq 1$ then
        goto Step 20
 5. $p \leftarrow q \leftarrow$ length of next run of $n$
 6. $d \leftarrow e \leftarrow \lfloor (p+1)/2 \rfloor$
 7. $c \leftarrow c + 1 + p$
 8. if $l - p > e$ then goto Step 20
 9. while $q - p \leq d$
10.     $k \leftarrow$ length of next run of $n$
11.     if $l - k > e$ then
12.         $c \leftarrow c + 1 + k$
13.         break
14.     if $k - p \leq e$ then $c \leftarrow c + 1 + k$
15.     else $c \leftarrow c + 1 + p + e$
16.     if $k < p$ then
17.         $p \leftarrow k$
18.         $d \leftarrow e \leftarrow \lfloor (p+1)/2 \rfloor$
19.     if $k > q$ then $q \leftarrow k$
20. $u \leftarrow u + c$
21. $\mathcal{A} \leftarrow \mathcal{A} \cup \{u\}$
22. $u \leftarrow u + 1$  ▷ next start point
23. repeat from Step 2 until $\mathcal{C}$ is finished
```

```
Procedure FIND-PARAMS ($\mathcal{C}, u$)

Steps:
 1. $i \leftarrow u$
 2. if $\mathcal{C}[i] = \mathcal{C}[i+1]$ then
 3.     $n \leftarrow \mathcal{C}[i]$
 4.     $s \leftarrow$ element $\neq n$ following $\mathcal{C}[i+1]$
 5.     $l \leftarrow$ leftmost run length of $n$
 6.     return
 7. else
 8.     $n \leftarrow \mathcal{C}[i]$, $s \leftarrow \mathcal{C}[i+1]$, $l \leftarrow 1$
 9.     $i \leftarrow i + 1$
10.     while $\mathcal{C}[i+1] \in \{n, s\}$
        ▷ $\mathcal{C}$ ends if $\mathcal{C}[i+1] = -1$
11.         if $\mathcal{C}[i] = \mathcal{C}[i+1]$ then
12.             if $\mathcal{C}[i] = s$ then
13.                 swap $n$ and $s$
14.                 $l \leftarrow 0$
15.             return
16.         else
17.             $i \leftarrow i + 1$
18.     return
```

Figure 5.6: Algorithm EXTRACT-ADSS and the procedure FIND-PARAMS to find out the ordered list $\mathcal{A}$ of end points of ADSS in the input curve $\mathcal{C}$ that contains the chain code for each irreducible digital curve segment.

the maximum possible length of the rightmost run (checked in Step 14), then we consider k as a valid run of the current ADSS (Step 14), else we truncate it to the maximum permissible length $(p+e)$ as the rightmost run (Step 15). Note that, if $k > p+e$, then $k > q$ (for $p+e \geq p+d \geq q$), and Step 19 updates q to k, whence (c1) will be false in Step 9 in the next iteration, and so, the ADSS will end here with the (maximally) truncated part $(p+e)$ as its rightmost run.

Time Complexity: Determination of the parameters $(\mathtt{n}, \mathtt{s}, l)$ in Find-Params consists of two cases — the first one (steps 2–6) being easier than the second (steps 7–18). In either of these two cases, the procedure searches linearly in $\mathcal{C}$ for two distinct (but not necessarily consecutive) chain code values and determines the parameters accordingly. As evident from the loop in either case, the three parameters are obtained using only a few integer comparisons. The number of comparisons is $l+1$ for the first case, and that for the second case is the number of characters in $\mathcal{C}$ until two consecutive non-singular characters are found.

The parameters $\mathtt{n}, \mathtt{s}, l$ obtained in Find-Params are successively passed through a number of check points, as mentioned earlier, which take constant time as evident in steps 3–8 of Extract-ADSS. In Step 5 of Extract-ADSS, the first run length of $\mathtt{n}$ is measured immediately after the leftmost run length of $\mathtt{n}$, if any, and it starts from the first non-singular character out of the two consecutive characters detected in Find-Params. In Step 10 of Extract-ADSS, we have another simple (and silent) loop that determines in linear time each valid run of $\mathtt{n}$ in $\mathcal{C}$, the validity criteria being verified and updated in steps 9–19, each of these steps taking constant time. Hence, for the ADSS, $\mathbf{L}_i^{(k)}$, the algorithm Extract-ADSS, together with the procedure Find-Params, takes linear time; wherefore the time complexity for extraction of all ADSS in $\mathcal{C}$ is strictly linear on the number of points in $\mathcal{C}$.

5.2.3 Error Points

An ADSS extracted from an input digital curve may not be a perfect DSS, and erroneous points may occur. An erroneous point or error point is that whose isothetic distance (i.e., minimum of the vertical distance and the horizontal distance) from the real straight line corresponding to the concerned ADSS is greater than $\frac{1}{2}$.

Let S_i and E_i be the start point and the end point of the ith ADSS, and let P_{ij} be the jth digital point on the ith ADSS, as shown in Fig. 5.7. Let $\overline{\mathsf{S}_i\mathsf{E}_i}$ denote the real line segment joining S_i and E_i. Let the point on $\overline{\mathsf{S}_i\mathsf{E}_i}$ vertically above (or below) P_{ij} be V_{ij}, and let the

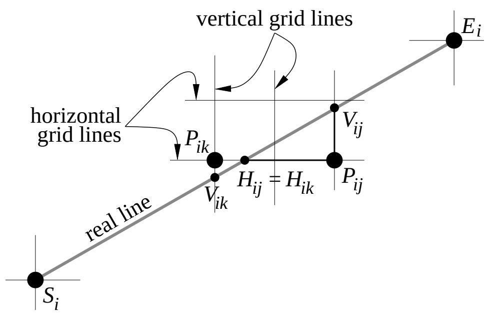


Figure 5.7: Isothetic distance of P_{ij} (from $\overline{\mathsf{S}_i\mathsf{E}_i}$) is $\overline{\mathsf{P}_{ij}\mathsf{V}_{ij}}$, which is greater than $\frac{1}{2}$, thereby making P_{ij} an error point, whereas, P_{ik} is not an error point, since the isothetic distance of P_{ik} is $\overline{\mathsf{P}_{ik}\mathsf{V}_{ik}}$, which is lesser than $\frac{1}{2}$.

point on $\overline{\mathsf{S}_i\mathsf{E}_i}$ horizontally left (or right) of P_{ij} be H_{ij}. Also, let $\mathsf{S}_i = (x_s, y_s)$, $\mathsf{E}_i = (x_e, y_e)$, $\mathsf{P}_{ij} = (x_p, y_p)$, $\mathsf{V}_{ij} = (x_v, y_v)$, and $\mathsf{H}_{ij} = (x_h, y_h)$, where the suffixes ($i$ and j) are dropped for notational simplicity. Now, the slope of $\overline{\mathsf{S}_i\mathsf{E}_i}$ is given by $m = (y_e - y_s)/(x_e - x_s)$. In addition, as shown in Fig. 5.7, we have $x_p = x_v$ and $y_p = y_h$, whence $\mathsf{V}_{ij} = (x_p, y_v)$ and $\mathsf{H}_{ij} = (x_h, y_p)$.

So $|y_v - y_p|$ is the vertical distance and $|x_h - x_p|$ is the horizontal distance of P_{ij} from $\overline{\mathsf{S}_i\mathsf{E}_i}$. Now it can be said that P_{ij} is not a digitization point of $\overline{\mathsf{S}_i\mathsf{E}_i}$, or, in other words, P_{ij} is an error point, if and only if $\min\{|x_h - x_p|,\ |y_v - y_p|\} > \frac{1}{2}$. Hence, w.l.o.g., if $|x_h - x_p| \geq |y_v - y_p|$, i.e., if $|x_e - x_s| \geq |y_e - y_s|$, then P_{ij} is an error point if and only if $|y_v - y_p| > \frac{1}{2}$. Now, since V_{ij} and S_i are on the same line $\overline{\mathsf{S}_i\mathsf{E}_i}$ with slope m, we have

$$\begin{aligned}
& mx_p - y_v = mx_s - y_s, \\
\text{or, }\ & y_v = m(x_p - x_s) + y_s = \{(y_e - y_s)(x_p - x_s) + y_s(x_e - x_s)\}/(x_e - x_s), \\
\text{or, }\ & y_v = \{y_e(x_p - x_s) + y_s(x_e - x_p)\}/(x_e - x_s),
\end{aligned}$$

whence the condition for P_{ij} to be an error point is given by

$$\begin{aligned}
& |y_v - y_p| > \frac{1}{2}, \\
\text{or, }\ & \frac{|y_e(x_p - x_s) + y_s(x_e - x_p) - y_p(x_e - x_s)|}{|x_e - x_s|} > \frac{1}{2}, \\
\text{or, }\ & 2\,|y_e(x_p - x_s) + y_s(x_e - x_p) - y_p(x_e - x_s)| > |x_e - x_s|, \\
\text{or, }\ & 2\,|(x_e - x_s)(y_e - y_p) - (x_e - x_p)(y_e - y_s)| > |x_e - x_s|, \\
\text{or, }\ & 2\,|\mathbf{x}_{\mathsf{ES}}\mathbf{y}_{\mathsf{EP}} - \mathbf{x}_{\mathsf{EP}}\mathbf{y}_{\mathsf{ES}}| - |\mathbf{x}_{\mathsf{ES}}| > 0, \text{if }\ |\mathbf{x}_{\mathsf{ES}}| \geq |\mathbf{y}_{\mathsf{ES}}|, \qquad (5.6)
\end{aligned}$$

where, $\mathbf{x}_{\mathsf{ES}} = x_e - x_s,\ \mathbf{y}_{\mathsf{ES}} = y_e - y_s$, etc.

On the contrary, if $|x_h - x_p| < |y_v - y_p|$, i.e., if $|x_e - x_s| < |y_e - y_s|$, then in order that P_{ij} becomes an error point, $|x_h - x_p|$ should be greater than $\frac{1}{2}$; wherefore, by similar deduction, we have the condition

$$2\left|\mathbf{x}_{\mathsf{ES}}\mathbf{y}_{\mathsf{EP}} - \mathbf{x}_{\mathsf{EP}}\mathbf{y}_{\mathsf{ES}}\right| - \left|\mathbf{y}_{\mathsf{ES}}\right| > 0, \text{if } \left|\mathbf{x}_{\mathsf{ES}}\right| < \left|\mathbf{y}_{\mathsf{ES}}\right|. \tag{5.7}$$

Combining Eqn. 5.6 and Eqn. 5.7, therefore, it can be said that $\mathsf{P}_{ij} := (x_p, y_p)$ is an error point corresponding to the line $\overline{\mathsf{S}_i\mathsf{E}_i}$, if and only if

$$2\left|\mathbf{x}_{\mathsf{ES}}\mathbf{y}_{\mathsf{EP}} - \mathbf{x}_{\mathsf{EP}}\mathbf{y}_{\mathsf{ES}}\right| - \max\left\{\left|\mathbf{x}_{\mathsf{ES}}\right|, \left|\mathbf{y}_{\mathsf{ES}}\right|\right\} > 0. \tag{5.8}$$

Although Eqn. 5.8 is not required at any stage in our algorithm to find the ADSS or to find the approximate polygon, the condition given by this equation, however, enables us to check whether or not P_{ij} is an error point without using any floating-point arithmetic. Further, the realization of Eqn. 5.8 does not even require any integer multiplication, the explanation being as follows.

In order to check the error points in the ith ADSS, we start from $\mathsf{P}_{i1} := (x_1, y_1) = \mathsf{S}_i := (x_s, y_s)$, which is (trivially) not an error point. W.l.o.g., if we consider that $0 \le m \le 1$, then for the next point $\mathsf{P}_{i2} := (x_2, y_2)$, we have $x_2 = x_1 + 1$, and $y_2 = y_1$ or $y_1 + 1$. That is, in general, for the point $\mathsf{Q} := \mathsf{P}_{i,j+1}$ following P_{ij}, we have $x_{j+1} = x_j + 1$, and $y_{j+1} = y_j + a$, where $a \in \{0, 1\}$. It may be observed that, this owes to the property (R1) of DSS (and is also possessed by an ADSS), as explained in Sec. 5.2.1. Thus, the parameter $\phi_{j+1} = \mathbf{x}_{\mathsf{ES}}\mathbf{y}_{\mathsf{EQ}} - \mathbf{x}_{\mathsf{EQ}}\mathbf{y}_{\mathsf{ES}}$ for Q is obtained from the parameter $\phi_j = \mathbf{x}_{\mathsf{ES}}\mathbf{y}_{\mathsf{EP}} - \mathbf{x}_{\mathsf{EP}}\mathbf{y}_{\mathsf{ES}}$ for P, using one or two integer additions only, as follows.

$$\begin{aligned}\phi_{j+1} &= \mathbf{x}_{\mathsf{ES}}(\mathbf{y}_{\mathsf{EP}} - a) - (\mathbf{x}_{\mathsf{EP}} - 1)\mathbf{y}_{\mathsf{ES}} \\ &= \phi_j - a\mathbf{x}_{\mathsf{ES}} + \mathbf{y}_{\mathsf{ES}}, \text{where } a \in \{0, 1\}.\end{aligned} \tag{5.9}$$

It may be noted that, for all other possible intervals of m, the erroneousness of a point on an ADSS can be verified similarly with an appropriate form of Eqn. 5.9.

5.3 Polygonal Approximation

Extraction of the ADSS for each curve $\mathcal{C}_k$ in the given set (binary image) $\mathcal{I} := \{\mathcal{C}_k\}_{k=1}^{K}$ of DC generates an ordered set of ADSS, namely $\mathcal{A}_k := \langle \mathbf{L}_i^{(k)} \rangle_{i=1}^{n_k}$, corresponding to $\mathcal{C}_k$. In each such set $\mathcal{A}_k$, several consecutive ADSS may occur, which are approximately collinear and, therefore, may be combined together to form a single segment.

Let $\langle \mathbf{L}^{(k)} \rangle_{j_1}^{j_2}$ be the maximal (ordered) subset of the ADSS starting from $\mathbf{L}_{j_1}^{(k)}$ that conforms to some approximation criterion. Then these $j_2 - j_1 + 1$ segments in A_k are combined together to form a single straight line segment starting from the start point of $\mathbf{L}_{j_1}^{(k)}$ and ending at the end point of $\mathbf{L}_{j_2}^{(k)}$. This procedure is repeated for all such maximal subsets of $\mathcal{A}_k$ in succession to obtain the polygonal approximation (in case $\mathcal{C}_k$ is a closed curve) or polychain approximation (in case $\mathcal{C}_k$ is open), namely $\mathcal{P}_k$, corresponding to $\mathcal{C}_k$.

In the proposed algorithm, depending on the approximation criterion, we have used a greedy method of approximating the concerned curve $\mathcal{C}_k$ starting from the very first ADSS in $\mathcal{A}_k$. Determination of a minimal set of DSS (and a minimal set $\mathcal{A}_k$ of ADSS, thereof) corresponding to a given curve $\mathcal{C}_k$ is known to be computationally intensive [Klette and Rosenfeld (2004a), Rosenfeld and Klette (2001)]; so for real-time applications, a near-optimal but speedy solution is often preferred than the optimal one.

5.3.1 Approximation Criterion

There are several variants of approximation criteria available in the literature [Rosin (1997)]. We have tested our algorithms with two variants of the approximation measures based on area deviation [Wall and Danielsson (1984)]. Both the algorithms are realizable in purely integer domain subject to few primitive operations only. The approximation criterion is defined w.r.t. the *approximation parameter* or *error tolerance*, denoted by τ, as follows.

5.3.1.1 Cumulative Error (criterion $\mathrm{C}_{\sum}$)

Let $\langle \mathbf{L}^{(k)} \rangle_{j_1}^{j_2}$, be an ordered subset of A_k as discussed above. Then the ADSS ($j_2 - j_1 + 1$ in number) in A_k are replaced by a single straight line segment starting from the start point of $\mathbf{L}_{j_1}^{(k)}$ and finishing at the end point of $\mathbf{L}_{j_2}^{(k)}$, if:

$$\sum_{j=j_1}^{j_2-1} \left| \triangle \left(s(\mathbf{L}_{j_1}^{(k)}), e(\mathbf{L}_{j}^{(k)}), e(\mathbf{L}_{j_2}^{(k)}) \right) \right| \leq \tau d_{\top} \left(s(\mathbf{L}_{j_1}^{(k)}), e(\mathbf{L}_{j_2}^{(k)}) \right) \tag{5.10}$$

where, $s(\mathbf{L}_{j}^{(k)})$ and $e(\mathbf{L}_{j}^{(k)})$ represent the respective start point and the end point of the ADSS $\mathbf{L}_{j}^{(k)}$, etc. The start point of $\mathbf{L}_{j}^{(k)}$ coincides with the end point of the preceding ADSS, if any, in $\mathcal{A}^k$, and the end point of $\mathbf{L}_{j}^{(k)}$ coincides with the succeeding one, if any. In Eqn. 5.10, $|\triangle (p,q,r)|$ denotes twice the magnitude of area of the triangle with vertices $p := (x_p, y_p)$, $q := (x_q, y_q)$, and $r := (x_r, y_r)$, and $d_{\top}(p,q)$ the maximum isothetic distance between two points p and q. Since all these points are in two-dimensional digital space, the above measures are computable in the integer domain as shown in the following equations.

$$d_{\top}(p,q) = \max\{|x_p - x_q|, |y_p - y_q|\} \tag{5.11}$$

$$\triangle(p,q,r) = \begin{vmatrix} 1 & 1 & 1 \\ x_p & x_q & x_r \\ y_p & y_q & y_r \end{vmatrix} \tag{5.12}$$

From Eqn. 5.12, it is evident that $\triangle(p,q,r)$ is a determinant that gives twice the signed area of the triangle with vertices p, q, and r. Hence the ADSS in the given subset are merged to form a single straight line segment, say $\widetilde{\mathbf{L}}$, provided the cumulative area of the triangles ($j_2 - j_1$ in number), having $\widetilde{\mathbf{L}}$ as base and the third vertices being the end points of the ADSS (excepting the last one) in the subset $\langle \mathbf{L}^{(k)} \rangle_{j_1}^{j_2}$, does not exceed the area of the triangle with base $\widetilde{\mathbf{L}}$ (isothetic length) and height τ.

5.3.1.2 Maximum Error (criterion $\mathrm{C}_{\max}$)

With similar notations as mentioned above, using the maximum error criterion, the ADSS in $\langle \mathbf{L}^{(k)} \rangle_{j_1}^{j_2}$ would be replaced by a single piece, provided the following condition is satisfied.

$$\max_{j_1 \leqslant j \leqslant j_2 - 1} \left| \triangle\left(s(\mathbf{L}_{j_1}^{(k)}), e(\mathbf{L}_{j}^{(k)}), e(\mathbf{L}_{j_2}^{(k)}) \right) \right| \\ \leq \tau d_{\top}\left(s(\mathbf{L}_{j_1}^{(k)}), e(\mathbf{L}_{j_2}^{(k)}) \right) \tag{5.13}$$

The rationale of considering two such criteria is as follows. Since we would be replacing a number of ADSS, which are almost straight, and more importantly, are not ordinary digital curves of arbitrary patterns and arbitrarily curvatures, the end point of each ADSS makes a triangle with the replacing segment, namely $\widetilde{\mathbf{L}}$. So the sum of the areas of triangles formed by the end points of these ADSS in combination with the replacing line $\widetilde{\mathbf{L}}$ gives a measure of error due to approximation of *all ADSS in* $\langle \mathbf{L}^{(k)} \rangle_{j_1}^{j_2}$ by $\widetilde{\mathbf{L}}$. Alternatively, if we are guided by the worst case approximation, that is, if the mostly digressing ADSS is considered to estimate the error, then the maximum of the areas of these triangles should be considered as the error measure for approximation of *worst ADSS in* $\langle \mathbf{L}^{(k)} \rangle_{j_1}^{j_2}$ by $\widetilde{\mathbf{L}}$.

Empirical observations as reported in Sec. 5.4, reveal that the above two criteria are essentially similar in the sense that they produce almost identical polygons for different DC for different values of the error tolerance (i.e., τ). This is quite expected as far as the output is concerned.

As mentioned earlier in Sec. 5.3, to construct polygonal approximation we consider the start point of the first ADSS (i.e., $\mathbf{L}_{j_1}^{(k)}$), and the end point of the last ADSS (i.e., $\mathbf{L}_{j_2}^{(k)}$). This can be justified as follows.

Fact 1. The sum (for criterion $\mathrm{C}_{\sum}$) or the maximum (for criterion $\mathrm{C}_{\max}$) of the isothetic distances of the end points of each ADSS from the replacing line $\widetilde{\mathbf{L}}$ never exceeds the specified error tolerance τ. This follows easily on expansion of the left hand side of the corresponding Eqns. 5.10 and 5.13, and from the fact that the term $d_\top\left(s(\mathbf{L}_{j_1}^{(k)}), e(\mathbf{L}_{j_2}^{(k)})\right)$ represents the isothetic length of $\widetilde{\mathbf{L}}$.

Fact 2. Since each ADSS $\mathbf{L}_j^{(k)}$ is approximately a DSS, we consider that $\nexists p \in \mathbf{L}_j^{(k)}$ such that the isothetic distance of p from DSL passing through the end points of $\mathbf{L}_j^{(k)}$ exceeds unity (as testified in our experiments). Although for sufficiently long ADSS, this may not hold for the underlying conditions (c1) and (c2) as stated in Sec. 5.2; however, in our experiments with real world images, this was found to hold. In the case of any violation, some heuristics may be employed to find the error points and to find smaller ADSS to resolve the problem.

5.3.2 Algorithm for Polygonal Approximation

The algorithm for polygonal approximation of a sequence of ADSS in the set $\mathcal{A}$, using the approximation criterion of Eqn. 5.10, is described in Fig. 5.8. To take care of the criterion $\mathrm{C}_{\max}$ of Eqn. 5.13, a similar procedure may be written.

Final Time Complexity: As explained in Sec. 5.2.2, the time complexity for extracting the ADSS in a set of DC, $\mathcal{I} := \{\mathcal{C}_k\}_{k=1}^{K}$, is given by $\Theta(N_1) + \Theta(N_2) + \ldots + \Theta(N_K) = \Theta(N)$, where $N (= N_1 + N_2 + \ldots + N_K)$ is the total number of points representing $\mathcal{I}$. Now, in the algorithm MERGE-ADSS, we have considered only the ordered set of vertices of the ADSS corresponding to the curves, so that the worst-case time complexity in this stage is linear in N. Hence, the overall time complexity is given by $\Theta(N) + \mathcal{O}(N) = \Theta(N)$, whatsoever may be the error of approximation τ.

5.3.3 Quality of Approximation

The goodness of an algorithm for polygonal approximation is quantified, in general, by the amount of discrepancy between the approximate polygon(s) (or polychain(s)) and the original set of DC. There are several measures to assess the approximation of a curve $\mathcal{C}_k$, such as

```
**Algorithm** MERGE-ADSS($\mathcal{A}, n, \tau$)
**Steps:**
 1. **for** $m \leftarrow 1$ to $n$
 2.     **for** $S \leftarrow 0, i \leftarrow 1$ to $(n - m - 1)$
 3.         $S \leftarrow S + \triangle(\mathcal{A}[m], \mathcal{A}[m+i], \mathcal{A}[m+i+1])$
 4.         $dx \leftarrow |\mathcal{A}[m].x - \mathcal{A}[m+i+1].x|$
 5.         $dy \leftarrow |\mathcal{A}[m].y - \mathcal{A}[m+i+1].y|$
 6.         $d \leftarrow \max\{dx, dy\}$
 7.         **if** $S \leq d\tau$
 8.             delete $\mathcal{A}[m+i]$ from $\mathcal{A}$
 9.         **else**
10.             **break**
11.     $m \leftarrow m + i - 1$
```

Figure 5.8: Algorithm MERGE-ADSS for polygonal approximation of a sequence of ADSS in $\mathcal{A}$ using criterion $\mathrm{C}_{\max}$.

(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 by 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

$$dev_\perp(p \to \widetilde{p}) = \min\{|x - \widetilde{x}|, |y - \widetilde{y}|\}. \tag{5.14}$$

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

$$f(\tau, d_\perp) = \frac{1}{N} \left|\{p \in \mathcal{I} : dev_\perp(p \to \widetilde{p}) = d_\perp\}\right|, \tag{5.15}$$

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. 5.15 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

Table 5.2: Comparison of DSS and ADSS extraction algorithms on different images

Image name & size (pixels)	No. of points	P.A.$^{\#}$ (secs.)	No. of segs.		Avg. length		CPU time (secs.)		
			DSS	ADSS	DSS	ADSS	DSS	ADSS	$\{\mathcal{P}\}^{\star}$
bird-nestlings 480×320	3041	6.38	902	327	3.37	9.30	5.42	0.17	0.05
climber 320×330	2750	5.92	1170	419	2.35	6.56	6.74	0.20	0.05
india 325×285	2552	7.15	1735	597	1.47	4.27	9.06	0.29	0.06
leaf 240×256	1312	3.88	341	106	3.85	12.38	2.17	0.08	0.02
spider 292×286	1767	4.20	583	157	3.03	11.25	3.93	0.11	0.03
test-001 140×1050	2809	6.26	858	276	3.27	10.18	4.48	0.14	0.04
vase 408×333	6066	10.81	1972	681	3.07	8.91	14.52	0.43	0.10

$^{\#}$CPU time for polygonal approximation using area deviation [Wall and Danielsson (1984)] without ADSS.
*average CPU time for polygonal approximation with ADSS using criteria $C_{\sum}$ and $C_{\max}$, and $\tau = 1-20$.

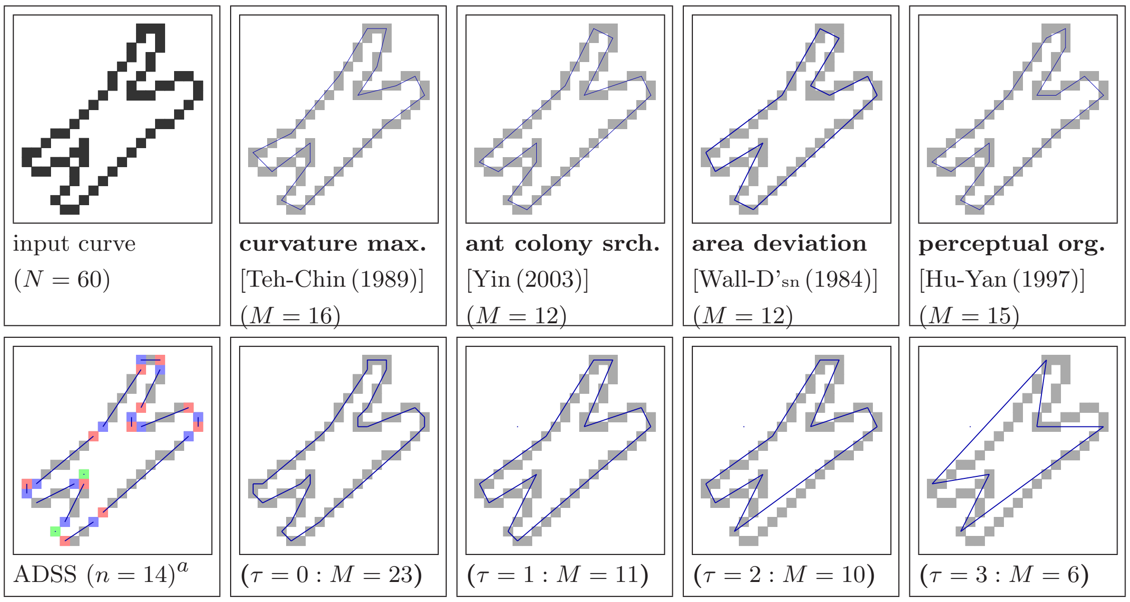
study on the nature of the error frequency distribution for a sufficiently large population of arbitrary digital curves may be, therefore, a promising area of theoretical analysis of polygonal approximation of digital curves.

5.4 Experimental Results

We have implemented one algorithm for DSS extraction based on DSS recognition [Creutzburg *et al.* (1982)][1], and the proposed algorithm Extract-ADSS, in C on the SunOS Release 5.7 Generic of Sun_Ultra 5_10, Sparc, 233 MHz. The results of the two algorithms on several binary image files of curves and contours are reported here.

We have also compared the results of our method with those of some existing methods as shown in Table 5.1. In Fig. 5.9 and in Fig. 5.10, we have presented the comparative re-

[1]We have implemented the DSS-recognition algorithm [Creutzburg *et al.* (1982)] given in [Klette and Rosenfeld (2004a)], and have not incorporated the errata given in [Klette (2004)].



[a] A blue point indicates the start point and a red point indicates the end point of an ADSS. A green point indicates an ADSS with its end point coinciding with its start point, which is a degenerate case arising out of our consideration of the chain code of a start point w.r.t. the end point of the previous ADSS in the sequence.

Figure 5.9: Results of polygonal approximations on "chromosome" by the existing methods mentioned in Table 5.1 and by the proposed method, MERGE-ADSS (using criterion $C_{\sum}$), after extraction of ADSS (using algorithm EXTRACT-ADSS) from the input curve.

sults on polygonal approximation of two benchmark curves, namely, "chromosome" (given in Appendix B of Teh and Chin (1989)), and "semicir" respectively. The results show that our method compares favorably with others when we consider $\tau = 1$ in MERGE-ADSS.

It may be noted that, for $\tau = 2, 3, \ldots$, the approximation obtained by our method requires fewer number of vertices (M), but at the cost of some error incurred, and may not be profitable for small curves like "chromosome" (and the other test/benchmark curves considered in the existing works [Rosin (1997), Teh and Chin (1989), Wall and Danielsson (1984), Wu (1984), Yin (2003)]). However, for sufficiently large curves (a few being presented in Figs. 5.14, 5.16, 5.17, and 5.18), slackening of τ reduces the number of vertices to the desired limit, as explained later in this section.

In our implementation, the procedure for extraction of the straight line segments (DSS/ADSS) from a given curve starts from one end of the curve if it is an open-end, or starts from a point with chain code (w.r.t. its neighbors) not satisfying property (R1) if it is a closed one; in the case of failure of the above two criteria, the start point is chosen arbitrarily.

The results for the algorithm on DSS extraction and that on ADSS extraction for an image of a leaf (cropped and zoomed for pixel-wise clarity) are shown in Fig. 5.11. Earlier, in Fig. 5.2, similar results have been shown for another image (cropped). It is evident from the extracted set of DSS and that of ADSS that, not only an extracted ADSS is reasonably straight, but also the number of ADSS is appreciably smaller than that of DSS, a result that is used to expedite the subsequent algorithm for polygonal approximation. Because of the recursive nature (apropos the run of run lengths) of the DSS extraction algorithms, they are inherently much slower than the ADSS algorithm, as reflected by the CPU times reported in Table 5.2. In this table, we have also given the CPU time required for polygonal approximation by the algorithm based on area deviation [Wall and Danielsson (1984)] (by considering *all points* in the input set of curves instead of end points of their ADSS) to show how ADSS extraction prior to polygonal approximation accelerates the process. We have also furnished some other significant parameters to justify the use of ADSS in our polygonal approximation algorithm. Since such an algorithm will run faster for a smaller set of segments (whether exact or approximate), the set of ADSS may be used instead of that of DSS.

We have tested our polygonal approximation algorithms on two classes of test images — one with 500 sets containing randomly generated DC of lengths varying from $N = 500$ to $N = 2000$, and another class with 20 sets of hand-drawn curves — one such hand-drawn test set is shown in Fig. 5.12. In addition, results on 150 real-world binary images (edge maps) of various natures have been generated.

Fig. 5.12 demonstrates the polygonal approximation with the criterion $\mathrm{C}_{\sum}$. The input image consists of a single (hand-drawn) curve, with seven orientations at 15^{O} increments, such as $0^{\mathrm{O}}, 15^{\mathrm{O}}, \ldots, 90^{\mathrm{O}}$. It is interesting to notice that as the approximation tolerance τ continues to increase, the sets of vertices tend to become increasingly similar, indicating the invariance of polygonal approximation to rotation (or, orientation) when τ becomes sufficiently high. Further, due to the anisotropic nature of the square grid and the rounding-off error in digitization and thinning [Gonzalez and Woods (1993), Rosenfeld and Kak (1982)] applied on a rotated curve, the difference chain code [Rosenfeld and Kak (1982)] of the rotated curve is likely to differ from that of the original curve. Hence, for smaller values of τ, the difference in such chain codes representing two orientations plays a critical role in ascertaining approximate straightness of a small component (whose "smallness" is decided by τ). A larger value of τ, however, compensates for such local anomalies in a DC, and thus produces almost identical polygons.

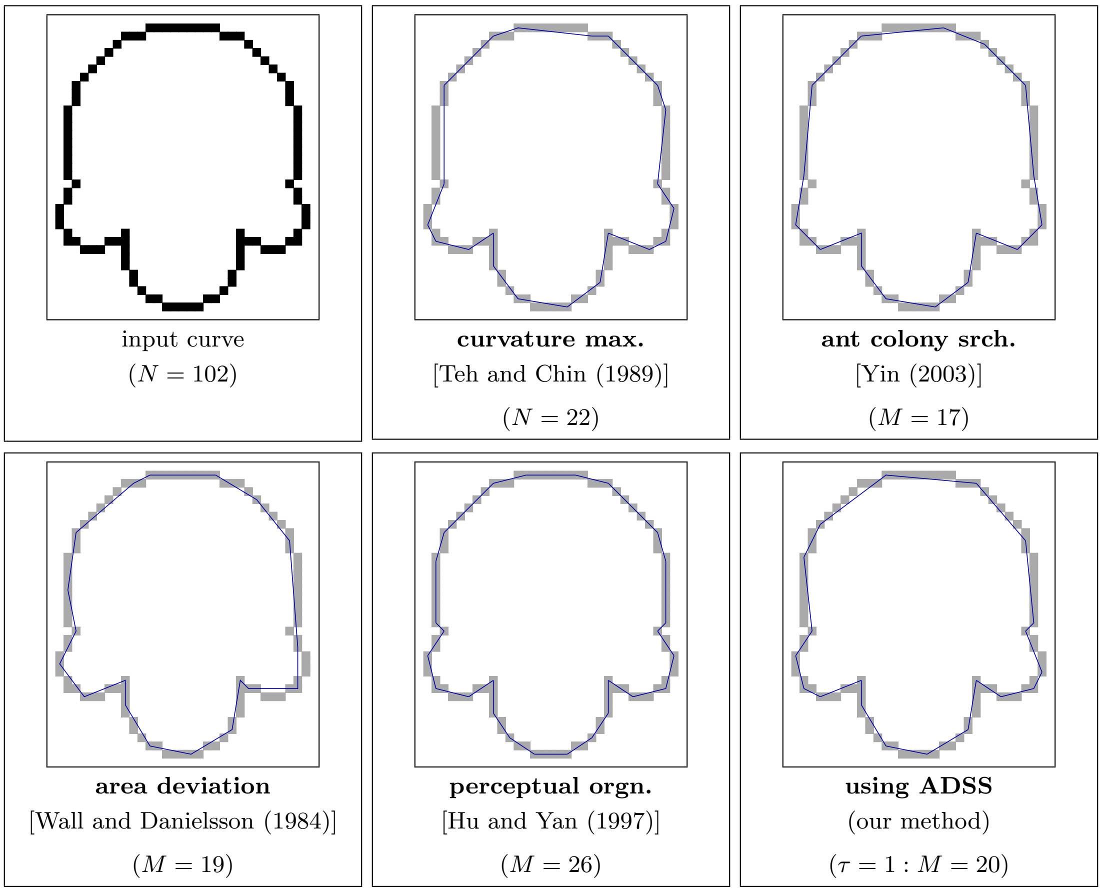


Figure 5.10: Results on "semicir": by some existing methods mentioned in Table 5.1 and by the proposed method.

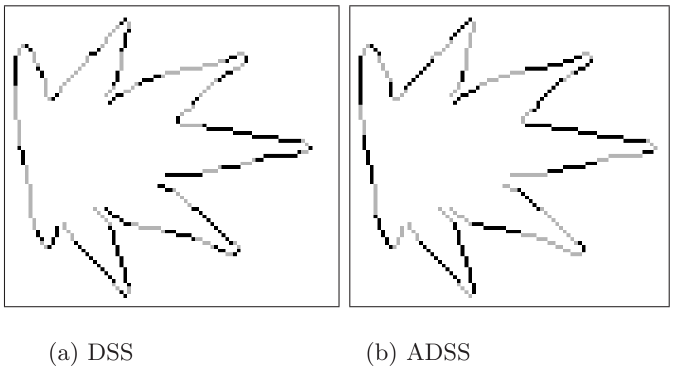


Figure 5.11: Set of DSS and that of ADSS extracted from the image of a leaf (cropped).

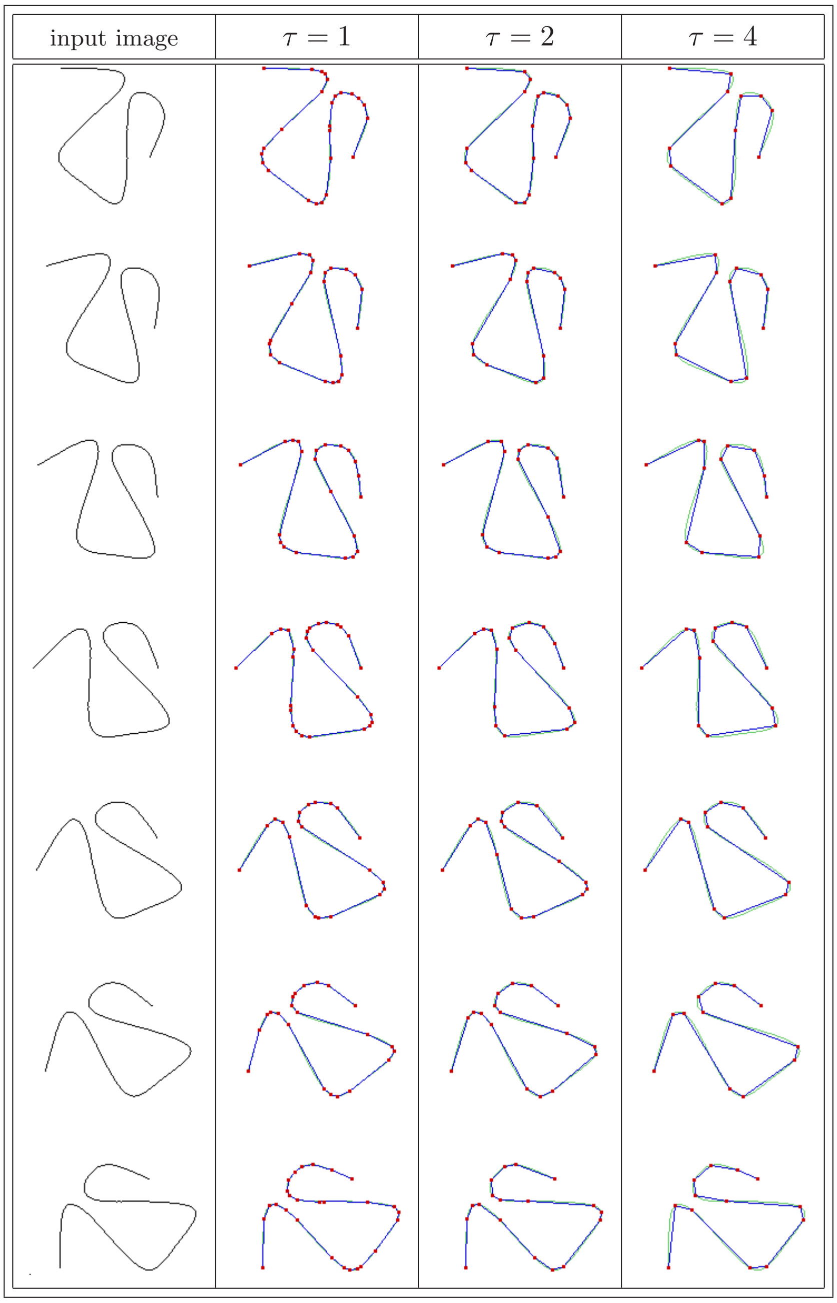


Figure 5.12: Polygonal approximations for a set of small test images, using criterion $C_{\sum}$. (The vertices of the approximate polygon have been shown in red, the edges in blue, and the original curve in faded green. There are 6 rotations, at 15^{o} increments, of the first object in the input set.)

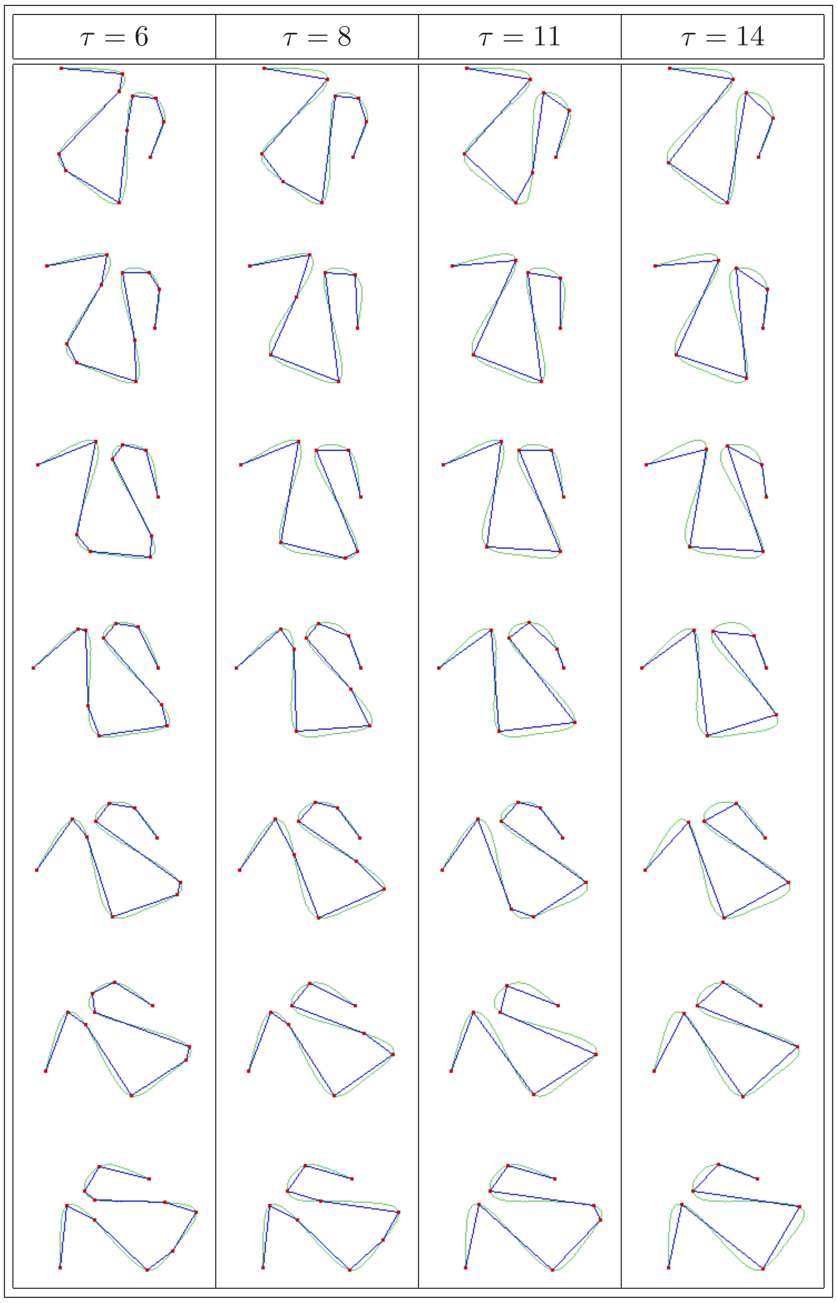
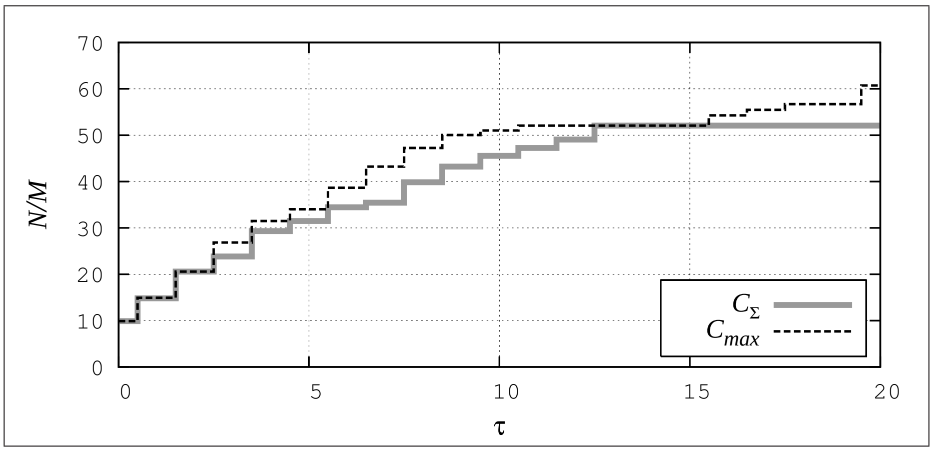
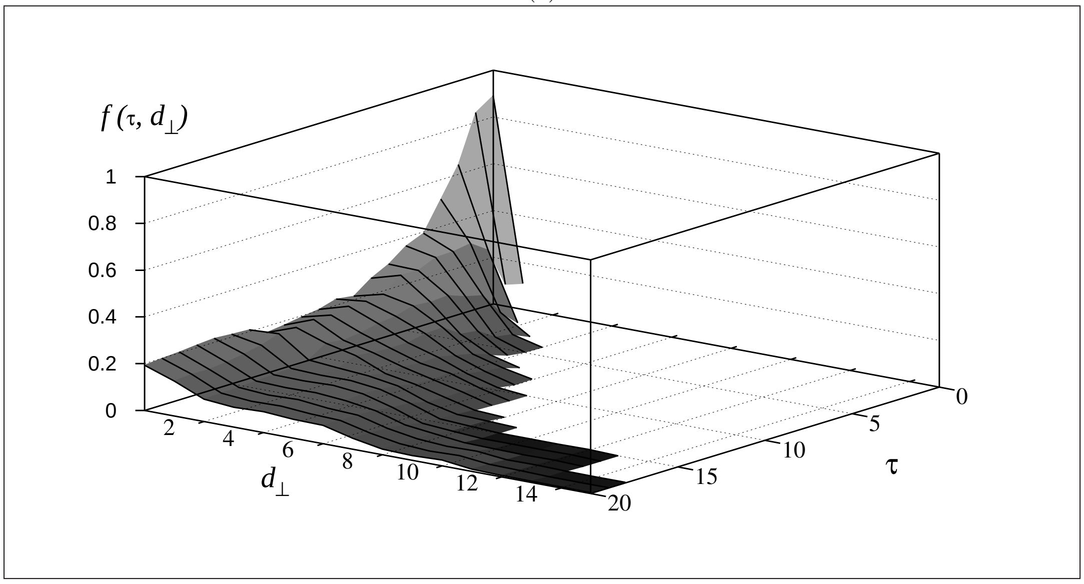


Figure 5.12 (continued): It is evident from the above results that the polygonal approximation of the object becomes almost invariant (w.r.t. the object orientation) when the approximation parameter τ is sufficiently high. See text for further explanation.

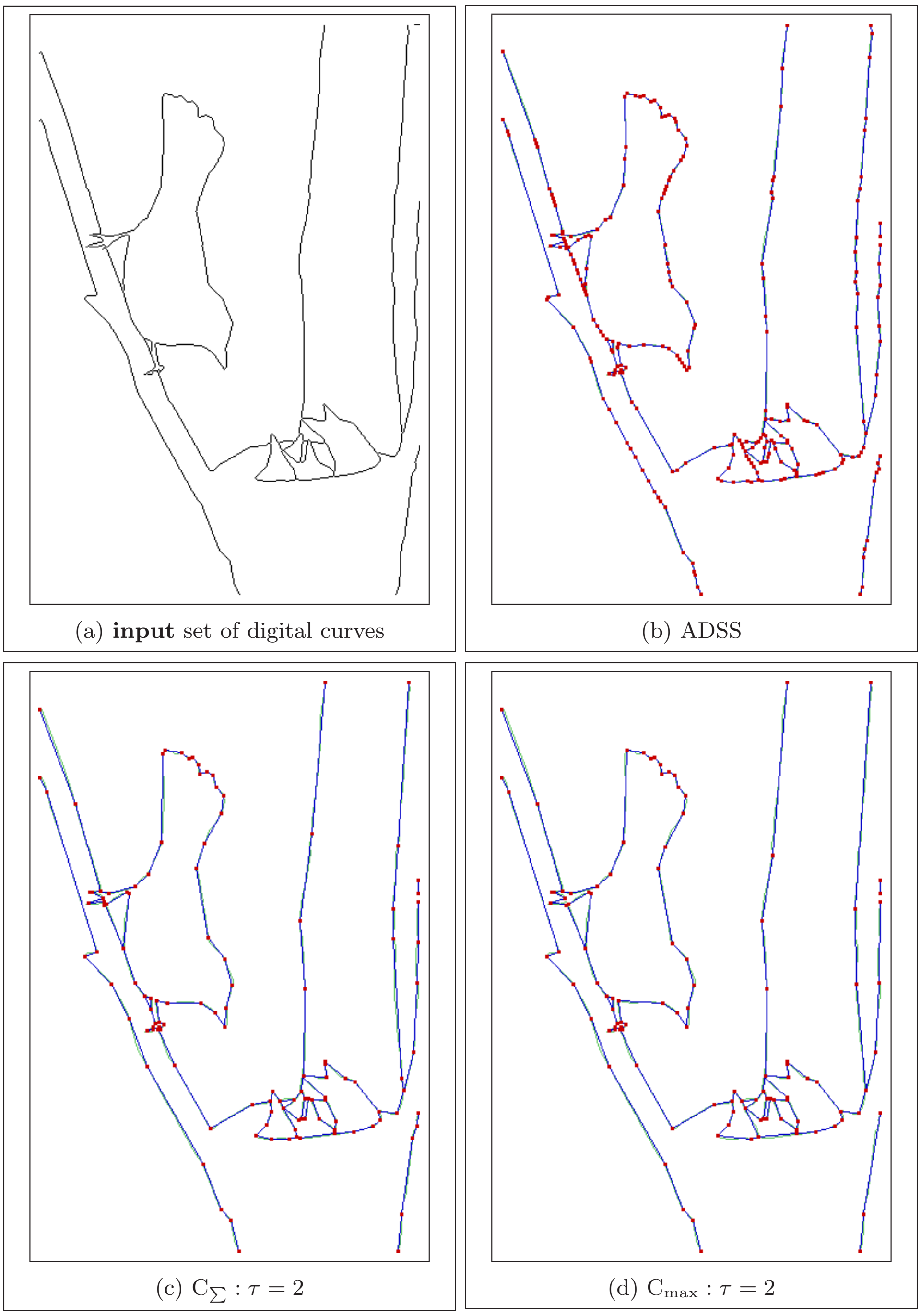


(a)



(b)

Figure 5.13: Quality of approximation for the "test-001" image shown in Fig. 5.12. In (a), two histogram profiles on N/M verses the error tolerance τ have been shown, where one histogram corresponds to criterion $\mathrm{C}_{\sum}$ and the other to criterion $\mathrm{C}_{\max}$. In (b), the plot on IEF $:= f(\tau, d_\perp)$ verses the error tolerance τ and the isothetic distance $d_\perp$ corresponding to criterion $\mathrm{C}_{\sum}$ has been shown, and that corresponding to criterion $\mathrm{C}_{\max}$ is very much similar. It may be noted that, here $\tau = 0$ corresponds to the number of vertices without any polygonal approximation (i.e., with ADSS only).

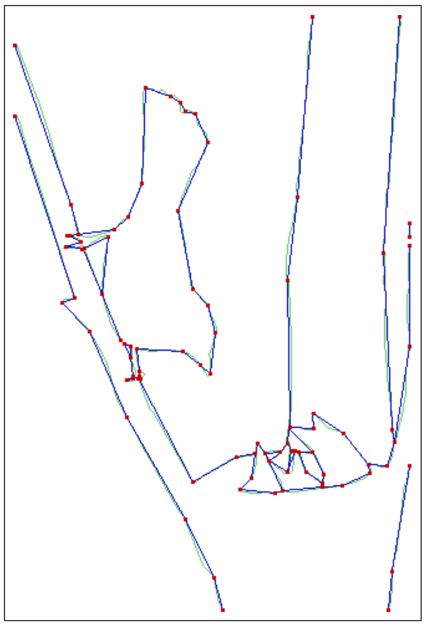

(a) **input** set of digital curves

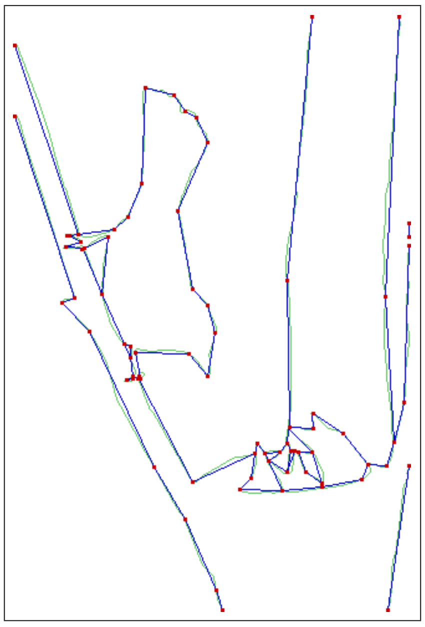
(b) ADSS

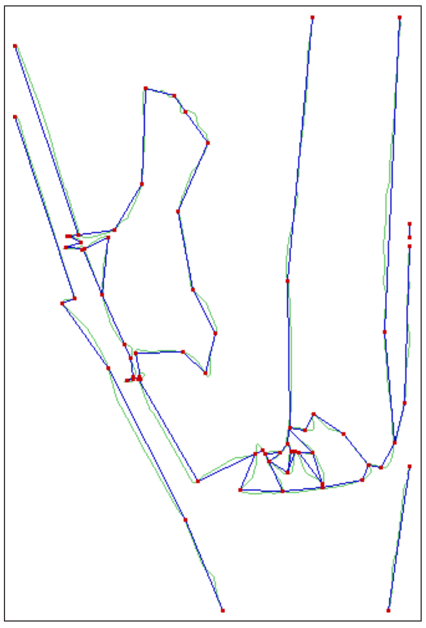
(c) $C_{\sum} : \tau = 2$

(d) $C_{\max} : \tau = 2$

Figure 5.14: Results on “bird-nestlings” image (shown in (a)) including extraction of ADSS (shown in (b) with end points highlighted in red).


(e) $C_{\sum} : \tau = 4$


(f) $C_{\max} : \tau = 4$


(g) $C_{\sum} : \tau = 8$

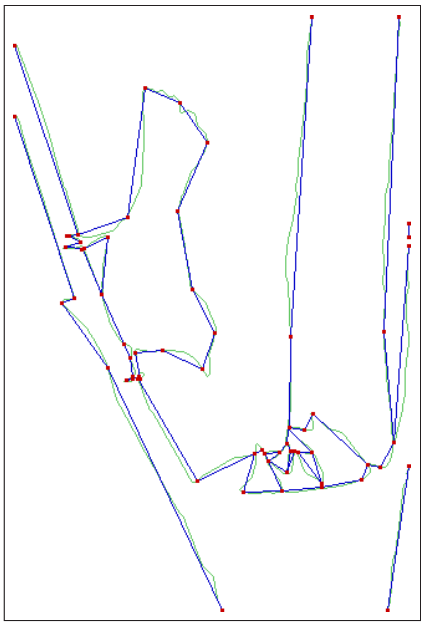
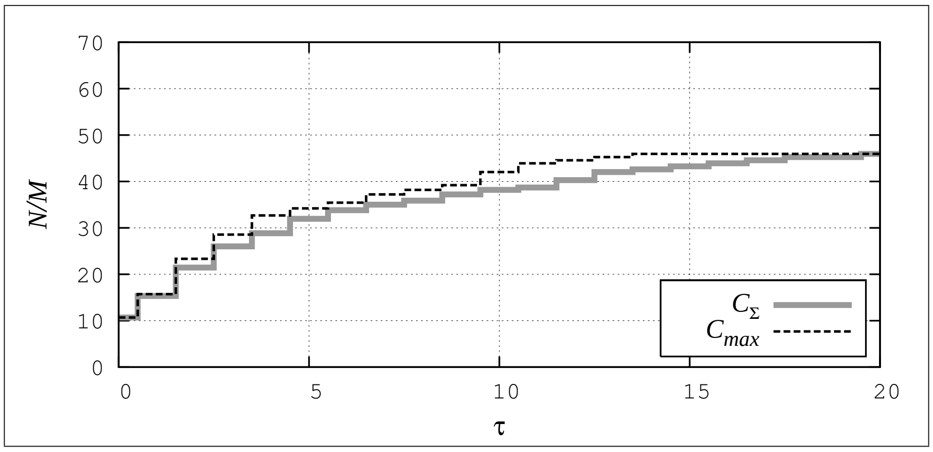
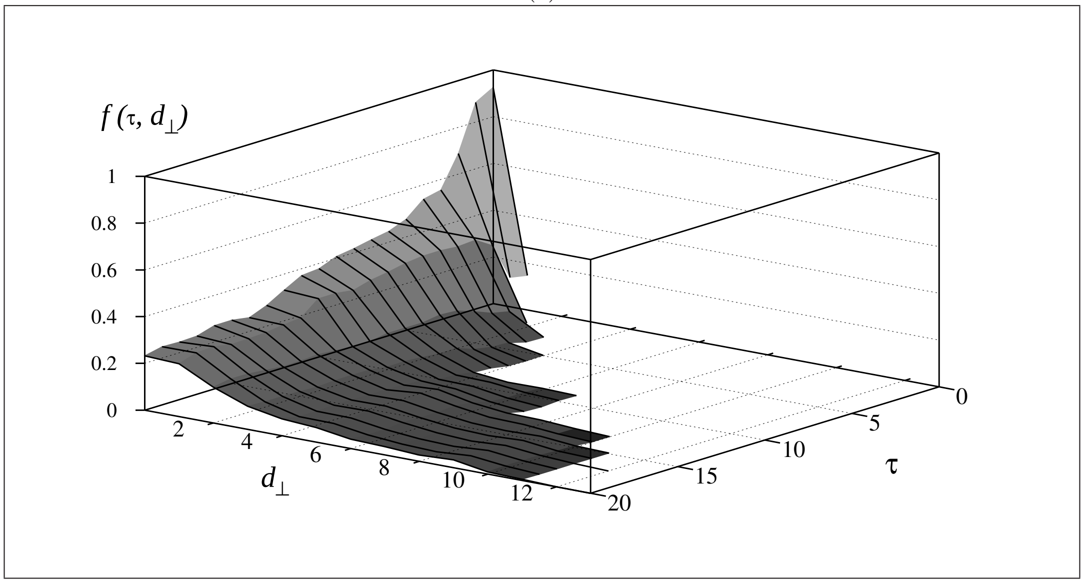

(h) $C_{\max} : \tau = 8$

Figure 5.14 (continued).



(a)



(b)

Figure 5.15: Quality of approximation for the “bird-nestlings” image shown in Fig. 5.14. See text and Fig. 5.13 for the significance of the plots.

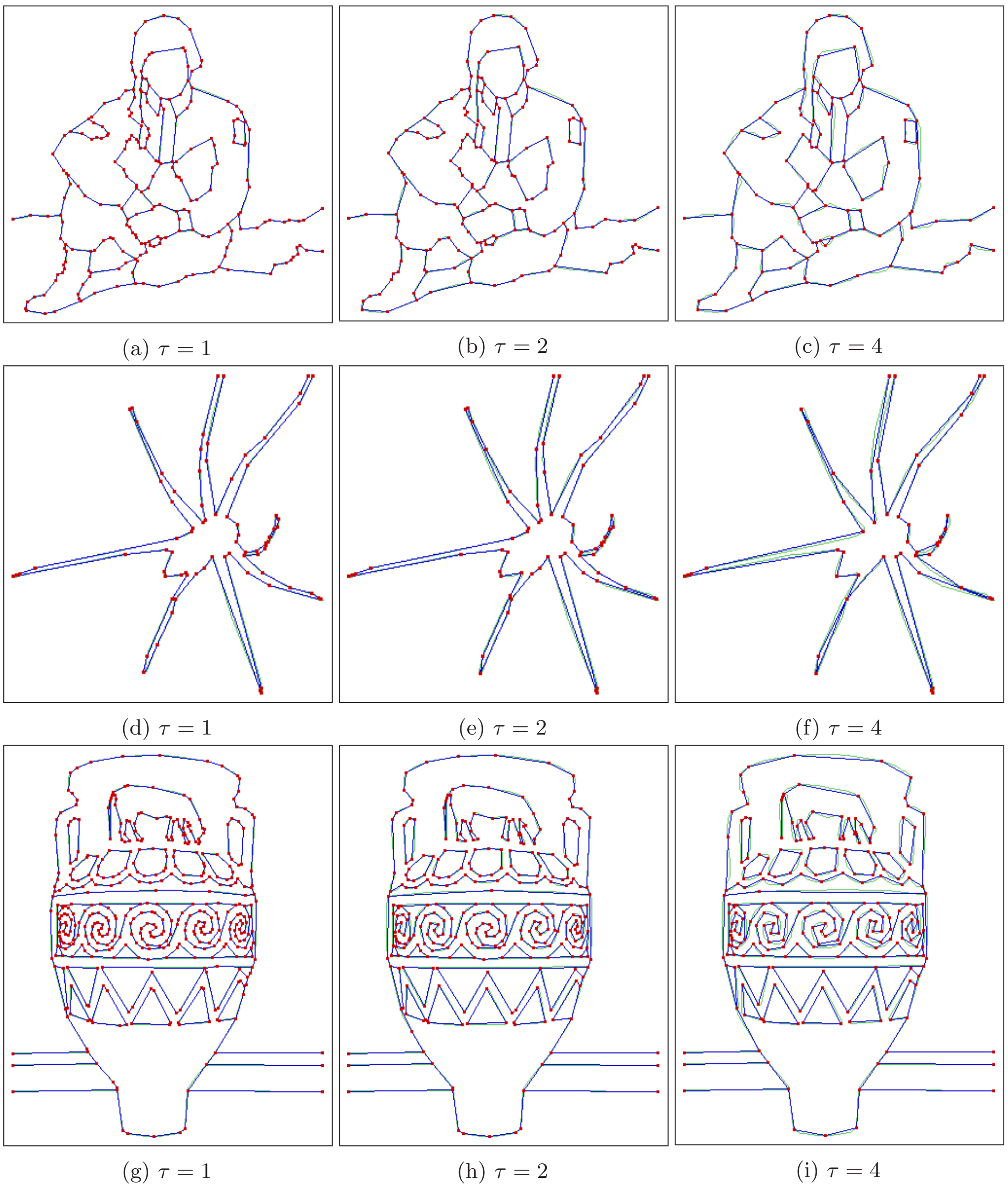


Figure 5.16: Approximate polygons for a few values of τ on “climber”, “spider”, and “vase” images, using criterion $C_{\sum}$.

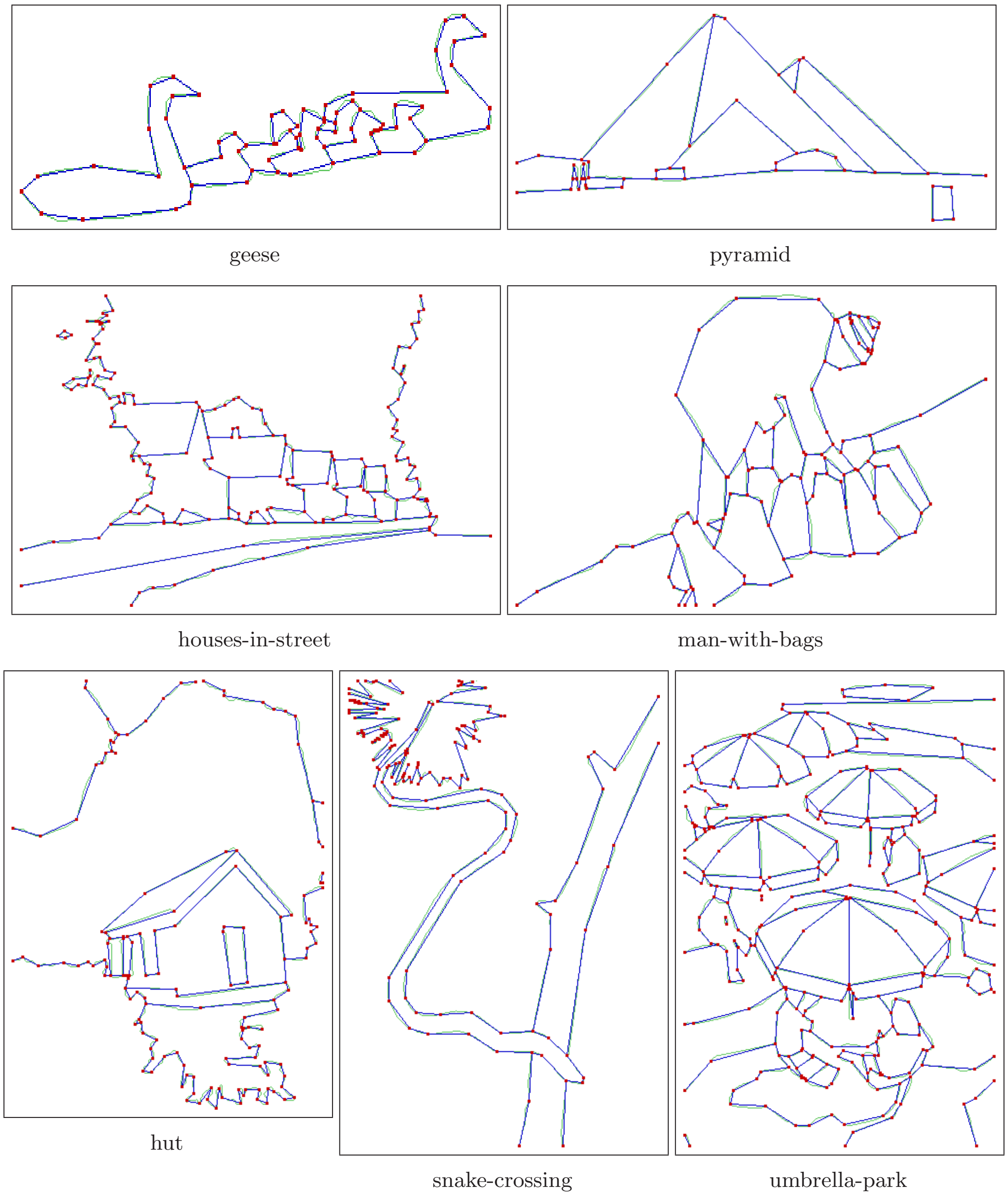


Figure 5.17: Approximate polygons for few more images with $\tau = 4$ and criterion $C_{\sum}$.

The two measures on quality of approximation for the "test-001" image have been rendered in Fig. 5.13. In Fig. 5.13(a), the profiles of two histograms demonstrating the distribution of the compression ratio CR $= N/M$ w.r.t. the error tolerance τ have been shown corresponding to criteria $\mathrm{C}_{\sum}$ and $\mathrm{C}_{\max}$. The stability of the algorithm on polygonal approximation is evident from the nature of the plots shown for both the criteria. For sufficiently high value of τ, the CR becomes almost asymptotically constant owing to the fact that nearly identical sets of vertices are produced for two close (and sufficiently high) values of τ, as shown in Fig. 5.12.

In Fig. 5.13(b), the plot on IEF using criterion $\mathrm{C}_{\sum}$ only has been shown, since that with criterion $\mathrm{C}_{\max}$ is very much similar. It may be observed from this plot that, for higher values of $\tau(\geq 6)$, the amount of maximum (isothetic) deviation (say, $d_{\perp(\max)}$) in all the seven curves in Fig. 5.12 falls quite short of the permissible limit (i.e., τ), and no less importantly, for a given value of τ, the number of points (say, $N_{d_{\perp}}$) with deviation $d_{\perp}$ (and IEF, thereof) decreases almost monotonically with $d_{\perp}$. A small subset of the numerical figures in our experiment with the "test-001" curve is furnished below.

τ	6	7	$\cdots$	16	18	20
$d_{\perp(\max)}$	4	5	$\cdots$	13	15	15
$N_{d_{\perp(\max)}}$	35	15	$\cdots$	2	9	13

The above data clearly shows that for the curves given in Fig. 5.12, for $\tau \geq 6$, we get $d_{\perp(\max)}$ appreciably smaller than τ, and more importantly, IEF for $d_{\perp} = d_{\perp(\max)}$ is very small (e.g., for $\tau = 20$, IEF $= 13/2809 = 0.0046$, etc.), which indicates that error values nearing $d_{\perp(\max)}$ have very low frequency, thereby reinforcing the expectation of high quality in the approximation process. This assertion is true as well for the other images, whether synthetic or real-world.

Since most of the images in practical applications involve real-world images, we have presented here results on few of such images considered in our experiment. Figure 5.14 exhibits the approximate polygons for "bird-nestlings" image for a few values of τ corresponding to each of the approximation criteria. It is apparent from this figure, that the resultant sets of polygons for the two criteria differ marginally (the set corresponding to criterion $\mathrm{C}_{\sum}$ is marginally better than $\mathrm{C}_{\max}$), and that the number of vertices falls off drastically in the lower end of the spectrum (of τ) and gradually becomes almost constant in the upper end of the spectrum. Further, that our algorithm in the case of this real-world "bird-nestlings" image has very closely similar characteristics with the test image (Fig. 5.12) as evidenced by the resemblance of its quality measures given in Fig. 5.15 with those in Fig. 5.13, certifies the efficiency and robustness of our algorithm, irrespective of the nature and complexity of the digital curves.

In Figs. 5.16, 5.17, and 5.18, approximate polygons for few other images with $\tau = 4$, corresponding to the approximation criterion $C_{\sum}$, have been shown. The quality measures for these images are not, however, included in this chapter for their almost similar patterns with those given in Figs. 5.13 and 5.15.

5.5 Conclusion

It is evident from the discussions and the algorithms, proposed here, that a set of ADSS extracted from a DC is significantly smaller in size than that of DSS extracted from the same, although each ADSS can be treated as sufficiently straight for various practical applications. Furthermore, the CPU time needed for ADSS extraction is remarkably lesser than that for DSS extraction.

Regarding polygonal approximation, the proposed method has been found to work well to determine a suboptimal solution from an arbitrary set of DC. It is evident from the experimental result and analysis that the polygon vertices are densely located in and around the regions with high curvature, and sparsely in the regions with low curvature, owing to the fact that the length of an ADSS (alternatively, a DSS) is small in the former region but high in the latter.

The major contributions of our work are summarized as follows:

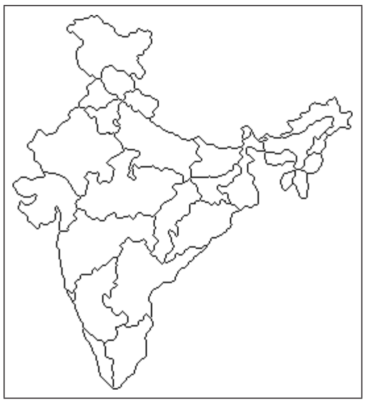
Approximate straightness. This is the major strength of the algorithm and marks its difference from the existing algorithms. The proposed algorithm utilizes the basic concepts of digital geometry, and outputs the polygon efficiently and successfully using low-level operations.

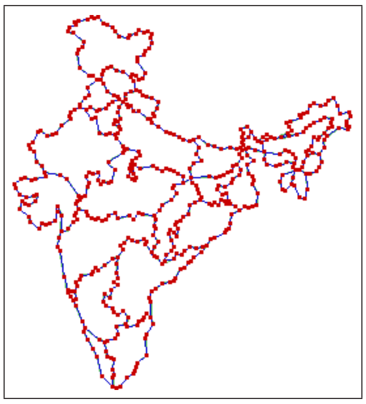
Primitive integer operations. As no floating-point operations are needed, the algorithms run very fast. Herein lies one of the major differences from the existing approaches.

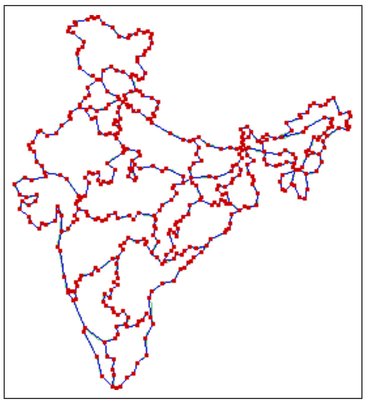
Convergence property. The algorithm converges to a quasi-static set of polygons for sufficiently high values of τ, as evident from the analysis and observations.

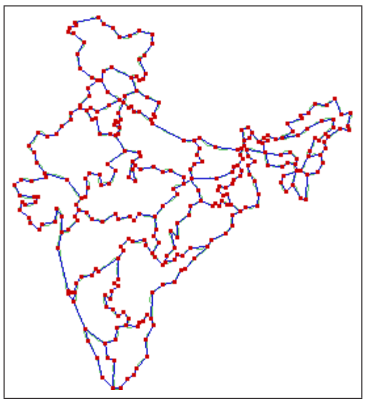
Robustness. The proposed algorithms are robust because of the fact that the basic properties of approximate straightness are inherited from those of the exact digital straightness.

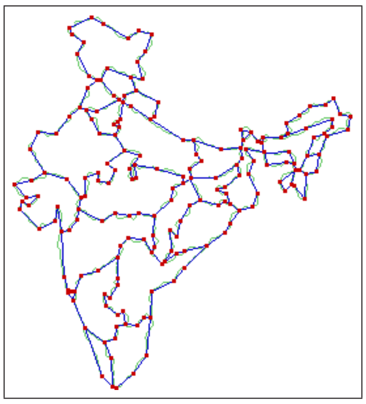
Minor dependence on optimality criterion. As the set of ADSS is generated first, the final stage of merging the collinear ADSS to a single edge of the approximate polygon becomes relatively simple, when either of the two approximation criteria proposed here is used.

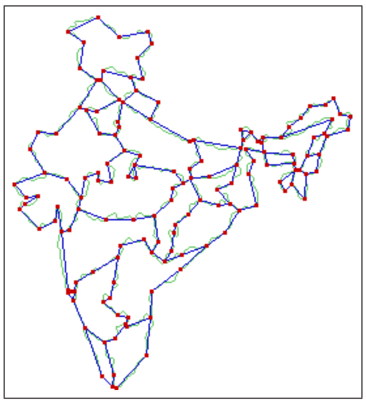

(a) “map of India” (**input**)

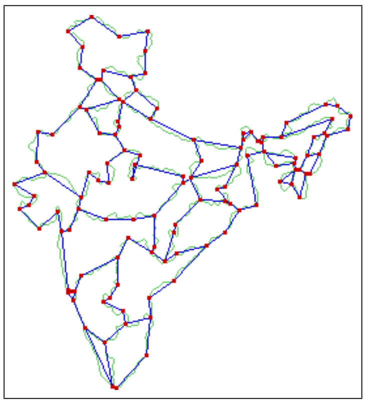

(b) ADSS

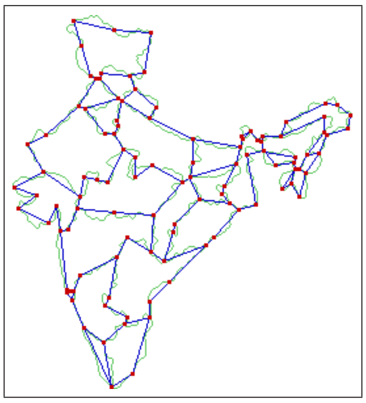

(c) $\tau = 1$


(d) $\tau = 2$


(e) $\tau = 4$


(f) $\tau = 6$


(g) $\tau = 8$


(h) $\tau = 10$

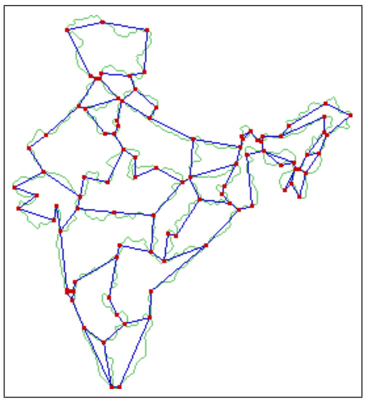

(i) $\tau = 12$

Figure 5.18: Polygonal approximations for “map of India” for different values of τ corresponding to criterion $C_{\sum}$.

In the future, optimizing the set of ADSS to cover a given DC would be a promising area of research, since the output set of the algorithm Extract-ADSS depends on the starting point and the direction of the traversal. A theoretical analysis of the error distribution in an ADSS extraction algorithm, as well as investigation of other underlying properties of ADSS, is another field for conducting further research.

Chapter 6

Digital Circles: Interpretation and Construction Using Number Theory

Why are numbers beautiful? It's like asking why is Beethoven's Ninth Symphony beautiful. If you don't see why, someone can't tell you. I know numbers are beautiful. If they aren't beautiful, nothing is.

Paul Erdös

6.1 Introduction

As mentioned in Chapter 1, theorization, interpretation, and rendition of the (digital) geometric primitives, especially those of straight lines and circles [Aken and Novak (1985), Foley *et al.* (1993), Klette and Rosenfeld (2004a)], mark a promising area of research in today's digital technology. The weird and challenging nature of circles and circular arcs in the digital domain have drawn immense interest since the early adoption of scan-conversion technique [Badler (1977), Bresenham (1977), Chung (1977), Danielsson (1978), Doros (1979), Horn (1976), Kulpa (1979), Pitteway (1974), Shimizu (1981)] for computing an efficient approximation of a (real) circle from the continuous domain to the digital domain. Subsequent improvements of these algorithms meant for generation of circular arcs were achieved by several researchers in the later periods, which may be seen in the literature [Blinn (1987), Bresenham (1985), Hsu *et al.* (1993), Mcllroy (1983), Suenaga *et al.* (1979), Wright (1990), Wu and Rokne (1987), Yao and Rokne (1995)]. Further, apart from the circle generation algorithms, other interesting studies on digital circles and related problems have also appeared from time to time, some of which are as follows:

(i) polygonal approximation of digital circles [Bhowmick and Bhattacharya (2005a), Hosur and Ma (1999)];

(ii) characterization of digital circles [Biswas and Chaudhuri (1985), Doros (1984), Haralick (1974), Nagy (2004), Worring and Smeulders (1995)];

(iii) detection/segmentation of circular arcs/objects in digital images [Chattopadhyay *et al.* (1994), Chen and Chung (2001a), Chiu and Liaw (2005), Coeurjolly *et al.* (2004), Davies (1987), Ho and Chen (1995), Kim and Kim (2001), Kulpa and Kruse (1983), Nakamura and Aizawa (1984), Pla (1996), Rosin and West (1988)];

(iv) parameterization of circular arcs [Chan and Thomas (1995), Davies (1988), Thomas and Chan (1989), Yuen and Feng (1996)];

(v) anti-aliasing solutions for digital circles [Field (1986)].

It may be mentioned that there exist various composite techniques [Blinn (1987), Hsu *et al.* (1993), Wright (1990), Wu and Rokne (1987), Yao and Rokne (1995)], designed to expedite and contend the procedure of Bresenham's Circle Drawing algorithm. However, these algorithms do not have considerably large gains over Bresenham's method. For instance, for radius from 1 to 128, the most efficient ones (see, for example, the algorithms and results by Hsu *et al.* (1993), and by Yao and Rokne (1995)) of these algorithms are 0.62 times to 1.53 times faster on the average with respect to Bresenham's algorithm. This is quite expected for the inherent brevity and simplicity of Bresenham's algorithm owing to the judicious usage of primitive arithmetic operations, which have been worked out sensibly and skillfully from the fundamentals of digital calculus.

However, the essence of all these algorithms is that, although a circle drawing algorithm primarily originates from the naive concept of intelligent digital applications of 1st order and 2nd order derivatives (differences) as in Bresenham's, a digital circle is endowed with some other interesting properties contributed by the classical theories in the discrete domain. It has been gradually made apparent from these studies that digital circles, similar to digital straight lines [Klette and Rosenfeld (2004b)], possess some striking characteristics, which, if interpreted rightly and exploited properly, may produce interesting results and scope for subsequent potential applications in the discrete domain.

This chapter reveals the relation between perfect squares (square numbers) in discrete (integer) intervals and few interesting and useful properties of a digital circle. Based on these number-theoretic properties, the problem of constructing a digital circle or a circular arc maps to the new domain of number theory. However, the construction of a digital circle using these properties should not be examined only in computational perspectives compared to Bresenham's algorithm or any other similar algorithm; rather, these number-theoretic properties enrich the understanding of digital circles from a perspective that is different from the customary aspects of digital calculus. It is also discussed in this chapter how these intervals can be obtained for the given radius of a circle, and what effects these intervals have on the construction of a digital circle.

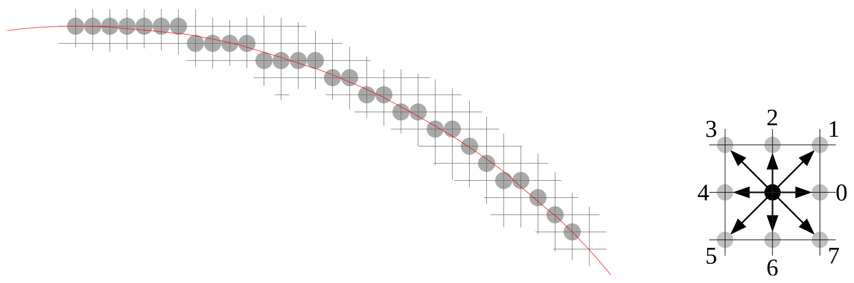


Figure 6.1: The digital circular arc $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, 41)$ for a real circle (shown in red) of radius 41. The chain codes of the eight neighbors of the grid point (i, j) are shown aside, with the convention of the right point $(i, j{+}1)$ as '0', followed by the other points in counterclockwise direction.

An instance of the digital circle for radius 41 is shown in Fig. 6.1. It may be seen that the corresponding *chain code* for this part of the circle, starting from the point $(0, 41)$, is given by $\mathtt{0^6 7 0^3 7 0^3 7 0 7 0 7 0 7 0 7^3 0 7^3}$, whence it is easy to observe that the runs of $\mathtt{0}$ "usually" decrease or continue to be same in length in the first octant as long as $\mathtt{7}$ appears as a singular character in the chain code. Alternatively, as shown in this chapter, the topmost run length (number of grid points with ordinate = 41) is given by the number of perfect squares in the (closed) interval $[0, 40]$, the next run length (number of grid points with ordinate = 40) given by that in $[41, 120]$, the next one (ordinate = 39) by that in $[121, 198]$, and so on. To elaborate, if $s[v, w]$ denotes the number of perfect squares in the closed interval $[v, w]$, then we get $s[0, 40] = 7$ (since 7 perfect squares, i.e., $0, 1, 4, \ldots, 36$, lie in $[0, 40]$), $s[41, 120] = 4$ (since $49, 64, 81, 100$ lie in $[41, 120]$), $s[121, 198] = 4$ (since $121, 144, 169, 196$ lie in $[121, 198]$), etc. Hence, the *square numeric code* of the above circle in the first octant is given by $\langle 7, 4, 4, 2, 2, 2, 2, 1, 1, 2, 1, 1, 1\rangle$, which can be shortened to $\langle 7, 4^2, 2^4, 1^2, 2, 1^3\rangle$, whence it is again apparent that the numbers of perfect squares are either non-increasing (predominant) or increasing by at most unity (occasional).

The brief outline of the chapter is as follows. In Sec. 6.2, we start with some fundamental properties of a digital circle, followed by the basic principle for its generation in

Sec. 6.2.1. In Sec. 6.2.2, we have explored few properties on the distribution of grid points for a digital circle based on the square numbers in discrete intervals, which may be used to design an algorithm on generation of the circle using few primitive integer operations only, given in Sec. 6.2.3, along with its analysis in Sec. 6.2.4 and comparison with Bresenham's algorithm in Sec. 6.2.5. Some other excellent properties have been presented in Sec. 6.3, which have been derived using the rudimentary number-theoretic concepts once again. These properties improve the former algorithm (Sec. 6.2.3) to actualize another algorithm, given in Sec. 6.3.1, using a modified binary search technique and without using any floating-point operation at any stage, which has been shown to be very effective for circles of higher radii, as evident from its analysis given in Sec. 6.3.2. Sec. 6.4 exhibits some test results that demonstrate the analytical strength and algorithmic applications of the novel approach proposed in this chapter. In Sec. 6.5, procedures for construction, segmentation, and characterization of digital circles/circular arcs by intelligent deployment of such number-theoretic properties, have been discussed.

6.2 Properties of Digital Circles

Let $\mathcal{C}^{\mathbb{Z}}(p,r)$ be the digital circle with center at $p \in \mathbb{Z}^2$ and radius $r \in \mathbb{Z}^+$. For simplicity of notations, $\mathcal{C}^{\mathbb{Z}}(p,r)$ is also used to denote the set of grid points/pixels constituting the digital circle with center at p and radius r in an appropriate context. Further, if $q \in \mathbb{Z}^2$ be such that $q \in$ (set) $\mathcal{C}^{\mathbb{Z}}(p,r)$, then q is said to be "lying on" (circle) $\mathcal{C}^{\mathbb{Z}}(p,r)$.

To start with, we observe a simple and very useful representation of a digital circle, which is required for construction of the digital circle about any grid point p as its center. The most convenient and customary representation of any digital curve (and a digital circle, thereof) is by means of Freeman's chain code [Freeman (1961a,b)], where the locally represented enumeration of the digital curve is mapped to the global coordinate system using a defined (global) point of reference, customized to specific requirements. For the digital circle $\mathcal{C}^{\mathbb{Z}}(p,r)$, if we consider the center p as the origin (point of reference) of the local coordinate system in $\mathbb{Z}^2$, then it can be shown that the set of grid points, enumerated in chain coded form w.r.t. p, representing the circle $\mathcal{C}^{\mathbb{Z}}(p,r)$, will be always independent of p and will be depending only on its radius r. Hence, we obtain the following lemma, which can be exploited to draw a digital circle $\mathcal{C}^{\mathbb{Z}}(p,r)$ centered at any point $p \in \mathbb{Z}^2$, provided $\mathcal{C}^{\mathbb{Z}}(\mathrm{O},r)$ is known, where, $\mathrm{O} = (0,0)$.

Lemma 6.2.1 *If $G(\mathcal{C}^{\mathbb{Z}}(p,r),q)$ represents the set of grid points lying on a digital circle $\mathcal{C}^{\mathbb{Z}}(p,r)$ with q as the point of reference for the representation, where, $p,q \in \mathbb{Z}^2$ and $r \in \mathbb{Z}^+$,* *then the sets* $G(\mathcal{C}^{\mathbb{Z}}(p, r), p)$ *and* $G(\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, r), \mathrm{O})$ *are identical.*

It may be noted that, the conversion of a digital circle $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, r)$ from the corresponding real circle $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, r)$ is done using the property of 8-axes symmetry of digital circles [Bresenham (1977), Foley *et al.* (1993)]. A sample digital circle $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, 11)$ along with its eight octants, and the corresponding real circle $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, 11)$ are shown in Fig. 6.2 for a ready reference. In order to obtain the complete circle $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, r)$, therefore, generation of the first octant, $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, r)$, (pixels or grid points in the closed interval $[0^0, 45^0]$, measured w.r.t. $+y$-axis in clockwise direction) of $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, r)$ suffices.

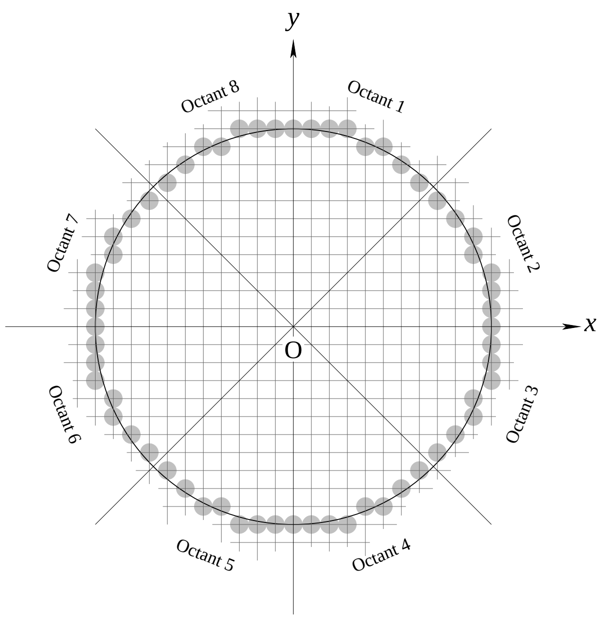


Figure 6.2: A real circle, $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, 11)$, and the eight octants of the corresponding digital circle, $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, 11)$.

6.2.1 Generation of a Digital Circle

While generating $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, r)$, decision is taken to select between east pixel $(\mathrm{E} := (i+1, j))$ or south-east pixel $(\mathrm{SE} := (i+1, j-1))$, standing at the current pixel $(\mathrm{P} := (i, j))$, depending on which one between E and SE is nearer to the point of intersection of the next ordinate line (i.e., $x = i + 1$) with the real circle $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, r)$. In the case of ties, any one between E and SE may be selected. It is, however, interesting to note that such a tie is never possible, and can apparently occur only if there is a computation error, the proof being as follows.

Let a tie occur when the real circle $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, r)$ has $(i, j+1/2)$ as the corresponding point of intersection in the first octant with the vertical grid line $x = i$. Since $(i, j+1/2)$ lies on $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, r)$, we have

$$\begin{aligned} & i^2 + (j+1/2)^2 = r^2, \\ \text{or,}\quad & r^2 - (i^2 + j^2 + j) = 1/4, \end{aligned}$$

which is impossible, since $r^2 \in \mathbb{Z}$, and $(i^2 + j^2 + j) \in \mathbb{Z}$. Hence we observe the following lemma.

Lemma 6.2.2 *A tie for selecting one of the two candidate pixels can never occur.*

Remark: Ideally, a tie as described above is impossible. Because of errors in floating-point computation, a tie may apparently occur.

6.2.2 Relation of Digital Circles with Square Numbers

Let $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, r)$ represent the first octant of $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, r)$ and let $\mathrm{P}(i, j)$ lie in $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, r)$. Hence the point of intersection Q of $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, r)$ with the vertical grid line $x = i$ should have distance less than $\frac{1}{2}$ from P, as shown in Fig. 6.3. Hence, if $(i, j-\delta)$ be the coordinates of Q, then we have $-\frac{1}{2} < \delta < \frac{1}{2}$, where it may be noted that $|\delta|$ can never be equal to $\frac{1}{2}$, as evident from Lemma 6.2.2.

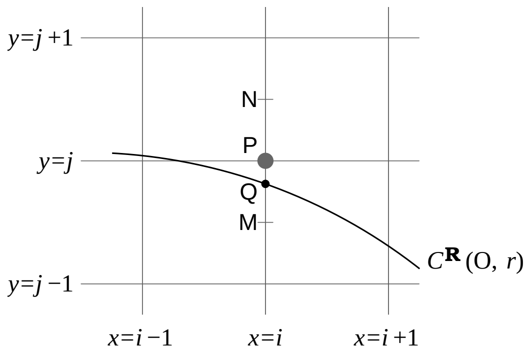


Figure 6.3: $\mathrm{P}(i, j)$ lies on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, r)$ provided the point of intersection Q of $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, r)$ with the vertical line $x = i$ lies between $\mathrm{M}(i, j-\frac{1}{2})$ and $\mathrm{N}(i, j+\frac{1}{2})$.

Now, since $\mathrm{Q}(i, j-\delta)$ lies on the real circle $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, r)$, we get

$$i^2 + (j-\delta)^2 = r^2,$$

$$
\begin{aligned}
\text{or,}\quad & \delta^2 - 2j\delta + (i^2 + j^2 - r^2) = 0,\\
\text{or,}\quad & \delta = \frac{1}{2}\left(2j \pm \sqrt{4j^2 - 4(i^2 + j^2 - r^2)}\right),\\
\text{or,}\quad & \delta = j - \sqrt{r^2 - i^2},\ \text{since}\ -\frac{1}{2} < \delta < \frac{1}{2}.
\end{aligned}
\tag{6.1}
$$

Therefore, extending Eqn. 6.1 and using the fact that $-\frac{1}{2} < \delta < \frac{1}{2}$, we get

$$
\begin{aligned}
& -\frac{1}{2} < j - \sqrt{r^2 - i^2} < \frac{1}{2},\\
\text{or,}\quad & -\frac{1}{2} - j < -\sqrt{r^2 - i^2} < \frac{1}{2} - j,\\
\text{or,}\quad & \frac{1}{2} + j > \sqrt{r^2 - i^2} > j - \frac{1}{2},\\
\text{or,}\quad & (j - \frac{1}{2})^2 < r^2 - i^2 < (j + \frac{1}{2})^2,\\
\text{or,}\quad & j^2 - j + \frac{1}{4} < r^2 - i^2 < j^2 + j + \frac{1}{4},\\
\text{or,}\quad & j^2 - j < r^2 - i^2 \leqslant j^2 + j,\ [\text{since}\ i, j, r \in \mathbb{Z}]\\
\text{or,}\quad & -j^2 - j \leqslant i^2 - r^2 < -j^2 + j,\\
\text{or,}\quad & r^2 - j^2 - j \leqslant i^2 < r^2 - j^2 + j.
\end{aligned}
\tag{6.2}
$$

Eqn. 6.2 reveals the pattern of the grid points that would represent the digital circle $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, r)$ in the first octant. Since the first grid point in the first octant is always $(0, r)$ (considering the clockwise enumeration), Eqn. 6.2 for the topmost grid points (i.e. $j = r$) becomes $0 \leqslant i^2 < r^2 - r^2 + r = r$, or, $0 \leqslant i^2 \leqslant r - 1$. This implies that, in the first octant, the grid points, having their ordinates as r, will have the squares of their abscissas in the (closed) interval $[0, r-1]$. Let us denote the interval $[0, r-1]$ by I_0 (zero-th interval).

Let I_1 be called as the first interval, which is obtained by substituting $j = r - 1$ in Eqn. 6.2. So, $I_1 = \left[r^2 - (r-1)^2 - (r-1), r^2 - (r-1)^2 + (r-1) - 1\right] = [r, 3r-3]$, which contains the squares of abscissas of all the grid points (in the first octant) whose ordinates are $r - 1$. Thus, in general, if I_k denotes the k-th ($k \geqslant 1$) interval, given by

$$
\begin{aligned}
I_k &= \left[r^2 - (r-k)^2 - (r-k), r^2 - (r-k)^2 + (r-k) - 1\right]\\
&= \left[(2k-1)r - k(k-1), (2k+1)r - k(k+1) - 1\right],
\end{aligned}
\tag{6.3}
$$

which is obtained by substituting $j = r - k$ in Eqn. 6.2, and then, proceeding in this way, for a digital circle with radius $r \geqslant 1$ and center at $(0, 0)$, we get the following lemma.

Lemma 6.2.3 *The interval* $I_k = [(2k-1)r - k(k-1), (2k+1)r - k(k+1) - 1]$ *contains the squares of abscissas of the grid points of* $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, r)$ *whose ordinates are* $r-k$*, for* $k \geqslant 1$.

The length l_k of the interval I_k ($k \geqslant 1$) is, therefore, given by

$$\begin{aligned} l_k &= ((2k+1)r - k(k+1) - 1)) - ((2k-1)r - k(k-1)) + 1 \\ &= 2r - 2k. \end{aligned} \tag{6.4}$$

Using Eqn. 6.4, the length l_{k+1} for interval I_{k+1} is given by $l_{k+1} = 2r - 2(k+1) = l_k - 2$, whence we have the following Lemma.

Lemma 6.2.4 *The lengths of the intervals containing the squares of equi-ordinate abscissas of the grid points in $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, r)$ decrease constantly by 2, starting from I_1.*

6.2.3 Algorithm DCS: Digital Circle Using Square Numbers

An algorithm for construction of digital circles can be designed, therefore, based on (numbers of) square numbers in the intervals
$I_0 = [0, r-1]$,
$I_1 = [r, 3r-3]$,
$I_2 = [3r-2, 5r-7]$,
$\ldots$,
$I_k = [(2k-1)r - k(k-1), (2k+1)r - k(k+1) - 1]$,
$\ldots$,
as shown in Sec. 6.2.2. The length of the first interval I_1 is greater than that of I_0 by $2(r-1)$, as given by Eqn. 6.4. More interestingly, for $k \geqslant 1$, the length of each interval I_{k+1} is lesser than that of I_k by 2, which is a constant. Hence, in the algorithm DCS, using square numbers, we search for the number of perfect squares in each interval $I_k, k \geqslant 0$, which, in turn, gives the number of grid points with ordinate $r-k$. The following theorem contains the above facts and findings in a concise way.

Theorem 6.2.5 *The squares of abscissas of grid points, lying on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, r)$ and having ordinate $= r-k$, lie in the interval $[u_k, v_k := u_k + l_k - 1]$, where u_k and l_k are given as follows.*

$$u_k = \begin{cases} u_{k-1} + l_{k-1} & \text{if } k \geqslant 1 \\ 0 & \text{if } k = 0 \end{cases} \tag{6.5}$$

$$l_k = \begin{cases} l_{k-1} - 2 & \text{if } k \geqslant 2 \\ 2r - 2 & \text{if } k = 1 \\ r & \text{if } k = 0 \end{cases} \tag{6.6}$$

PROOF. Follows easily from Lemma 6.2.3 and Lemma 6.2.4. □

Exerting Theorem 6.2.5, therefore, the algorithm DCS may be designed, as shown in Fig. 6.4. It may be noted that, since the $(i+1)$th square number $S_{i+1} = (i+1)^2$ can be obtained easily from the previous square number $S_i = i^2$, since $S_{i+1} = (i+1)^2 = S_i + 2i + 1$, which is equivalent to adding a "gnomon" [Shanks (1993)], as shown in Fig. 6.5 (courtesy, http://mathworld.wolfram.com). This is incorporated in the algorithm (steps 5–7), shown in Fig. 6.4, where the gnomon addition of $2i+1$ is realized by adding i with the previous square s, followed by adding an incremented value $(i++)$ of i, in order to optimize the primitive arithmetic operations. It is evident that, in Step 4, the procedure *include_8_sym_points* (i, j) includes the set of eight symmetric grid points, namely $\{(\pm x, \pm y) : \{x\} \cup \{y\} = \{i, j\}\}$, in $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, r)$.

```
Algorithm DCS (int $r$) {
1.  int $i = 0, j = r, s = 0, w = r - 1$;
2.  int $l = w << 1$;
3.  while $(j \geqslant i)$ {
4.          do { include_8_sym_points $(i, j)$;
5.                  $s = s + i$;
6.                  $i++$;
7.                  $s = s + i$; } while $(s \leqslant w)$;
8.          $w = w + l$;
9.          $l = l - 2$;
10.         $j--$; }}
```

Figure 6.4: Algorithm DCS.

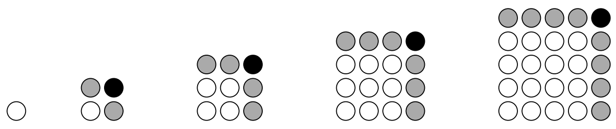


Figure 6.5: Generation of a square number S_{i+1} by adding a "gnomon" to S_i. All circles in the previous square number S_i are shown by unfilled circles in S_{i+1}, whereas the gray ones ($2i$ numbers) and the black one denote the gnomon added to S_i to get S_{i+1}.

It is worthwhile mentioning here that, although, from the two recurrence equations for u_k and l_k in Theorem 6.2.5, the run length $\lambda(r-k)$ (and the "square numeric code", thereof) at ordinate $r-k$ in the first octant can be obtained by

$$\lambda(r-k) = \left\lfloor \sqrt{u_k + l_k - 1} \right\rfloor - \left\lfloor \sqrt{u_k - 1} \right\rfloor , \tag{6.7}$$

evaluation of Eqn. 6.7 needs the square root operation, invoking unwanted floating-point computation, corresponding to each run. The algorithm DCS, given in Fig. 6.4, is a circumvention of Eqn. 6.7, using few primitive operations in the integer domain.

6.2.4 Analysis of Algorithm DCS

We discuss here about the number of elementary operations, namely comparison, addition (subtraction), and increment (decrement), required over all the iterations in the algorithm DCS. The inside **do–while** loop (steps 4–7) will find out the consecutive equi-ordinate (ordinate= j) grid points (to be precise, the squares of their abscissas) in $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O},r)$ at each iteration of the outside **while** loop (Step 3).

Now, each iteration of the outside **while** loop corresponds to a particular ordinate j, which gets decremented by unity (Step 10) for the next iteration corresponding to the next ordinate, i.e. $j-1$, for which, it may be observed, the upper limit w and the interval length l of the corresponding interval are being updated (Step 8 and Step 9) in accordance with Theorem 6.2.5. Hence, the total number of iterations of the outside **while** loop is given by the number of south-east (SE) transitions in the digital arc representing $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O},r)$. The total number of comparisons required for checking the condition ($j \geqslant i$) is, therefore, given by $|\mathrm{SE}|+1$, where, $|\mathrm{SE}|$ denotes the total number of SE transitions in $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O},r)$. The total number of comparisons for checking the condition ($s \leqslant w$) involved in the **do–while** loop, would be given by the total number of grid points in $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O},r)$, i.e. $|\mathrm{E}|+|\mathrm{SE}|$, where, $|\mathrm{E}|$ denotes the total number of east (E) transitions in $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O},r)$. Also, for each transition (whether E or SE), two additions (Step 5 and Step 7) and one increment (Step 6) are mandatory. In case of each SE transition, two extra additions (Step 8 and Step 9) and one extra increment (Step 10) are required. Hence, for the entire algorithm, the total numbers of comparisons (n_c), additions (n_a), and increments (n_i) are given by

$$\begin{array}{rcrcrcl} n_c &=& |\mathrm{E}| &+& 2|\mathrm{SE}| &+& 1 \\ n_a &=& 2|\mathrm{E}| &+& 4|\mathrm{SE}| && \\ n_i &=& |\mathrm{E}| &+& 2|\mathrm{SE}| && \end{array} \tag{6.8}$$

```
Algorithm Circle_Bresenham (int $r$) {
1.  int $i = 0, j = r, h = 1 - r, dE = 3, dSE = -2r + 5$;
2.  $include\_8\_sym\_points\ (i, j)$;
3.  while $(j > i)$ {
4.      if $(h < 0)$ {
5.          $h = h + dE$;
6.          $dE = dE + 2$;
7.          $dSE = dSE + 2$;
8.          $x++$; }
9.      else {
10.         $h = h + dSE$;
11.         $dE = dE + 2$;
12.         $dSE = dSE + 4$;
13.         $x++$; $y++$; }
14.     $include\_8\_sym\_points\ (i, j)$; }}
```

Figure 6.6: Bresenham's Algorithm.

6.2.5 Comparison with the Bresenham's Algorithm

Similar to the analysis given in Sec. 6.2.4, for the Bresenham's algorithm [Bresenham (1977)], given in Fig. 6.6, the total numbers of comparisons (nb_c), additions (nb_a), and increments (nb_i) would be given by

$$\begin{array}{rcl} nb_c &=& 2|\mathrm{E}| + 2|\mathrm{SE}| + 1 \\ nb_a &=& 3|\mathrm{E}| + 3|\mathrm{SE}| \\ nb_i &=& |\mathrm{E}| + 2|\mathrm{SE}| \end{array} \tag{6.9}$$

Hence, it is easy to observe that the algorithm DCS and Bresenham's are very much similar in their efficiency and ease in implementation. However, the algorithm DCS can be further endowed with some other (run length) properties of digital circles, which are discussed in the following Section.

6.3 Run Length Properties of Digital Circles

Lemma 6.3.1 *The number of perfect squares in a closed interval $[v, w]$ is at most one more than the number of perfect squares in the preceding closed interval $[u, v - 1]$ of equal*

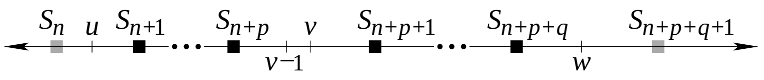


Figure 6.7: Square numbers on the number axis.

length, where the intervals are taken from the non-negative integer axis.

Proof. Let p and q denote the numbers of perfect squares in the closed intervals $[u, v-1]$ and $[v, w]$ respectively. Let $S_n = n^2$ be the largest square number lesser than u. Then, the square numbers lying in $[u, v-1]$ are $S_{n+1}, S_{n+2}, \ldots, S_{n+p}$, and those lying in $[v, w]$ are $S_{n+p+1}, S_{n+p+2}, \ldots, S_{n+p+q}$, as shown in Fig. 6.7. Now, using the relation $S_{m+1} = (m+1)^2 = S_m + 2m + 1$ and applying the method of induction, we get

$$S_{n+k} = S_n + 2kn + k^2. \tag{6.10}$$

Considering the fact that the difference between the highest and the lowest square numbers in $[v, w]$ is at most the length of the interval, that is $w - v + 1$ (see Fig. 6.7), we have

$$S_{n+p+q} - S_{n+p+1} \leqslant w - v, \tag{6.11}$$

where, it may be noted, the equality holds true when both v and w are perfect squares and, therefore, become equal to S_{n+p+1} and S_{n+p+q} respectively. Similarly, considering the square number (S_{n+p+1}) just greater than the largest square in $[u, v-1]$ and the square number (S_n) just lesser than the smallest square in $[u, v-1]$ (S_n would be equal to 0 for the degeneracy when $u = 0$), we get

$$S_{n+p+1} - S_n > v - u. \tag{6.12}$$

Hence, combining Eqn. 6.11 and Eqn. 6.12 with our consideration that $w-v = v-1-u$, we get

$$\begin{aligned}
& S_{n+p+q} - S_{n+p+1} \leqslant w - v < v - u < S_{n+p+1} - S_n, \\
\text{or,}\quad & S_{n+p+q} - S_{n+p+1} < S_{n+p+1} - S_n, \\
\text{or,}\quad & S_n + 2(p+q)n + (p+q)^2 - S_n - 2(p+1)n - (p+1)^2 \\
& < S_n + 2(p+1)n + (p+1)^2 - S_n, \\
\text{or,}\quad & 2(q-1)n + (2p+q+1)(q-1) < 2(p+1)n + (p+1)^2, \\
\text{or,}\quad & (q-1)(2n+2p+q+1) < (p+1)(2n+p+1).
\end{aligned} \tag{6.13}$$

Since the second factor in the left hand side of Eqn. 6.13 never falls short of the second factor in the right hand side, i.e., $2n + 2p + q + 1 \geqslant 2n + p + 1$ for all cases, including the degenerate cases when $p = 0$ or/and $q = 0$ or/and $n = 0$, validity of Eqn. 6.13 implies that the first factor in its left hand side is strictly lesser than the first term in its right hand side. Hence, $q - 1 < p + 1$, or, $q \leqslant p + 1$. $\square$

Theorem 6.3.2 *The run length of grid points of $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, r)$ with ordinate $j - 1$ never exceeds one more than the run length of its grid points with ordinate j.*

Proof. From Lemma 6.2.4, it is obvious that the length of the interval, containing the squares of abscissas (of grid points of $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, r)$) with ordinate $j - 1$, is 2 less than that corresponding to ordinate j. Even if the interval corresponding to ordinate $j - 1$ had been equal in length to the preceding interval corresponding to ordinate j, then from Lemma 6.3.1, the maximum number of grid points with ordinate $j - 1$ would not have exceeded one more than the number of grid points with ordinate j. Hence the proof. $\square$

It is interesting to note that Theorem 6.3.2 provides a good and useful upper bound on the number of grid points with ordinate j w.r.t. that corresponding to ordinate $j - 1$. The derivation of lower bound that we have obtained is, however, not so straightforward, as evident in the following statements.

If p and q denote the numbers of perfect squares in the closed intervals $[u, v - 1]$ and $[v, w]$ (of equal length, i.e., $v - 1 - u = w - v$) respectively, and $S_n = n^2$ be the largest square number lesser than u, as discussed earlier, then (see Fig. 6.7) it can be shown that[1]

$$\begin{aligned} & S_{n+p+q+1} - S_{n+p} > S_{n+p} - S_{n+1}, \\ \text{or,} \quad & 2n(p + q + 1) + (p + q + 1)^2 - 2np - p^2 > 2np + p^2 - 2n - 1, \\ \text{or,} \quad & 2n(q + 1) + 2p(q + 1) + (q + 1)^2 > 2n(p - 1) + p^2 - 1. \end{aligned} \tag{6.14}$$

Now, since our concern is to find a lower bound of q in terms of p, we consider $q + 1 \leqslant p - 1$, i.e., $2n(q + 1) \leqslant 2n(p - 1)$, whence Eqn. 6.14 produces the following relation.

$$\begin{aligned} & 2p(q + 1) + (q + 1)^2 > p^2 - 1, \\ \text{or,} \quad & (p + q + 1)^2 > 2p^2 - 1, \end{aligned}$$

[1] $S_{n+p+q+1} > w$ and $-S_{n+p} > -v$ implies $S_{n+p+q+1} - S_{n+p} > w - v$.
$S_{n+p} \leqslant v - 1$ and $-S_{n+1} \leqslant -u$ implies $S_{n+p} - S_{n+1} \leqslant v - 1 - u (= w - v)$.
Thus, $S_{n+p+q+1} - S_{n+p} > S_{n+p} - S_{n+1}$.

$$\begin{aligned}
\text{or,}\quad & (p+q+1)^2 \geqslant 2p^2,\\
\text{or,}\quad & q \geqslant (\sqrt{2}-1)p-1,\\
\text{or,}\quad & q \geqslant \left\lceil (\sqrt{2}-1)p-1 \right\rceil,\\
\text{or,}\quad & q \geqslant \left\lfloor (\sqrt{2}-1)p \right\rfloor, \qquad (6.15)\\
& \text{since } p \text{ and } q \text{ are integers and } (\sqrt{2}-1)p-1 \text{ is irrational.}
\end{aligned}$$

The bound for q in Eqn. 6.15 has two weaknesses. The first one is that we get a loose lower bound of q that would cause some redundant operations while deciding the run length of grid points at ordinate $j-1$ from that in the preceding ordinate j. Secondly, and more importantly, computation of the lower bound of q in accordance with Eqn. 6.15 requires a square root operation followed by a floor/truncation operation, which is computationally expensive and not desirable in the run length finding procedure of circle construction algorithm.

In order to circumvent the aforesaid problems, therefore, we turn around in a different way from Eqn. 6.14 to obtain a nicer bound for q as follows.

$$\begin{aligned}
& (q+1)(2n+2p+q+1) > (p-1)(2n+p+1),\\
\text{or,}\quad & (q+1)(p+q) > (p-q-2)(2n+p+1). \qquad (6.16)
\end{aligned}$$

It is easy to observe from Eqn. 6.16 that, if the interval $[u, v-1]$ containing p perfect squares be such that $p < 2n$, where S_n is the largest square number lesser than u, then $q \leqslant p+1$, or, $q < 2n+1$, which implies $p+q < p+2n+1$. Hence, from Eqn. 6.16, we get

$$\begin{aligned}
& q+1 > p-q-2,\\
\text{or,}\quad & 2q > p-3,\\
\text{or,}\quad & q > \frac{p-1}{2} - 1,\\
\text{or,}\quad & q \geqslant \frac{p-1}{2} \geqslant \left\lfloor \frac{p-1}{2} \right\rfloor. \qquad (6.17)
\end{aligned}$$

From the result obtained in Eqn. 6.17, we can, therefore, make the following observation.

Lemma 6.3.3 *The number of perfect squares in a closed interval $[v, w]$ is at least (floor of) half the number of perfect squares less one in the preceding closed interval $[u, v-1]$ of equal length, provided u is sufficiently high compared to the length of the interval $[u, v-1]$.*

Putting together the above findings, therefore, we get the following theorem that captures the run length properties of a digital circle in a precise form.

Theorem 6.3.4 *If $\lambda(j)$ be the run length of grid points of $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O},r)$ with ordinate j, then the run length of grid points with ordinate $j-1$ is given by*

$$\lambda(j-1) \geqslant \left\lfloor \frac{\lambda(j)-1}{2} \right\rfloor - 1.$$

Proof. From Lemma 6.2.4, the length of the interval containing the squares of abscissas of grid points with ordinate $j-1$ is 2 less than that corresponding to ordinate j. Had the interval corresponding to ordinate $j-1$ been equal in length to the preceding interval (ordinate j), then from Lemma 6.3.3, the number of grid points with ordinate $j-1$ would have been at least $\lfloor (\lambda(j)-1)/2 \rfloor$. In case the next integer or the next-to-next integer immediately after the interval corresponding to ordinate $j-1$ is a perfect square, the minimum possible value of $\lambda(j-1)$ would be further less by unity. □

6.3.1 Algorithm DCR

Combining Theorem 6.3.2 and Theorem 6.3.4, therefore, we obtain Eqn. 6.18, which can be used to derive the the horizontal run of grid points with ordinate $j-1$, from the previous run with ordinate j, for $j \leqslant r$.

$$\left\lfloor \frac{\lambda(j)-1}{2} \right\rfloor - 1 \leqslant \lambda(j-1) \leqslant \lambda(j)+1 \tag{6.18}$$

The algorithm that incorporates Eqn. 6.18 is shown in Fig. 6.8. It may be noted that, in Fig. 6.8, only the upper limit $(\lambda(j)+1)$ of the term $\lambda(j-1)$ has been considered for simplicity. The lower limit $(\lfloor \frac{1}{2}(\lambda(j)-1) \rfloor - 1)$ of $\lambda(j-1)$ may be considered in a similar fashion. In order to obtain the exact value of $\lambda(j-1)$, binary search has been used. The binary search assumes minimum value as 1 and maximum value as $\lambda(j)+1$ to start with for $\lambda(j-1)$. The binary search is performed on the Look-Up-Table, implemented in the form of a 1-dimensional array, namely *square*[ ], that contains the square S_n of each integer $n = 0, 1, 2, \ldots, N$, where N^2 is the largest square not exceeding the maximum value R of radius r. For example, for $R = 1000$, N (and the size of the Look-Up-Table, thereof) equals to 31.

It may be noted that the binary search procedure incorporated in Fig. 6.8 is a modified one, based on our requirements, from the conventional one found in the literature [Langsam *et al.* (2000)]. In accordance with the conventional procedure, one has to check at first whether the middle element (m in our figure) equals to the search key, failing which one between two other checks (smaller or greater) is performed. We have modified this (steps 7–10) to reduce the number of comparisons in each iteration of the inner **while** loop.

A demonstration of the algorithm DCR for $r = 106$ has been graphically shown in Fig. 6.9 till a run of unit length is found. For each row, the binary search is illustrated by circular dots, where each dot corresponds to the middle element (m) of the respective sub-array ($square[s..t]$). It may be noticed that, m in Fig. 6.8 (Step 6) denotes the abscissa (vertical) line $x = m - 1$ in Fig. 6.9. As the binary search, associated with a particular row, proceeds and converges to produce the final run of the corresponding row, the respective dots have been gradually darkened to depict the impact of the run length finding procedure. The end of the run at each row is emphasized by highlighted abscissa lines passing through the end point of that run for visual clarity. For instance, for the topmost row, m starts with 53, followed by 26, 13, and so on, until $s = t = 11$, whence m finally becomes 11, thereby yielding the run length equal to 11. Similarly, for the next row, since the start values of s and t are 11 and 23 respectively, the subsequent values of m are 17, 20, 19, and (finally) 18, which makes the corresponding run length to $18 - 11 = 7$. Thus, the run length of a particular row ($y = j$) is given by the cumulative run length up to that row minus the preceding cumulative run length, which is realized by the function $include\_run$ $(x, s - x, j)$ in Step 6 of the algorithm. The "square numeric code" of a circle with radius 106 in the first octant, therefore, turns out to be $\langle 11, 7, 5, 5, 3, 3, 3, 3, 2, 2, 2, 3, 1, 2, 2, 2, 1, 2, 1, 2, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1\rangle$, which can be compressed to $\langle 11, 7, 5^2, 3^4, 2^3, 3, 1, 2^3, 1, 2, 1, 2, 1^2, 2, 1^3, 2, 1^5\rangle$, which, in turn, can be used to easily derive the chain code representation of the circular arc, as mentioned in Sec. 8.1 for Fig. 6.1.

It is worth mentioning here that, implementation of the Look-Up-Table in the form, mentioned above, avoids floating-point operations of any sort at any stage, such as square root operation to find the run for $j = r$ in the algorithm proposed by Yao and Rokne (1995). Also, the above algorithm will not be so efficient for circles (1st octant) of small radii, or to find out a run length of small value, such as when $y/x \leqslant 4$ or so (see Sec. 6.4) since the operations (comparisons etc.) needed in the binary search for small run lengths would raise the overall time of the algorithm. Hence, the algorithm DCR may be used in some model of hybridization, whose realization, in some form, may be seen in some existing studies, e.g. Yao and Rokne (1995).

```
**Algorithm** DCR (int $r$) {
1.  int $i = 0, j = r, s = 0, w = r - 1, t = r, x = 0, m$;
2.  int $l = w << 1$;
3.  **while** $(j \geqslant i)$ {
4.          **while** $(s < t)$ {
5.                  $m = s + t$;
6.                  $m = m >> 1$;
7.                  **if** $(w \leqslant square[m])$
8.                          $t = m$;
9.                  **else**
10.                         $s = m + 1$; }
11.         **if** $(w < square[s])$
12.                 $s--$;
13.         $s++$;
14.         $include\_run\ (x, s - x, j)$;
15.         $t = s + s - x + 1$;
16.         $x = s$;
17.         $w = w + l$;
18.         $l = l - 2$;
19.         $j--$; }}
```

Figure 6.8: Algorithm DCR using run length properties in part (see text for explanation).

6.3.2 Analysis of Algorithm DCR

We would analyze the algorithm DCR for showing its readiness in hybridizing a circle construction algorithm, e.g. DCS given in Fig. 6.4. We assume that some of the top runs (i.e. $j = r, r-1, r-2, \ldots$) are generated by the algorithm DCR, followed by the generation of the remaining runs by a simpler algorithm like DCS. The problem is, therefore, for generation of how many runs (starting from the topmost one) the algorithm DCR should be used, so that the overall time of the resultant hybrid algorithm is minimized.

The analysis of the algorithm DCS, as put in Sec. 6.2.4, shows that the total number of operations (comparisons, additions, and increments) for generation of $k+1$ runs is given by $4|\mathrm{E}|+8|\mathrm{SE}|+1$, from which, by dropping '+1', the lower bound is given by $4|\mathrm{E}|+8|\mathrm{SE}|$.

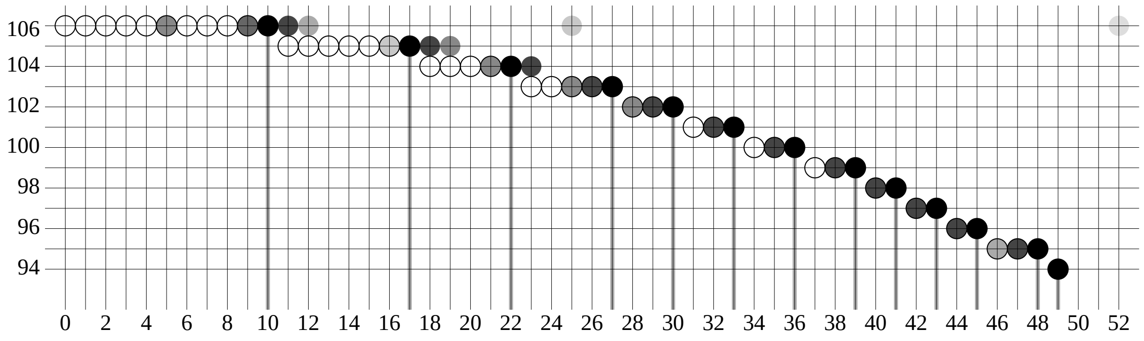


Figure 6.9: Demonstration of algorithm DCR for radius 106 (see text for explanation).

Now, observing that |SE| is the number of south-east transitions, which is simply k, and |E| is the number of east transitions, given by

$$\begin{aligned} |\mathrm{E}| &= \left\lfloor \sqrt{kr + r(k+1) - k(k+1) - 1} \right\rfloor - k \quad \text{[from Eqn. 6.3]} \\ &> \left\lfloor \sqrt{kr} \right\rfloor - k \quad \text{[since } r > k\text{]}. \end{aligned}$$

For producing the top $k+1$ runs (i.e., for $j = r, j = r-1, \ldots, j = r-k$) by the algorithm DCS, therefore, the total number of primitive operations is given by

$$n_1^{(k+1)} > 4\left(\left\lfloor \sqrt{kr} \right\rfloor - k\right) + 8k = 4\left\lfloor \sqrt{kr} \right\rfloor + 4k. \tag{6.19}$$

For the algorithm DCR, in accordance with the simple implementation, as shown in Fig. 6.8, for the topmost row (i.e., $j = r$), the number of iterations of the inner **while** loop (Step 4), meant for binary search as discussed in Sec. 6.3.1, is given by $\lceil \log r \rceil$, the base of $\log(\cdot)$ being 2 in this discussion. For the binary search on finding out each run length at $j < r$, the number of iterations depends on the length of the search interval (i.e., initial values of s and t for the corresponding j, see Fig. 6.8), which depends on the preceding run length due to the inherent recursive nature of Eqn. 6.18 used in the algorithm. This poses a problem in deriving the number of primitive operations in the algorithm for the individual runs. However, it is easy to observe that for each subsequent value of $j (< r)$, the inner **while** loop iterates for not more than $\lceil \log(\sqrt{r}) \rceil = \left\lceil \frac{1}{2} \log r \right\rceil$ times, since the run length for $j \leqslant r-1$ would be always lesser than that for $j = r$ (which is $\lfloor \sqrt{r} \rfloor + 1$); in fact, for $j \leqslant r-1$, the value $\lfloor \sqrt{r} \rfloor$ turns out to be a loose upper bound for its run length, where the actual run length for $j = r - k (k > 0)$ falls shorter and shorter of $\lfloor \sqrt{r} \rfloor$ as k goes higher and higher.

Now, for each iteration of the inner **while** loop, including the check ($s < t$) in Step 4, and assuming that the logical expression ($w \leqslant square[m]$) in Step 7 is always false (worst case), thereby requiring one addition in Step 10 at each iteration, the total number of operations (comparisons, additions, right shifts, and increments) per iteration turns out to be 5 in the worst case.

Similarly, for each iteration of the outer **while** loop, the total number of all primitive operations can be shown to be at most 11. Further, for the outer **while** loop, the total number of iterations is simply the number of SE transitions, which becomes k, if $k+1$ runs are produced by the algorithm DCR.

Thus, for producing $k+1$ runs, the total number of primitive operations (worst case) required by the algorithm DCR is given by

$$n_2^{(k+1)} < 5\left(\lceil k/2 \rceil + 1\right)\lceil \log r \rceil + 11k. \tag{6.20}$$

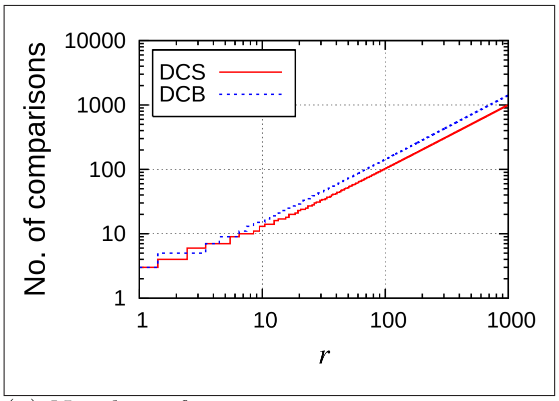
Hence, from Eqns. 6.19 and 6.20, the algorithm DCR would be faster than DCS for generating the top $k+1$ runs, provided

$$\begin{aligned} & 5\left(\lceil k/2 \rceil + 1\right)\lceil \log r \rceil + 11k < 4\left\lfloor \sqrt{kr} \right\rfloor + 4k, \\ \text{or,}\quad & 5\left(\lceil k/2 \rceil + 1\right)\lceil \log r \rceil + 7k < 4\left\lfloor \sqrt{kr} \right\rfloor. \end{aligned} \tag{6.21}$$

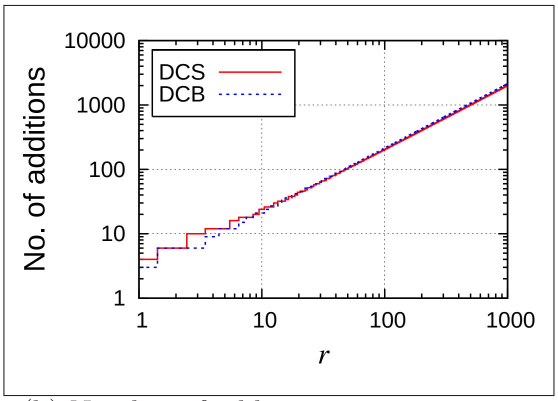
It may be noted that, while deriving the lower bound for $n_1^{(k+1)}$ and upper bound for $n_2^{(k+1)}$ in Eqns. 6.19 and 6.20 respectively, we have used some loosely-framed inequalities. The relation shown in Eqn. 6.21, therefore, does not impose a tight restriction on the number of runs (i.e., $k+1$) that should be produced by the algorithm DCR. However, from Eqn. 6.21, we get, for a given value of radius r, the least number of runs that should be generated by DCR, followed by the remaining runs by the algorithm DCS (or Bresenham's algorithm) for an efficient hybridization.

6.4 Experimental Results

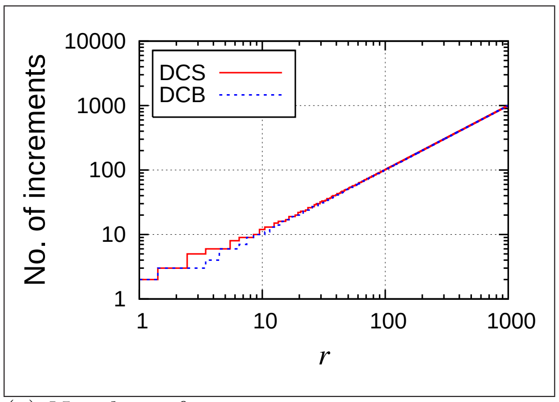
Implementation of the two algorithms DCS and DCR, whose pseudocodes are given in Fig. 6.4 and Fig. 6.8, is simple as that of Bresenham's. As mentioned in Sec. 8.1, since construction of a digital circle using the number-theoretic properties, discussed in this chapter, should not be considered as to contend Bresenham's algorithm, but as to supplement it in theoretical perspectives, we have not produced here results on (CPU) run times; rather, few results on the number of primitive operations required for the algorithms are



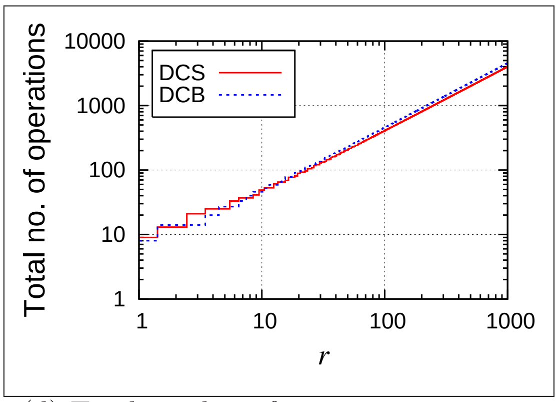
(a) Number of comparisons.



(b) Number of additions.



(c) Number of increments.

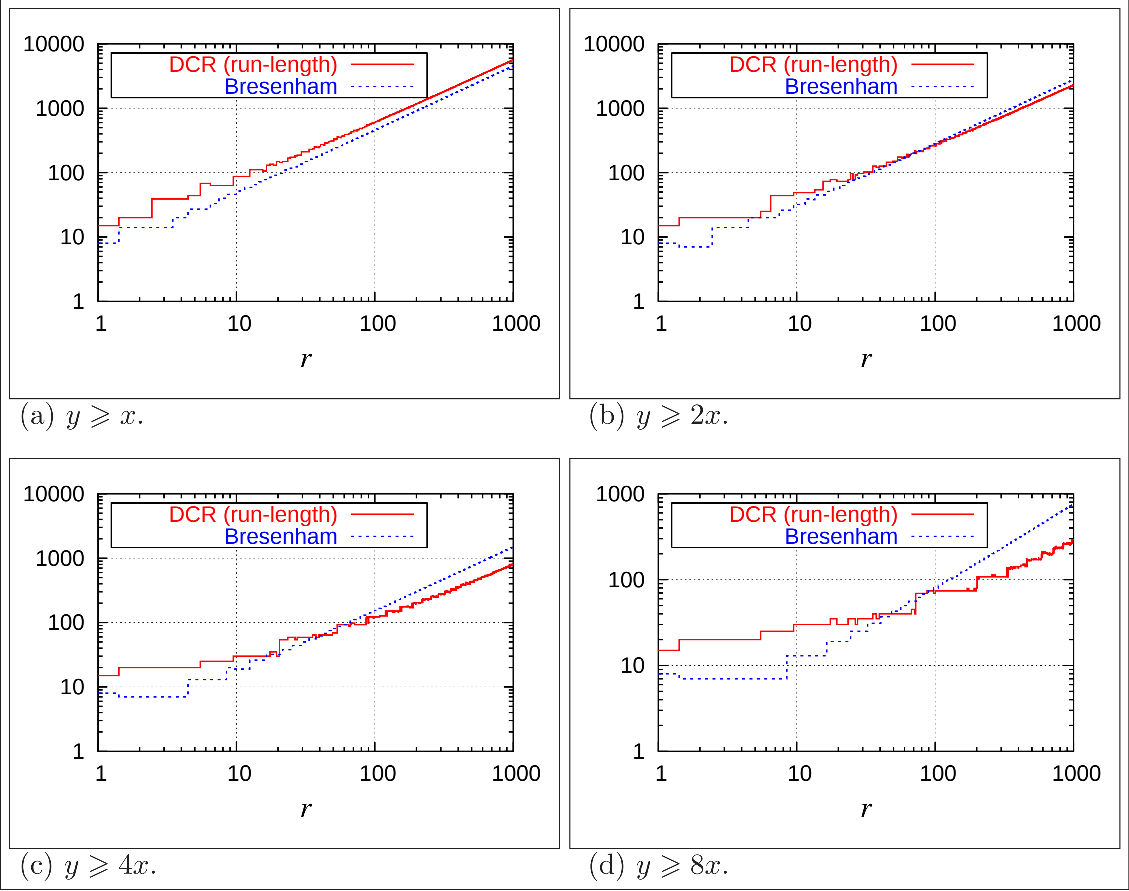


(d) Total number of operations.

Figure 6.10: Primitive operations in DCS algorithm and Bresenham's algorithm for radius r from 1 to 1000. The plots have been given in $\log_{10}$ scale.

presented below. It may be noted that, for a given radius r, the number of primitive operations to generate (first octant of) the corresponding digital circle is always fixed (i.e., deterministic) for any one of the algorithms on circle construction. Furthermore, since the CPU times are dependent on the machine configuration and code optimization by the associated compiler, CPU times not necessarily reflect the true picture of the speed and efficiency of an algorithm.

In Fig. 6.10, the major primitive operations, namely comparisons, additions, and increments, have been shown for the algorithm DCS in $\log_{10}$ scale for radius from 1 to 1000. It is evident from these plots, which are just experimental authetications of the theoretical values of the number of operations, given in Eqn. 6.8, that the number-theoretic algorithm DCS is no less efficient and no less dependable than Bresenham's algorithm.



(a) $y \geqslant x$. (b) $y \geqslant 2x$. (c) $y \geqslant 4x$. (d) $y \geqslant 8x$.

Figure 6.11: Total number of operations (given in $\log_{10}$ scale) in DCR algorithm (given in Fig. 6.8) and Bresenham's algorithm for different starting parts ($y \geqslant x$, $y \geqslant 2x$, …) of 1st octant.

In Fig. 6.11, the total number of primitive operations that includes comparisons, additions, right shifts, and increments, have been shown for the algorithm DCR. The plots in this figure are for the generation of some initial parts (till $y \geqslant tx, t = 1, 2, 4, 16$) of the first octant, starting from the topmost row (i.e., $j = r$). As discussed in Sec. 6.3 and Sec. 6.3.2, proper hybridization techniques may be designed that would first generate the runs (using a binary search technique, as discussed earlier, and an example illustrated in Fig. 6.9), starting from the topmost row, and then use DCS (or Bresenham's algorithm, or any similar algorithm) for remaining part of the first octant. The top runs would be generated by a number-theoretic algorithm, such as the algorithm DCR, as long as the run-generation procedure is appreciably faster than the other, as explained in Sec. 6.3.2.

The trade-off in the hybridization is a critical issue, which has been portrayed in the plots in Fig. 6.11. From these plots, it is apparent that for radius exceeding 100 or so, the number-theoretic algorithm (DCR) has appreciable margin over Bresenham's algorithm. Further, as evident from the plots, the gain for the run length finding approach using the number-theoretic technique goes on increasing as the radius increases. For very large radius, say that exceeding 1000, the number-theoretic technique would contribute substantial improvements to any circle generation procedure.

6.5 Conclusion

We have shown how the number-theoretic method can be used to identify some fundamental and interesting properties of a digital circle, which are of immense significance to its mathematical interpretation and its subsequent graphical construction. These number-theoretic properties not only conform to, but also supplement and enrich further, the properties originated from the conventional concepts of digital calculus, such as those used in Bresenham's circle construction algorithm.

Although we have discussed in this chapter about the scope and merits of designing circle construction algorithms (regular as well as hybridized) using number-theoretic properties, further improvement of these algorithms still remains an open problem. For instance, to explore the possibility of finding all the run lengths in the first octant for a given radius r, without using the information of the other (preceding) run length(s), is an engrossing problem that may be worked upon, which, if solved, would bring down the computation time, and also might lead to an efficient parallelization of the circle construction algorithm.

Further, although there exist several studies at present on segmentation and characterization of digital circles and circular arcs using geometric properties, as indicated in Sec. 8.1, an intricate problem lies in discovering some advanced number-theoretic properties that may aid and expedite the process of segmentation/characterization of circular arcs in a digital image. For instance, given the chain code representation (the "square numeric code", thereof) of a digital curve, what policy should be adopted in order to decide whether or not the corresponding curve is an arc of a digital circle. To design an efficient algorithmic approach for this, using the primitive operations in the integer domain only, several other necessary (and desirably, sufficient) number-theoretic properties have to be discovered, apart from those presented in this chapter. It has been already made apparent in studies on processing and applications of digital images (some of which have been mentioned in Sec. 8.1) that segmentation of circular arcs in a digital image is really a captivating and intriguing problem with numerous applications in digital geometry.

Chapter 7

Digital Circles: Approximation by Real Polygons

By natural selection our mind has adapted itself to the conditions of the external world. It has adopted the geometry most advantageous to the species or, in other words, the most convenient. Geometry is not true, it is advantageous.

Jules Henri Poincaré

Science and Method

7.1 Introduction

As discussed in Chapter 6, the construction, properties, and characterization of digital circles constitute a very interesting area of research, and there exist several studies on digital circles and related problems — originating from different theoretical perspectives and depending on their application areas in the digital domain. Described in this chapter is a novel work that explores the subtleties and various possibilities on the problem of approximating a digital circle by a regular (convex) polygon.

It may be emphasized that the approximation of a given digital circle by a suitable regular polygon has significant applications to *approximate* matching of sets of points on two-dimensional plane [Chapters 1, 2, and 3]. For instance, in fingerprint matching, given a set A of grid points, which are the feature points called "minutiae" in a fingerprint image [Chapter 3], the task of finding those points (minutiae) in A, which would be lying on or inside a given digital circle, becomes tedious and time-consuming. The process (circular range query [Chapter 2]) becomes faster and efficient if we can find a suitable regular polygon in $\mathbb{R}^2$ (meant for polygonal range query) corresponding to the given digital circle. The *ideal* regular polygon should be such that it should not produce a matching result different from the result obtained with the digital circle, which is possible if and only if there exists no grid point on or inside the digital circle that lies outside the polygon, and also, there exists no grid point outside the digital circle that lies on or inside the polygon. In other words, all the grid points lying on and inside the digital circle should lie on and inside the (ideal) polygon, and vice versa.

It may be noted that, with the increase in the number of vertices of the regular polygon approximating the circle in the discrete domain, the final matching results go on improving but the space and time complexity goes on worsening owing to the rise in complexity of the allied algorithms and increase in dimensionality of the supporting data structures [Berg *et al.* (2000)]. With lesser size (i.e., fewer number of vertices) of the approximating regular polygon (and with a bit of compromise on the acceptable error part, thereof), the query and the subsequent matching process become faster with desired near-exact results.

In this chapter, we show that an ideal regular polygon corresponding to a digital circle is analytically possible for some of the digital circles, especially for the ones having lower radii. For larger digital circles, ideal regular polygons are rarely possible. However, approximate polygons, tending to ideal ones, are possible, which would have very low error rate. We have reported the conditions under which an ideal regular polygon definitely exists corresponding to a digital circle, and also the conditions under which the existence of an ideal regular polygon becomes uncertain. Furthermore, when the complexity (i.e., number of vertices or number of edges) of the ideal regular polygon is high, or when an ideal regular polygon does not exist, a regular polygon with smaller number of vertices can be a useful approximation of the ideal regular polygon. Results on testing different parameters of characterization of digital circles have been given to indicate the possibilities of approximation and the subsequent errors, thereof.

7.2 Approximation of a Digital Circle to a Real Polygon

A disc $\mathcal{D}^{\mathbb{R}}(q,r)$ lying in the real plane $\mathbb{R}^2$, $q \in \mathbb{R}^2$ and $r \in \mathbb{R}^+$ being its center and radius respectively, can be conceived in the discrete domain $\mathbb{Z}^2$ by the corresponding digital disc, namely $\mathcal{D}^{\mathbb{Z}}(\mathcal{D}^{\mathbb{R}}(q,r))$, in either of the following three possible ways (Fig. 7.1):

$$\mathcal{D}^{\mathbb{Z}}(\mathcal{D}^{\mathbb{R}}(q,r)) = \begin{cases} \mathcal{D}^{\mathbb{Z}}(\alpha,\rho) = \{p \in \mathbb{Z}^2 : round(d(p,\alpha)) \leq \rho\} & \text{for Type 1} \\ \mathcal{D}^{\mathbb{Z}}(q,\rho) = \{p \in \mathbb{Z}^2 : round(d(p,q)) \leq \rho\} & \text{for Type 2} \\ \mathcal{D}^{\mathbb{Z}}(q,r) = \{p \in \mathbb{Z}^2 : d(p,q) \leq r\} & \text{for Type 3} \end{cases}$$

Here, $d(p,q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$ with $p = (x_p, y_p)$ etc., α is the nearest grid point in $\mathbb{Z}^2$ corresponding to q in $\mathbb{R}^2$, and ρ is the nearest integer corresponding to r; i.e., if $\alpha = (x_\alpha, y_\alpha)$, then $x_\alpha = round(x_q)$, $y_\alpha = round(y_q)$, and $\rho = round(r)$, considering

$round(x) = \lfloor x + 0.5 \rfloor$ for $x \in \mathbb{R}$. It is easy to observe that the grid points defining the boundary of a digital disc (of any type) constitute the corresponding digital circle. This is evident from the digital discs and their respective digital circles shown in Fig. 7.1.

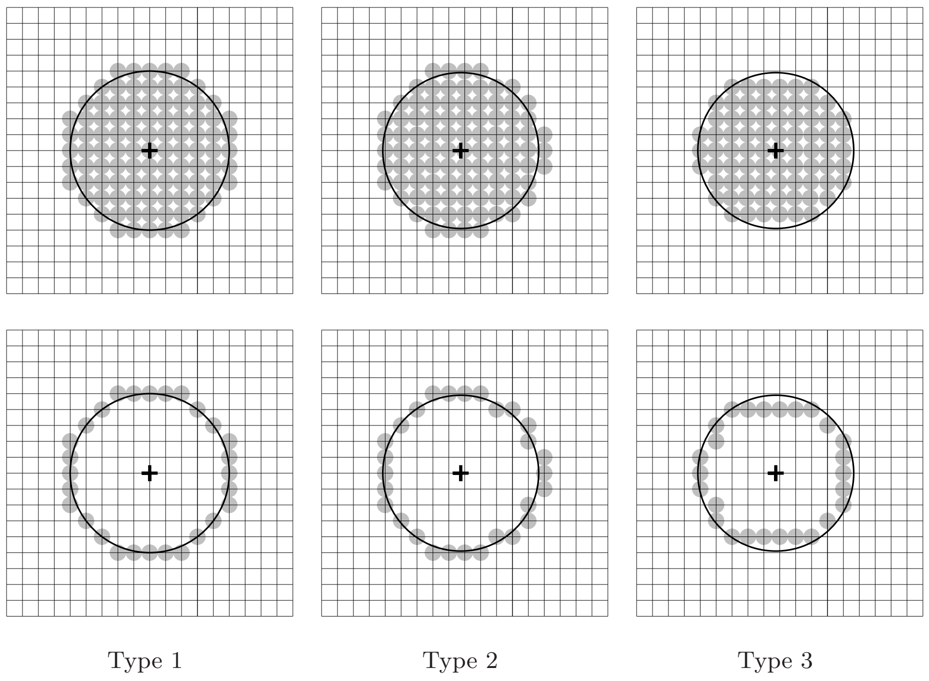


Figure 7.1: Three types of digital discs (top row) and the corresponding digital circles (bottom row) obtained by three different procedures on discretization of a real disc/circle with center $q = (-0.25, 0.00)$ and radius $r = 4.90$. See text for details.

It is known that [Hosur and Ma (1999), Worring and Smeulders (1995)], when the real disc $\mathcal{D}^{\mathbb{R}}(q, r)$ is discretized without *round*ing q and r as mentioned above, the result (digital disc of Type 2) is not necessarily same as that for Type 1 digital disc. In another procedure to discretize a real disc [Worring and Smeulders (1995)], the digital disc is defined to have the same center (q) and the same radius (r) as that of the given real disc, such that, finally each grid point, defining this digital disc (Type 3), has distance not exceeding r as measured from q. Thus, contrary to Type 1, the center or/and radius (of the digital discs) are not necessarily integers in Type 2 and Type 3. Since for an APSPM involving the sets of points in the digital plane [Chapter 2], the coordinates of the center and radius of the digital disc for circular range query are integers, we have considered only Type 1 digital discs in this work. Nevertheless, Type 2 and Type 3 discs may be investigated for exploring other possible applications to circular range query in the APSPM problem.

In this chapter, $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$ is used to denote the digital circle with center at α and radius ρ. For simplicity of notations, $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$ is also used to denote the set of grid points (pixels) constituting the digital circle with center at α and radius ρ in an appropriate context. Further, if $\nu \in \mathbb{Z}^2$ be such that $\nu \in$ (set) $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$, then ν is said to be "lying on" (circle) $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$. Now, since the points lying on $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$ are considered to be connected in 8-neighborhood, $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$ gives a closed digital arc that partitions the set of background points $(\mathbb{Z}^2 \smallsetminus \mathcal{C}^{\mathbb{Z}}(\alpha,\rho))$ into two disconnected components (in 4-neighborhood), whose locations are easily understood — one inside $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$, and the other outside $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$, which are denoted by the sets $\overline{\mathcal{C}}^{\mathbb{Z}}_{in}(\alpha,\rho)$ and $\overline{\mathcal{C}}^{\mathbb{Z}}_{out}(\alpha,\rho)$, respectively. If $\nu \in \overline{\mathcal{C}}^{\mathbb{Z}}_{in}(\alpha,\rho)$, then ν is said to be "lying inside" $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$, and if $\nu \in \overline{\mathcal{C}}^{\mathbb{Z}}_{out}(\alpha,\rho)$, then ν is said to be "lying outside" $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$. Similarly, any real point $p \in \mathbb{R}^2$ is said to lie inside, or lie on, or lie outside the real circle $\mathcal{C}^{\mathbb{R}}(q,r)$, the regions being denoted by $\overline{\mathcal{C}}^{\mathbb{R}}_{in}(q,r)$, $\mathcal{C}^{\mathbb{R}}(q,r)$, and $\overline{\mathcal{C}}^{\mathbb{R}}_{out}(q,r)$ respectively, depending on whether the line segment joining p and q is less than, or equal to, or greater than r, respectively.

For construction of the digital circle $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$ about any grid point α as its center, we can use $\mathcal{C}^{\mathbb{Z}}(\mathrm{O},\rho)$ and Freeman's chain code [Freeman (1961a,b)], as shown in Chapter 6. Using Lemma 6.2.1 stated in Chapter 6, we can draw the digital circle $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$ centered at any point $\alpha \in \mathbb{Z}^2$, provided $\mathcal{C}^{\mathbb{Z}}(\mathrm{O},\rho)$ is known, where $\mathrm{O} = (0,0)$. Further, the conversion of a digital circle $\mathcal{C}^{\mathbb{Z}}(\mathrm{O},\rho)$ from the corresponding real circle $\mathcal{C}^{\mathbb{R}}(\mathrm{O},\rho)$ is done using the property of 8-axes symmetry of digital circles, as explained in Chapter 6. The eight octants of a digital circle $\mathcal{C}^{\mathbb{Z}}(\mathrm{O},6)$ are shown in Fig. 7.2.

7.2.1 Definitions and Preliminaries

As explained in Chapter 6, for generating $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O},\rho)$, decision is taken to select the nearer pixel between the east pixel (E: $(i+1,j)$) and the south-east pixel (SE: $(i+1,j-1)$) w.r.t. the current pixel (i,j). It is also important to note that, there does not arise any case of ties (between E and SE), for ρ being an integer, as shown in Lemma 6.2.2 of Chapter 6.

To obtain the regular polygonal enclosure of $\mathcal{C}^{\mathbb{Z}}(\mathrm{O},\rho)$, the following definitions are now stated, whose underlying significance is detailed in Fig. 7.3.

Definition 7.2.1 A point (x,y) (in $\mathbb{Z}^2$ or in $\mathbb{R}^2$, as the case may be) is said to be *lying in the first octant* (with respect to O, unless stated otherwise) if and only if $0 \leqslant x \leqslant y$.

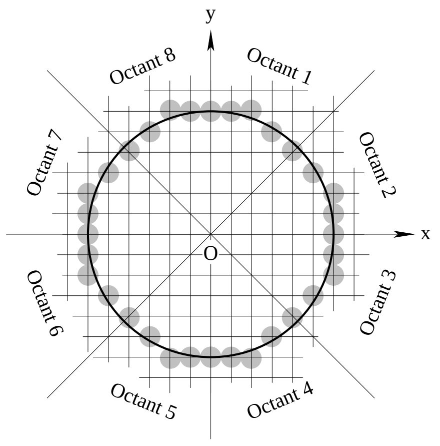


Figure 7.2: A real circle, $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, 6)$, and the eight octants of the corresponding digital circle (Type 1), $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, 6)$.

Definition 7.2.2 If a grid point (i, j) lies in the first octant and outside $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$ (Fig. 7.3), then its *x-distance* (also called *horizontal distance*) from $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$ is given by

$$d_x = \begin{cases} i - x_j, & \text{if } j \leqslant \rho, \text{ where the horizontal grid line through } (i,j) \text{ intersects} \\ & \mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho) \text{ in the first quadrant at } (x_j, j); \\ \infty, & \text{otherwise.} \end{cases}$$

Definition 7.2.3 If a grid point (i, j) lies in the first octant and outside $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$, then its *$y$-distance* (also called *vertical distance*) from $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$ is given by

$$d_y = \begin{cases} j - y_i, & \text{if } i \leqslant \rho, \text{ where the vertical grid line through } (i,j) \text{ intersects} \\ & \mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho) \text{ in the first quadrant at } (i, y_i); \\ \infty, & \text{otherwise.} \end{cases}$$

Definition 7.2.4 If a grid point (i, j) lies in the first octant and outside $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$, then its *$xy$-distance* (also called *isothetic distance*) from $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$ is given by

$$d_\perp = \begin{cases} d_x, & \text{if } d_x \neq \infty, \text{ and } d_x < d_y; \\ d_y, & \text{if } d_y \neq \infty, \text{ and } d_y \leqslant d_x; \\ \infty, & \text{otherwise.} \end{cases}$$

Definition 7.2.5 If a grid point (i, j) lies outside $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$, then its *radial distance* from $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$ is given by $d_r = \sqrt{i^2 + j^2} - \rho$.

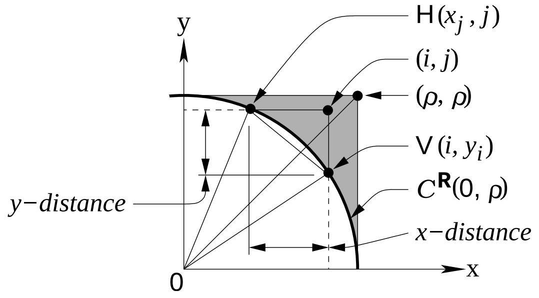


Figure 7.3: *x-distance* and *y-distance* of a grid point (i, j) from $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$, where, (i, j) lies in the first octant and outside $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$. The horizontal and the vertical grid lines passing through (i, j) intersect $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$ in its first quadrant at $\mathsf{H}(x_j, j)$ and $\mathsf{V}(i, y_i)$ respectively, and M is the midpoint of chord HV.

As shown in Fig. 7.3, it may be noted that, if (i, j) lies inside the shaded region, then both the *x-distance* and the *y-distance* are finite; otherwise, at least one of them is ∞, in accordance with Defs. 7.2.2 and 7.2.3. It may be also noted that Def. 7.2.1 can be easily extended to interprete the location of a point lying in one of the remaining seven octants. Similarly, the definitions of *x-distance* and *y-distance* for grid points, located in the remaining seven octants can be properly modified from those in octant 1 given in Defs. 7.2.2 and 7.2.3 respectively. It can be proved that if a grid point (i, j) lies in any one of octants 1, 8, 4, and 5, then $d_\perp = d_y$, and if it lies any one of the other four octants, namely octants 2, 7, 3, and 6, then $d_\perp = d_x$.

For instance, if (i, j) lies in the first octant, then Def. 7.2.4 ensures that $d_\perp = d_y$, since d_y never exceeds d_x. The proof is as follows. If $i \leqslant \rho$ and $j > \rho$, then $d_x = \infty, d_y < \infty$. If $i > \rho$, then both d_x and d_y are ∞. If $i \leqslant \rho$ and $j \leqslant \rho$, i.e., (i, j) lies inside the shaded region as shown in Fig. 7.3, then consider the midpoint $\mathsf{M}(x_{\mathsf{M}}, y_{\mathsf{M}})$ of the chord HV, where, the horizontal and the vertical grid lines passing through (i, j) intersect $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$ in the first quadrant (the first two octants) at $\mathsf{H}(x_j, j)$ and $\mathsf{V}(i, y_i)$ respectively. Let θ_{H} be the angle between OH and $+y$ axis, and θ_{V} be the angle between OV and $+x$ axis. Let the chord HV subtend an angle θ at O. M being the midpoint of chord HV, the line OM bisects the angle VOH. So, angle between $+y$ axis and OM is $\theta_{\mathsf{H}} + \theta/2$, and that between +x axis and OM is $\theta_{\mathsf{V}} + \theta/2$. Now, since (i, j) lies in the first octant, therefore, from Def. 7.2.1, $0 \leqslant i \leqslant j$. Hence, we get

$\cos^{-1}(i/\rho) \geqslant \cos^{-1}(j/\rho)$,

or, $\theta_{\mathsf{V}} \geqslant \theta_{\mathsf{H}}$, or, $\theta_{\mathsf{V}} + \theta/2 \geqslant \theta_{\mathsf{H}} + \theta/2$,

or, angle between $+y$ axis and $\mathsf{OM} \leqslant 45^o$,
or, M lies in the first octant, or, $x_{\mathsf{M}} \leqslant y_{\mathsf{M}}$ [from Def. 7.2.1],
or, $\frac{1}{2}(x_j + i) \leqslant \frac{1}{2}(j + y_i)$, or, $(j + y_i)/(i + x_j) \geqslant 1$.

Now, since $\mathsf{H}(x_j, j)$ and $\mathsf{V}(i, y_i)$ lie on $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$, we have
$x_j^2 + j^2 = \rho^2 = i^2 + y_i^2$, or, $i^2 - x_j^2 = j^2 - y_i^2$,
or, $(i - x_j)/(j - y_i) = (j + y_i)/(i + x_j)$,
or, $d_x/d_y = (j + y_i)/(i + x_j) \geqslant 1$, or, $d_y \leqslant d_x$.

For all other octants, the proof is similar as given above. These findings are put together in Theorem 7.2.1.

Theorem 7.2.1 *If a grid point* (i, j) *lies outside* $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$*, then its xy-distance from* $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$ *is given by*

$$d_{\perp} = \begin{cases} d_y, & \text{if } (i,j) \text{ lies in octant } 1/8/4/5; \\ d_x, & \text{if } (i,j) \text{ lies in octant } 2/7/3/6. \end{cases}$$

Hence, if any grid point (i, j) lies on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, \rho)$ but outside $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$, then it must have isothetic distance strictly less than $\frac{1}{2}$ grid unit from $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$. And this is true as well if the center of the circle is at any grid point α instead of O. The fact that isothetic distance of (i, j) is "strictly less" than $\frac{1}{2}$ grid unit from $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$ owes to our tie-resolution policy, where the grid point lying on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, \rho)$, corresponding to a tie (occurring due to computation error only), lies "inside" the real circle $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$. This, in turn, ensures that any grid point, lying outside $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$ in any octant with isothetic distance not less than $\frac{1}{2}$ grid unit from $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$, does not lie on $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, \rho)$. This analysis is captured in Theorem 7.2.2.

Theorem 7.2.2 *Any grid point, not lying on* $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$ *and lying outside* $\mathcal{C}^{\mathbb{R}}(\alpha, \rho)$*, must be located at xy-distance of at least* $\frac{1}{2}$ *grid unit from* $\mathcal{C}^{\mathbb{R}}(\alpha, \rho)$*.*

7.2.2 Enclosing Circle

Let $\mu(i_\mu, j_\mu)$ be a grid point, lying on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, \rho)$, such that μ has maximum y-distance from $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$, the corresponding y-distance being δ_μ, where, $0 < \delta_\mu < \frac{1}{2}$ for $\rho \geqslant 2$ (for $\rho = 1$, $\delta_\mu = 0$). If $(\rho_\mu, \theta_\mu) \in \mathbb{R}^2$ be the corresponding polar coordinates of μ, then $\rho_\mu = \sqrt{i_\mu^2 + j_\mu^2} = \rho + \varepsilon_\mu$, and $\theta_\mu = \tan^{-1}(j_\mu/i_\mu)$, where, $\varepsilon_\mu (< \delta_\mu \sin\theta_\mu)$ is the radial distance of μ from $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$, as shown in Fig. 7.4.

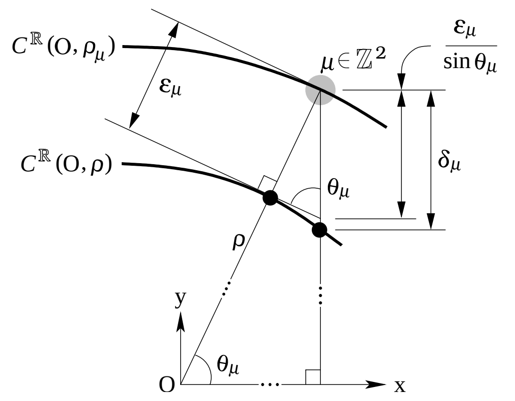


Figure 7.4: Isothetic distance, δ_μ $\left(0 < \delta_\mu < \frac{1}{2}\right)$, and radial distance, ε_μ $\left(\frac{1}{\sqrt{2}}\delta_\mu < \varepsilon_\mu < \delta_\mu\right)$, for $\mu \in \mathcal{C}^{\mathbb{Z},\mathrm{I}}(O,\rho)$, $\rho \geqslant 2$.

The next point of curiosity about a digital circle is that, whether μ (the grid point, lying on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(O,\rho)$, having maximum y-distance from $\mathcal{C}^{\mathbb{R}}(O,\rho)$) is unique or not. To uncover the fact about the uniqueness of μ, let us consider two distinct grid points, μ_1 and μ_2, lying on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(O,\rho)$, such that each of them has maximum y-distance, δ_μ, from $\mathcal{C}^{\mathbb{R}}(O,\rho)$. Let $\mu_1 = (i_1, j_1)$ and $\mu_2 = (i_2, j_2)$. Since in the first octant, no two grid points lying on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(O,\rho)$ can have same abscissa, $i_1 \neq i_2$. Therefore, w.l.g., let $i_1 > i_2$, which implies $j_2 > j_1$, since both μ_1 and μ_2 lie on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(O,\rho)$. It may be observed that, since $\mu_1 = (i_1, j_1)$ and $\mu_2 = (i_2, j_2)$ have same y-distance from $\mathcal{C}^{\mathbb{R}}(O,\rho)$, $j_1 = j_2$ is not possible. Let the corresponding points of intersection of $\mathcal{C}^{\mathbb{R}}(O,\rho)$ with the vertical grid lines $x = i_1$ and $x = i_2$ in the first octant be (i_1, y_1) and (i_2, y_2) respectively, where, $y_1, y_2 \in \mathbb{R}^+ \smallsetminus \mathbb{Z}^+$. Since (i_1, y_1) and (i_2, y_2) lie on $\mathcal{C}^{\mathbb{R}}(O,\rho)$, we have

$i_1^2 + y_1^2 = i_2^2 + y_2^2 = \rho^2$, and $j_1 - y_1 = j_2 - y_2 = \delta_\mu$.

So, $y_2^2 - y_1^2 = i_1^2 - i_2^2$,
or, $y_2^2 - y_1^2 \in \mathbb{Z}^+$ [since $i_1^2 - i_2^2 \in \mathbb{Z}^+$],
or, $(y_1 + y_2)(y_2 - y_1) \in \mathbb{Z}^+$,
or, $(y_1 + y_2)((j_2 - \delta_\mu) - (j_1 - \delta_\mu)) \in \mathbb{Z}^+$,
or, $(y_1 + y_2)$ is rational [since $(j_2 - j_1) \in \mathbb{Z}^+$].

Now $y_1^2 = \rho^2 - i_1^2 = a$ (say) is an integer but not a perfect square, since $y_1 \notin \mathbb{Z}$. Similarly, $y_2^2 = \rho^2 - i_2^2 = b$ (say) is also a non-square integer. Therefore, $y_1 + y_2 = \sqrt{a} + \sqrt{b}$ would be irrational. The reason is as follows. Let $\sqrt{a} + \sqrt{b}$ be a rational number c, if possible. Then, $a = (c - \sqrt{b})^2 = c^2 + b - 2c\sqrt{b}$. Since the product of any rational number and any irrational number is always irrational, $2c\sqrt{b}$ is irrational. Hence a becomes irrational, which is a contradiction. [It may be noted that the sum of two irrational numbers is not necessarily an irrational number; but if a and b are any two non-square integers, then $\sqrt{a} + \sqrt{b}$ would be always irrational.]

Therefore, $y_1 + y_2$ can never be rational, thereby contradicting our assumption that μ is not unique. Hence μ is a unique grid point in the first octant that lies on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, \rho)$, and has maximum y-distance from $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$. The result obtained, along with its proof (with minor modifications), can be applied equally well for any arbitrary center $\alpha \in \mathbb{Z}^2$. Thus we have the following theorem.

Theorem 7.2.3 *In the set of grid points lying on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha, \rho)$, there is a unique grid point that has maximum y-distance from $\mathcal{C}^{\mathbb{R}}(\alpha, \rho)$.*

Now consider the circle $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho_\mu)$ passing through μ (Fig. 7.4). Since μ lies in the first octant, we have $0 < i_\mu \leqslant j_\mu$ (for $\rho \geqslant 2$, $i_\mu \neq 0$), whence $45^0 < \theta_\mu < 90^o$, which implies $\frac{1}{\sqrt{2}}\delta_\mu < \varepsilon_\mu < \delta_\mu$. Let $\nu(i_\nu, j_\nu)$, $\nu \neq \mu$, be any grid point that lies on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, \rho)$. Let δ_ν be the y-distance of ν from $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$, and (ρ_ν, θ_ν) be the corresponding polar coordinates of ν, such that $\rho_\nu = \sqrt{i_\nu^2 + j_\nu^2} = \rho + \varepsilon_\nu$, and $\theta_\nu = \tan^{-1}(j_\nu / i_\nu)$, where $\varepsilon_\nu = \delta_\nu \sin\theta_\nu$. Therefore, ν would lie on or inside $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho_\mu)$ if and only if $\varepsilon_\nu =\leqslant \varepsilon_\mu$. Further, since μ is the grid point having maximum y-distance from $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$, we have $\delta_\nu < \delta_\mu$ ($\delta_\nu \neq \delta_\mu$, since from Theorem 7.2.3, the grid point in the first octant with maximum y-distance is unique). Hence, if ν be such that in spite of δ_ν being less that δ_μ, θ_ν is so high compared to θ_μ that ε_ν is higher than ε_μ, then ν does not lie on or within $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho_\mu)$.

On the contrary, if we call the point $\breve{\mu}$ as the grid point lying on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, \rho)$, for which $\varepsilon_{\breve{\mu}}$ is maximum in $\{\varepsilon_\nu : \nu \text{ lies on } \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, \rho)\}$, then it may happen that there exists some grid point $\nu' = (i', j')$ lying in the first octant and outside $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, \rho)$ for which $\varepsilon_{\nu'}$ does not exceed $\varepsilon_{\breve{\mu}}$ (although $\delta_{\nu'}$ is not less than $\frac{1}{2}$, as stated in Theorem 7.2.2), and therefore, ν' lies on or inside $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho_{\breve{\mu}})$.

Based on the above findings, it can be inferred that, for the set of all digital circles $\{\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, \rho) : \rho \in \mathbb{Z}^+\}$, we cannot find the canonical solution for the set of enclosing circles, $\{\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho') : \rho' \in \mathbb{R}^+\}$, for all possible values of ρ, so that for each $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, \rho)$, no grid points but those lying on or inside $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, \rho)$ would lie on or inside $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho')$. And no less importantly, we can for sure construct the circle $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho_\mu)$, passing through μ, where, μ has maximum y-distance from $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$, so that all grid points, lying on or inside $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, \rho)$, would lie on or inside $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho_\mu)$, if and only if μ has maximum radial distance (ε_μ) from $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho)$ among all grid points lying on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, \rho)$. This is stated in Theorem 7.2.4, where the center of the digital circle, in general, is considered at $\alpha \in \mathbb{Z}^2$, with obvious justification.

Definition 7.2.6 If there exists a circle $\mathcal{C}^{\mathbb{R}}(\alpha, \rho')$, such that all grid points on and inside $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$ would lie on and inside $\mathcal{C}^{\mathbb{R}}(\alpha, \rho')$ and no grid point outside $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$ would lie on or inside $\mathcal{C}^{\mathbb{R}}(\alpha, \rho')$, then the circle $\mathcal{C}^{\mathbb{R}}(\alpha, \rho')$ is called *an enclosing circle* for the digital circle $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$. Such an enclosing circle with minimum radius is termed as *the smallest enclosing circle*, and that with maximum radius is termed as *the largest enclosing circle*, for the digital circle $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$.

Theorem 7.2.4 *If there exists a grid point μ lying on $dcirI(\alpha, \rho)$, such that*
$$\delta_\mu = \max\left\{\delta_\nu : \nu \textit{ lies on } \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha, \rho)\right\} \textit{ and } \varepsilon_\mu = \max\left\{\varepsilon_\nu : \nu \textit{ lies on } \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha, \rho)\right\},$$
then $\mathcal{C}^{\mathbb{R}}(\alpha, \rho_\mu)$ is the smallest enclosing circle for $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$.

If no such grid point μ exists for $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha, \rho)$, then the existence of an enclosing circle for $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$ becomes uncertain.

7.2.3 Enclosing Polygon

The following definitions are needed in the context of enclosing polygons.

Definition 7.2.7 A *central chord* of a regular polygon in $\mathbb{R}^2$ is any line segment in $\mathbb{R}^2$ whose end points lie on the polygon and which passes through the center of the polygon.

Definition 7.2.8 A regular polygon in $\mathbb{R}^2$ that encloses the digital circle $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$, such that each point in $\mathbb{Z}^2$ lying on or inside the digital circle $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$ lies on or inside the polygon, and no point in $\mathbb{Z}^2$ lying outside the digital circle $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$ lies on or inside the polygon, is defined as a *regular polygonal enclosure*, $\mathcal{E}^{\mathbb{R}}(\mathcal{C}^{\mathbb{Z}}(\alpha, \rho))$, for the digital circle $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$.

Let, for a given value of $\rho \in \mathbb{Z}$, there be a grid point μ lying on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, \rho)$, such that $\delta_\mu = \max\left\{\delta_\nu : \nu \text{ lies on } \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, \rho)\right\}$ and $\varepsilon_\mu = \max\left\{\varepsilon_\nu : \nu \text{ lies on } \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O}, \rho)\right\}$. Then, from Theorem 7.2.4, $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho_\mu)$ is the smallest enclosing circle for $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, \rho)$. Hence, any grid point ν', lying in the first octant and outside $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, \rho)$, lies outside $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho_\mu)$.

Let μ' be a grid point, lying in the first octant and outside $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho_\mu)$, such that $\varepsilon_{\mu'} = \min\left\{\varepsilon_{\nu'} = \rho_{\nu'} - \rho : \nu' \in \mathbb{Z}^2 \text{ lies in the first octant and outside } \mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho_\mu)\right\}$, where, $(\rho_{\nu'}, \theta_{\nu'}) \in \mathbb{R}^2$ and $(\rho_{\mu'}, \theta_{\mu'}) \in \mathbb{R}^2$ are the polar coordinates of ν' and μ' respectively. Let $\tilde{\rho} \in \mathbb{R}^+$ be the quantity given by $\tilde{\rho} = \max\left\{\rho' : \rho_\mu < \rho' < \rho_{\mu'}\right\}$, i.e.,

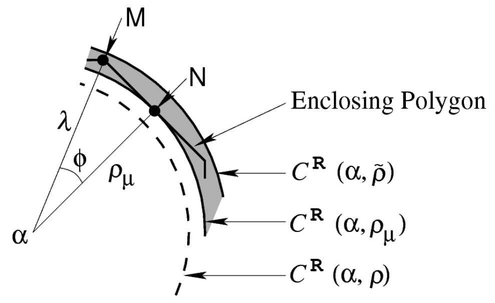

M
N
Enclosing Polygon
λ
φ
ρ_μ
α
$C^{\mathbf{R}}(\alpha, \tilde{\rho})$
$C^{\mathbf{R}}(\alpha, \rho_\mu)$
$C^{\mathbf{R}}(\alpha, \rho)$

Figure 7.5: Generation of the enclosing polygon of $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$.

$\tilde{\rho} = \text{limit}_{h \to 0+} \big(\rho_{\mu'} - h\big)$. Therefore, any grid point in the first octant that lies outside $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho_\mu)$ also lies outside $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \tilde{\rho})$. Furthermore, $\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \tilde{\rho})$ has to be the largest enclosing circle of $\mathcal{C}^{\mathbb{Z}}(\mathrm{O}, \rho)$ in accordance with our consideration of $\tilde{\rho}$. And this argument is true as well if we shift the center of the digital circle from O to α.

Thus, all grid points, lying on and inside $\mathcal{C}^{\mathbb{R}}(\alpha, \rho_\mu)$, lie (strictly) inside $\mathcal{C}^{\mathbb{R}}(\alpha, \tilde{\rho})$; and all grid points, lying (strictly) outside $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho_\mu)$, lie (strictly) outside $\mathcal{C}^{\mathbb{R}}(\alpha, \tilde{\rho})$. As a result, there exists no grid point that lies on or inside $\mathcal{C}^{\mathbb{R}}(\alpha, \tilde{\rho})$ and outside $\mathcal{C}^{\mathbb{R}}(\alpha, \rho_\mu)$. That is, the annular space defined by $\Big(\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \tilde{\rho}) \cup \overline{\mathcal{C}}^{\mathbb{R}}_{in}(\mathrm{O}, \tilde{\rho})\Big) \smallsetminus \Big(\mathcal{C}^{\mathbb{R}}(\mathrm{O}, \rho_\mu) \cup \overline{\mathcal{C}}^{\mathbb{R}}_{in}(\mathrm{O}, \rho_\mu)\Big)$ does not contain any grid point.

Now, in $\mathbb{R}^2$, we can construct a regular polygon, centered at α and having a set of n vertices, such that the following conditions are simultaneously satisfied.

(**c1**) each vertex lies on or inside $\mathcal{C}^{\mathbb{R}}(\alpha, \tilde{\rho})$;

(**c2**) each edge touches $\mathcal{C}^{\mathbb{R}}(\alpha, \rho_\mu)$;

(**c3**) each edge subtends same angle 2ϕ ($= \frac{2\pi}{n}$ rads.) at α.

Let N be the point of contact of one edge of the enclosing polygon for the digital circle $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$ with $\mathcal{C}^{\mathbb{R}}(\alpha, \rho_\mu)$, as shown in Fig. 7.5. Let M be one of the two vertices adjacent to N, and λ be the distance of M from α. So, the angle subtended at α by the line segment MN is ϕ, which implies $\cos\phi = \rho_\mu/\lambda$. Since M should lie inside the annular region between $\mathcal{C}^{\mathbb{R}}(\alpha, \rho_\mu)$ and $\mathcal{C}^{\mathbb{R}}(\alpha, \tilde{\rho})$, including the circumference of $\mathcal{C}^{\mathbb{R}}(\alpha, \tilde{\rho})$, λ should not exceed $\tilde{\rho}$. Hence, $\cos\phi$ should be at least $\rho_\mu/\tilde{\rho}$, which implies ϕ should be at most $\cos^{-1}(\rho_\mu/\tilde{\rho})$, or, ϕ should be strictly less than $\cos^{-1}(\rho_\mu/\rho_{\mu'})$, since $\tilde{\rho} < \rho_{\mu'}$. Also, ϕ (in radians) should be such that $\frac{\pi}{\phi} \in \mathbb{Z}$, since $n = \frac{\pi}{\phi}$ gives the number of edges of the enclosing polygon.

Since there exists no grid point that lies in the annular space between $\mathcal{C}^{\mathbb{R}}(\alpha, \rho_\mu)$ and $\mathcal{C}^{\mathbb{R}}(\alpha, \tilde{\rho})$ (including the circumference of $\mathcal{C}^{\mathbb{R}}(\alpha, \tilde{\rho})$), this polygon would be a *regular polygonal enclosure* for the digital circle $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$. Note that the shortest central chord for

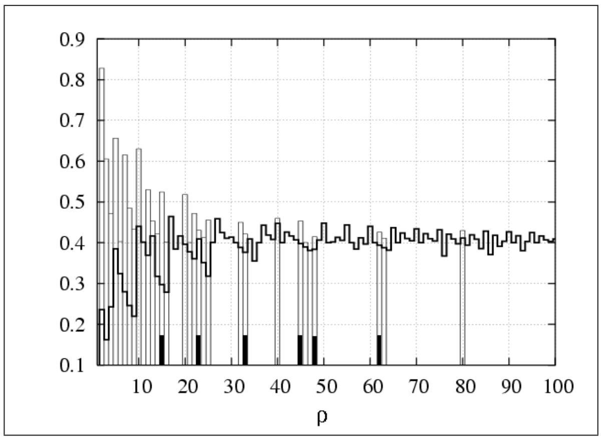


Figure 7.6: ε_μ (thick-lined silhouette) and $\varepsilon_{\mu'}$ (thin-lined bars) plotted against ρ. $\varepsilon_{\mu'}$ has been shown only when it exceeds ε_μ. A regular polygonal enclosure exists for a digital circle with radius ρ if $\varepsilon_{\mu'} > \varepsilon_\mu$.

this polygon would be the diameter of the circle $\mathcal{C}^{\mathbb{R}}(\alpha, \rho_\mu)$; and the longest central chord for this polygon would be at most the diameter of the circle $\mathcal{C}^{\mathbb{R}}(\alpha, \tilde{\rho})$ (when both the end points of each longest central chord of the polygon lie on $\mathcal{C}^{\mathbb{R}}(\alpha, \tilde{\rho})$). The following theorem, therefore, concludes these findings.

Theorem 7.2.5 *If there exists a grid point μ lying on $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha, \rho)$, such that*

$$\delta_\mu = \max\left\{\delta_\nu : \nu \textit{ lies on } \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha, \rho)\right\} \textit{ and } \varepsilon_\mu = \max\left\{\varepsilon_\nu : \nu \textit{ lies on } \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha, \rho)\right\},$$

then there always exists some $\mathcal{E}^{\mathbb{R}}(\mathcal{C}^{\mathbb{Z}}(\alpha, \rho))$ corresponding to $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$.

Now, subject to the conditions **c1**, **c2**, and **c3**, if 2ϕ be the angle subtended at α by each edge of the regular polygonal enclosure for the digital circle $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$, where, $\phi < \cos^{-1}(\rho_\mu/\rho_{\mu'})$, then the corresponding number of minimum edges $n_{\min}$ of the corresponding polygon can be easily obtained using Theorem 7.2.6.

Theorem 7.2.6 *Minimum number of edges $n_{\min}$ of $\mathcal{E}^{\mathbb{R}}(\mathcal{C}^{\mathbb{Z}}(\alpha, \rho))$ is given by $n_{\min} = \lfloor \pi/\phi_0 \rfloor + 1$, where, $\phi_0 = \cos^{-1}(\rho_\mu/\rho_{\mu'})$, μ' being the grid point with properties as discussed above.*

7.3 Experimental Results

As discussed in Sec. 7.2.2 and Sec. 7.2.3, and stated in Theorem 7.2.5, the existence of a regular polygonal enclosure (vide Def. 7.2.8) is always true, provided there exists

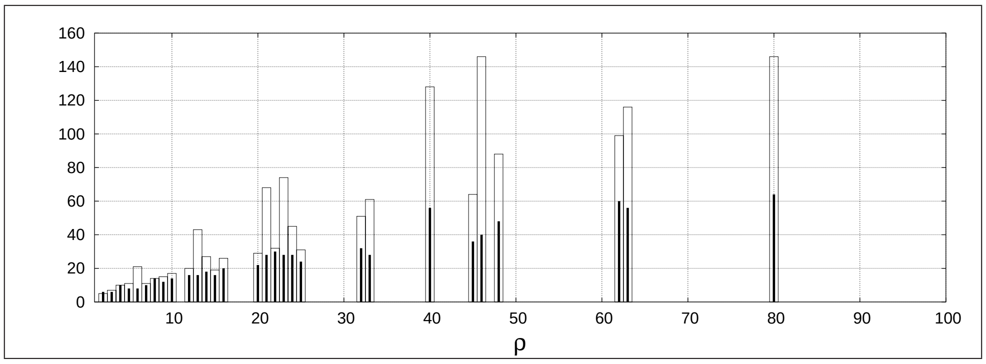


Figure 7.7: $2k_{\min}$ (thick line) and $n_{\min}$ (thin bars) plotted against ρ.

a grid point $\mu \in \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha,\rho)$, such that $\delta_\mu = \max\left\{\delta_\nu : \nu \text{ lies on } \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha,\rho)\right\}$ and $\varepsilon_\mu = \max\left\{\varepsilon_\nu : \nu \text{ lies on } \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha,\rho)\right\}$, that is, μ has simultaneously maximum isothetic distance and maximum radial distance from $\mathcal{C}^{\mathbb{R}}(\alpha,\rho)$.

Exhaustive procedural checking has revealed that each of the digital circles with radii from $\rho = 1$ to $\rho = 10$ has such a grid point μ that has simultaneously maximum isothetic distance and maximum radial distance, which implies that each of the digital circles with radii from $\rho = 1$ to $\rho = 10$ has regular polygonal enclosure. The chance of existence of such a grid point μ, however, goes on decreasing as the radius ρ of the circle gets increasing. In other words, the existence of a regular polygonal enclosure for a digital circle with radius ρ becomes more and more uncertain as the radius ρ goes on increasing. To cite a few more, for $\rho = 12-16, 20-25, 32, 33, 40$, the corresponding digital circles possess regular polygonal enclosures.

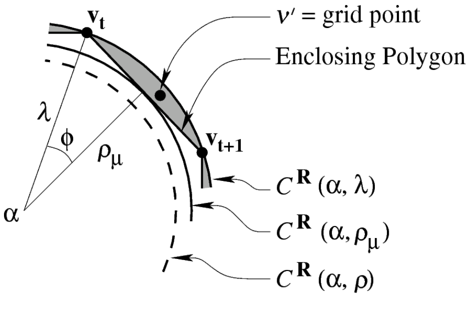


Figure 7.8: Enclosing polygon with $2k_{\min}$ vertices for the digital circle $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$. The gray region denotes the region where each grid point ν', lying outside $\mathcal{C}^{\mathbb{R}}(\alpha,\rho_\mu)$ and inside $\mathcal{C}^{\mathbb{R}}(\alpha,\lambda)$, always lies outside the polygon.

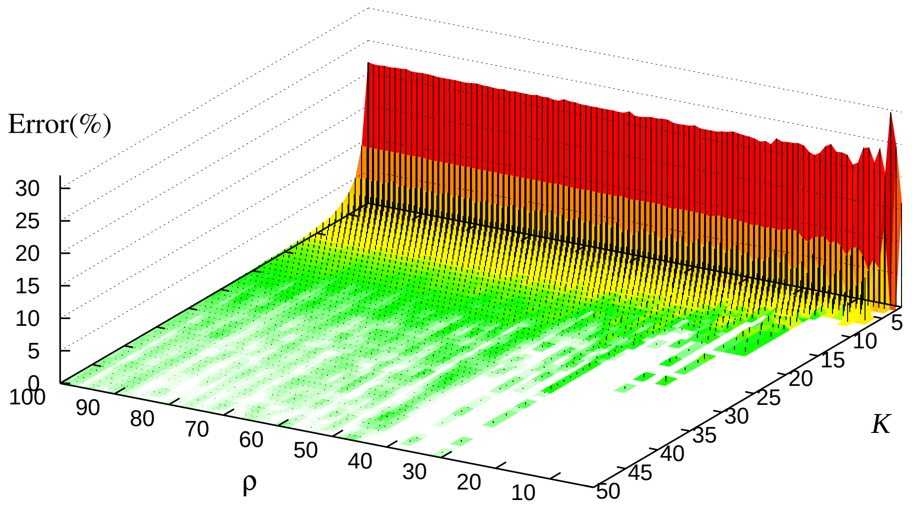


Figure 7.9: Error (%) plotted against ρ and k shows that higher values of $k = n/2$ drastically reduce the error.

The thin-lined vertical bars, shown in Fig. 7.6, indicate the circles for which regular polygonal enclosures exist. But it should be noticed that for each of these circles having regular polygonal enclosures, it is not always true that $\exists\ \mu \in \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha,\rho)$, such that $\delta_\mu = \max\left\{\delta_\nu : \nu \text{ lies on } \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha,\rho)\right\}$ and $\varepsilon_\mu = \max\left\{\varepsilon_\nu : \nu \text{ lies on } \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha,\rho)\right\}$ are valid simultaneously (vide Sec. 7.2.2 and Theorem 7.2.4). The fillets on the axis of ρ within the thin-lined bars, shown in Fig. 7.6, indicate the circles for which $\exists\ \mu \in \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha,\rho)$, such that $\delta_\mu < \max\left\{\delta_\nu : \nu \text{ lies on } \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha,\rho)\right\}$ and $\varepsilon_\mu = \max\left\{\varepsilon_\nu : \nu \text{ lies on } \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha,\rho)\right\}$. And those without fillets indicate the circles for which $\exists\ \mu \in \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha,\rho)$, such that $\delta_\mu = \max\left\{\delta_\nu : \nu \text{ lies on } \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha,\rho)\right\}$ and $\varepsilon_\mu = \max\left\{\varepsilon_\nu : \nu \text{ lies on } \mathcal{C}^{\mathbb{Z},\mathrm{I}}(\alpha,\rho)\right\}$.

Theoretically, if there exists a regular polygonal enclosure for a digital circle $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$, then in accordance with conditions **c1**, **c2**, and **c3**, and as expressed in Theorem 7.2.6, minimum number of edges $n_{\min}$ for the polygon is given by $n_{\min} = \lfloor \pi/\phi_0 \rfloor + 1$. While testing in our program with different digital circles, however, it is found that if a digital circle $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$ possesses a regular polygonal enclosure with minimum number of vertices $n_{\min}$ (theoretical), then in practice, it can be often enclosed by a regular polygonal enclosure with lesser number of vertices. This is illustrated in Fig. 7.7.

In Fig. 7.7, the thin-lined vertical bars denote the values of $n_{\min}$, plotted against the radii ρ of digital circles, for the regular polygonal enclosures possible for the corresponding digital circles. In this figure, the thick vertical lines represent the corresponding minimum even number of vertices, $2k_{\min}$, found experimentally, that the regular polygonal enclosures should possess, satisfying Def. 7.2.8. It may be observed that, for a digital circle, having $\rho > 2$ and whose regular polygonal enclosure exists, $2k_{\min}$ is appreciably lesser than the corresponding $n_{\min}$, which occurs due to the following reason.

Let λ be the distance of each of the $2k_{\min}$ ($< n_{\min}$) vertices of the regular polygonal enclosure from the center α of the corresponding circle $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$, as shown in Fig. 7.8. Then each of these $2k_{\min}$ vertices lies on $\mathcal{C}^{\mathbb{R}}(\alpha, \lambda)$. Let v_t and v_{t+1} be two such adjacent vertices of the polygon. Now it may happen that each grid point ν', lying outside $\mathcal{C}^{\mathbb{R}}(\alpha, \rho_\mu)$ and inside $\mathcal{C}^{\mathbb{R}}(\alpha, \lambda)$, always lies outside the enclosing polygon, as shown in Fig. 7.8.

It should be noted that, for those circles with $\varepsilon_{\mu'} \not> \varepsilon_\mu$, (regular) polygonal enclosures, as defined in Def. 7.2.8, are certainly not possible. For such digital circles, which are found to be the majority in the output of our program (some part of which is illustrated in Fig. 7.6), approximate polygonal enclosures with low error rates are possible. If $\mathcal{D}^{\mathbb{Z}}(\alpha, \rho)$ (disc with radius ρ and center α) represents the set of grid points in $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho) \cup \overline{\mathcal{C}}^{\mathbb{Z}}_{in}(\alpha, \rho)$, and $\mathcal{P}^{\mathbb{R}}\left(\mathcal{C}^{\mathbb{Z}}(\alpha, \rho), 2k\right)$ represents the set of grid points lying on and inside the approximate polygon with $2k$ vertices and one edge parallel to x-axis, such that $\mathcal{D}^{\mathbb{Z}}(\alpha, \rho) \subset \mathcal{P}^{\mathbb{R}}\left(\mathcal{C}^{\mathbb{Z}}(\alpha, \rho), 2k\right)$, then the number of erroneous grid points is given by $\left|\mathcal{P}^{\mathbb{R}}\left(\mathcal{C}^{\mathbb{Z}}(\alpha, \rho), 2k\right) \smallsetminus \mathcal{D}^{\mathbb{Z}}(\alpha, \rho)\right|$. The relative error, therefore, is given by

$$\text{error} = \frac{\left|\mathcal{P}^{\mathbb{R}}\left(\mathcal{C}^{\mathbb{Z}}(\alpha, \rho), 2k\right) \smallsetminus \mathcal{D}^{\mathbb{Z}}(\alpha, \rho)\right|}{|\mathcal{D}^{\mathbb{Z}}(\alpha, \rho)|} \times 100\%,$$

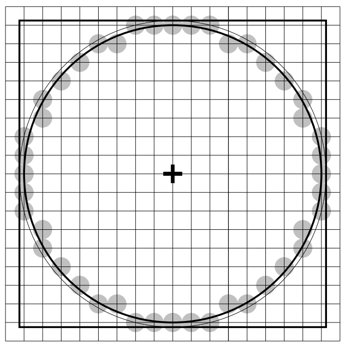
which is a measure of inaccuracy of the approximate polygon. The error, however, falls of drastically as the number of vertices of the approximate polygons increases, which is evident from Fig. 7.9. In Fig. 7.10, few approximate polygons have been shown for a sample digital circle with radius $\rho = 8$ to demonstrate the rapid convergence of the polygon towards ideal situation with increase in its number of vertices. In Fig. 7.11, we have shown the (exact/ideal) enclosing polygons for digital circles with radius r from 2 to 10.

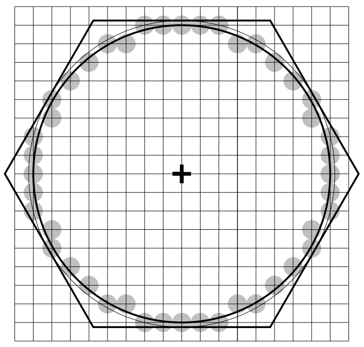
7.4 Conclusion

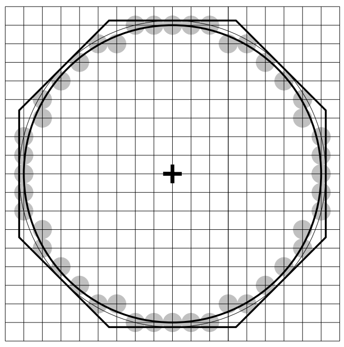
This work explores some interesting and useful properties of a digital circle, based on which an approximate (if not ideal) regular polygon of a digital circle can be determined.

As discussed in Sec. 7.1, identifying an approximate regular polygon will have several applications to the Approximate Point Set Pattern Matching (APSPM) problem. It is encouraging to note that, in fingerprint matching using the algorithms of APSPM, the circular range query is evoked for a circular range (disc) with radius not exceeding 10 pixels, which can be “ideally” replaced by a regular polygon. This will enable faster and better matching results, using an appropriate angular tree as explained in Chapter 2.

We have applied the facts and results discussed in this chapter to APSPM, with relevant extensions to fingerprint matching [Chapter 3] and for matching one object with another with their detected sets of corners [Chapter 4]. Observing the variation of accuracy and speed of the matching process with the level of approximation of the digital circle by a regular polygon, may be investigated in the future.


(a) $k = 2$: $\eta = 68$.


(b) $k = 3$: $\eta = 18$.


(c) $k = 4$: $\eta = 8$.

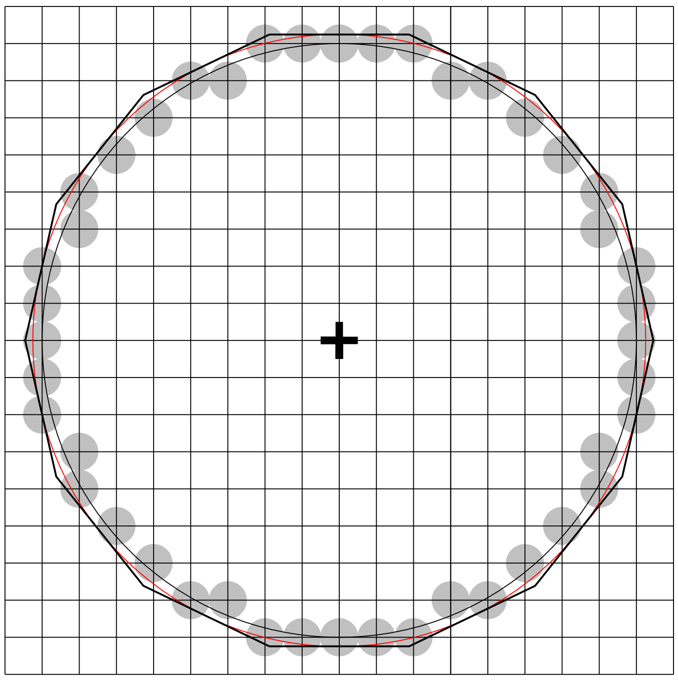

(d) $k = 7$: $\eta = 0$.

Figure 7.10: Few approximate regular polygons for the digital circle with radius $\rho = 8$. For other values of k, the corresponding errors are: $\eta(k = 5) = 8$, $\eta(k = 6) = 4$, and $\eta(k \geqslant 7) = 0$.

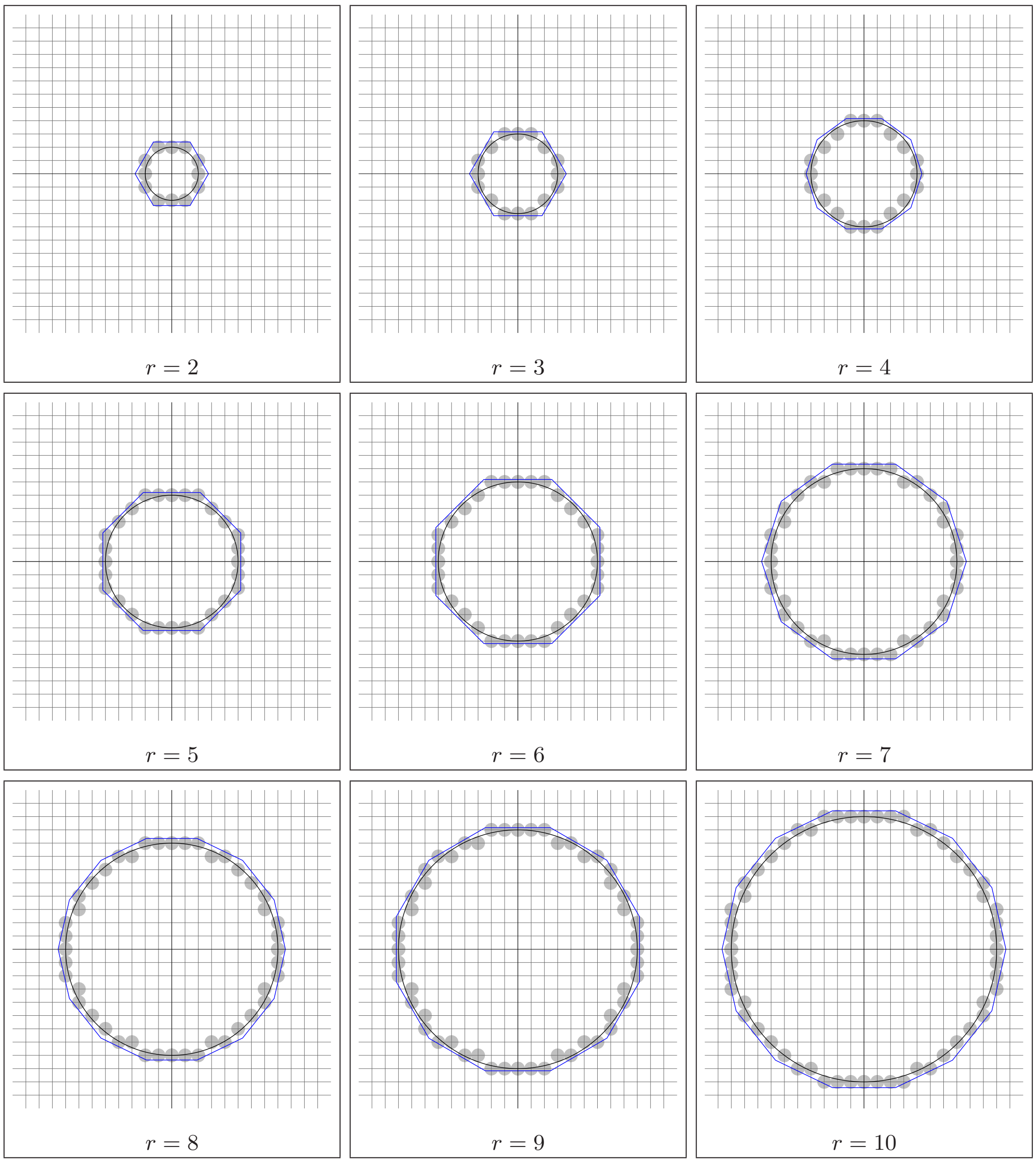


Figure 7.11: Enclosing polygons for radius r from 2 to 10.

Chapter 8

Cubic B-spline and its Usage for Correcting Digital Aberrations in Fingerprints

Even as the finite encloses an infinite series
And in the unlimited limits appear,
So the soul of immensity dwells in minutia
And in the narrowest limits no limit in here.
What joy to discern the minute in infinity!
The vast to perceive in the small, what divinity!

Jacob Bernoulli
Ars Conjectandi

8.1 Introduction

Score-based minutia matching in fingerprint images has been elaborated earlier in Chapter 3. In this chapter, we focus on a refinement technique targeted to eliminate digital aberrations from a fingerprint image. Such a refinement is required to improve the score finding mechanism [Chapter 3], in which the flow patterns of ridgelines are exploited for evaluation of the score of each valid minutia in a fingerprint image. The ridge refinement method, proposed here, successfully removes the digital aberrations present in a ridgeline. It may be pointed out that ridgeline aberrations creep in during the stage of ridge extraction (and the thinning procedure) from a gray-scale fingerprint image. The proposed method mainly involves smoothing of the aberrated ridgelines using appropriate cubic B-splines defined by a set of control points chosen judiciously for each ridgeline of a fingerprint image.

8.1.1 Fingerprint Image Processing

Most of the contemporary AFIS are based on a dual strategy that combines minutiae matching with some method that captures the ridge structure properties in order to improve the overall matching performance [Bhowmick *et al.* (2005a), Ceguerra and Koprinska (2002), He *et al.* (2003), Jain *et al.* (2001), Jiang and Yau (2000), Luo *et al.* (2000), Maltoni *et al.* (2003)]. Appropriate correction of the ridge topography is, therefore, necessary to design and implement a robust AFIS model. A scheme on adaptive filtering for enhancing distorted and damaged fingerprint images, which uses foreground and background characteristics (i.e., ridge and valley minutiae), especially to correct the broken ridges, has been discussed by Hung (1993). Combination of some statistical and structural techniques can also be employed for detecting the false minutia patterns in a fingerprint image [Xiao and Raafat (1991)]. A multi-resolution analysis of global texture and local orientation by wavelet transform has been proposed by Hsieh *et al.* (2003) to improve the clarity and continuity of ridge structures. Use of Gabor filter for fingerprint image enhancement has also been studied [Yang *et al.* (2003)]. A thin-plate spline (TPS) model has been proposed to treat elastic distortions in fingerprints in the minutiae matching method [Bazen and Gerez (2003)].

Thin-plate splines (TPS) have been also used very recently [Ross *et al.* (2006)] to estimate the nonlinear distortion in a pair of fingerprint images based on ridge curve correspondences. When multiple impressions of a specific finger are available, the corresponding "optimal" deformation is subsequently utilized to distort the template fingerprint prior to matching it with the input fingerprint. In another contemporary work [Chikkerur *et al.* (2007)], contextual filtering using Short Time Fourier Analysis (STFT) has been proposed in order to obtain the local ridge orientation and the ridge frequency information associated with a fingerprint image (and also to segment the image using the energy map received during STFT analysis). All these studies throughout the years, in essence, show the need and significance of related fingerprint enhancement schemes to take care of various deformations associated with a fingerprint image while designing an efficient fingerprint identification system.

The motivation of this work lies in deciding the authenticity of a local ridge structure (and an associated minutia, thereof) in its way of participating in the fingerprint matching process. It may be observed that, if the ridge and valley lines in the local neighborhood of a minutia P have a smooth nature of flow, then the corresponding minutia P should have a genuine contribution in the fingerprint matching. On the contrary, if in some region, the ridge and valley lines have an erratic or uneven nature of flow, a minutia P' in that region should not predominate the matching procedure. The former minutia (P), being located in a tidy region, lends more confidence in the matching procedure than the latter (P'), which is located in a noisy region. The relative role played by these two classes of minutiae in fingerprint matching has been discussed in detail by Bhowmick *et al.* (2005a).

An appropriate rectification of the ridgelines in a fingerprint image is, therefore, mandatory in order to achieve desired results in the subsequent processes. A fingerprint image, acquired in non-ideal situations, may have low and uneven contrast with poorly defined features. Hence, a poor quality fingerprint image may lead to a very frustrating matching result. Several problems listed below are responsible for the deteriorating performance of fingerprint matching.

(a1) fingerprints captured with unpredictable non-rigid transformations;
(a2) cuts and abrasions on the finger;
(a3) dirt, oil, or moisture on the finger tip or scanner;
(a4) digitization error in the acquisition mechanism;
(a5) nondeterministic/non-ideal behavior of fingerprint processing algorithms (e.g. ridge extraction, thinning of ridgelines, etc.).

The above-mentioned aberrations and malformations of a fingerprint image result in relative deviation of features (ridges and minutiae) from their actual locations, thereby posing severe problems in the subsequent steps of fingerprint matching.

Considering the above artifacts, a necessary foundation for a robust and efficient AFIS is to design a reliable post-processing stage that takes into account the aberrations present in the ridge topography of a fingerprint image (acquired and processed). A schematic diagram for a modified AFIS that incorporates (B-)spline correction for removal of ridgeline aberrations from the extracted binary skeleton of a fingerprint image is shown in Fig. 8.1. The stage on spline correction will effectuate the smooth flow of the ridgelines that, in turn, will ensure better processing in the subsequent stages.

The topography of ridgelines (coupled with valley-lines) in a fingerprint image has been observed to play a crucial role in estimating the authenticity of a detected minutia [Bhowmick *et al.* (2005a)], where the authenticity of a minutia has been referred to and measured as "score" (a numerical value that lies in $[0, 100]$) of the corresponding minutia. The score of a minutia is estimated using several distance measures defined on the ridge topography in and around the corresponding minutia, which correspond to the orderliness and smoothness of ridgelines prevailing in its associated neighborhood. A minutia having a

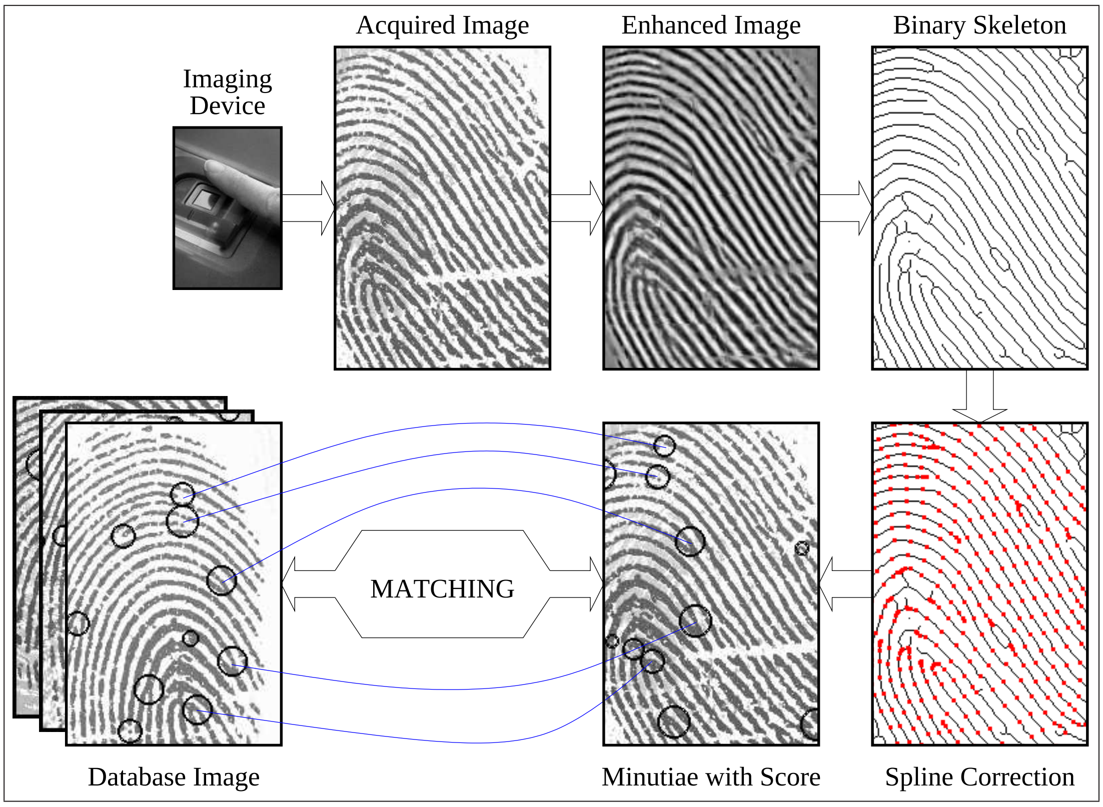


Figure 8.1: A generic scheme of AFIS that includes the proposed stage on spline correction for removal of ridgeline aberrations.

high score indicates its genuineness in its way of participation in the subsequent fingerprint matching, whereas a minutia with a low score is considered as a less reliable one in the matching stage. This leads to an efficient and reliable AFIS based on minutiae matching [Bhowmick and Bhattacharya (2004a)]. A high matching index is produced in the case of a good matching, whereas a poor matching produces a low index.

To further strengthen the score finding process (and the AFIS, thereof) that utilizes the ridge flow pattern in a fingerprint image, smoothing or correction of the ridgelines is thus necessary. The proposed work is focused on removal of digital aberrations in the fingerprint ridgelines, arising mainly out of the ridge extraction, followed by thinning, of a fingerprint image (problem (a5)). However, this problem (a5) is not independent of all the other problems mentioned earlier. In most of the cases, the problems (a2) and (a3) may cause problem (a5), since a poor quality gray-scale image is more liable to aberrate the corresponding binary ridge structure obtained by the ridge-extraction and thinning procedures. The proposed method mainly involves smoothing of the aberrated ridgelines using appropriate cubic B-splines defined by a set of control points chosen judiciously for each ridgeline of a fingerprint image. Experimental results on four different classes of benchmark data sets indicate the strength and novelty of the proposed algorithm.

8.1.2 Existing Methods

Existing approaches used to eliminate the unwanted deformations/distortions in fingerprints can be broadly categorized into two classes. The simplest is a combination of physical design and operator training. By proper use of a fingerprint capturing device, e.g., by guiding the finger to the capture surface with appropriate guides/moulds around the scanner, the fingertip pressure during capture can be controlled, specially for cooperative users. In the non-cooperative situation, although a trained person is there to monitor the acquisition process, the quality of the acquired print may not be always satisfactory.

In the second category, the methods are unsupervised. One such method, proposed by Dorai *et al.* (2000), measures distortion in a video sequence of fingerprint images when a finger is presented to the scanner. In the case of excessive distortion, the print would be rejected and a new print requested. In another method by Ratha and Bolle (1998), the forces and torques on the scanner are measured directly to prevent a capture when excessive force is applied. These methods, requiring specialized hardware, have the limitation that once a print is acquired, nothing can be done about the data imperfections. Large legacy databases are in use, containing prints of poor quality, which are not benefited by these techniques, since they, in principle, prevent the acceptance of the aberrated/distorted data.

Residual errors are, therefore, unavoidably present in the acquired image of a fingerprint, and in order to cope with them, an efficient AFIS should have the provision to withstand reasonable amount of errors in the input image. For instance, tolerance regions (circular/rectangular) have been used by Bhowmick and Bhattacharya (2004a), and by Ratha *et al.* (1996), so that a minutia in the query print can be made to match a minutia lying inside the corresponding tolerance region in the database print. Alternatively, the parameters of representation (using minutiae triplets) are binned/quantized in a procedure by Germain *et al.* (1997) in order to impart robustness to distortion and noise. The cumulative effects of small local perturbations have been shown with graphical demonstration to enhance the matching process [Kovács-Vajna (2000)]. This method also uses tolerance bounds of inter-minutiae distances and angles for minutiae correspondences. Another method based on local warping is reported, which tends to remove distortions to some extent but becomes vulnerable to high rate of false acceptance [Thebaud (1999)].

The proposed method differs from the existing ones in the sense that it considers the individual pattern of each ridgeline, and reconstructs it after minimizing its undulations/aberrations, if any. The reconstructed ridgeline is smoother and steadier than its earlier erratic form, and hence can be used more reliably in subsequent applications. One such application is discussed earlier [Bhowmick *et al.* (2005a)], where it has been shown how the flow pattern of ridges and valleys in the local neighborhood of a minutia can be exploited to decide whether or not a minutiae is true and also, how the distances of the ridges and valleys from a true minutia can be used to estimate its score. The scores of the true minutiae, can be used, in turn, to enhance the matching result in an (score-based) AFIS [Bhowmick and Bhattacharya (2004a), Bhowmick *et al.* (2005a)]. Hence, for validating the true minutiae in a fingerprint image and for proper estimation of scores of the true minutiae, an efficient algorithm for correcting the aberrated ridgelines is necessary.

8.2 Proposed Work

Digital aberrations in a fingerprint image are caused by unevenly located sequences of ridge and valley pixels, which upsets the smooth configuration of its ridge-valley topography. Given a binary image $\mathcal{I}$ of a fingerprint, our task is to remove such aberrations from the ridgelines by fitting appropriate *uniform non-rational B-splines* [Foley *et al.* (1993)]. In the rectified image $\mathcal{J}$, produced as the output, however, the set of minutiae and non-minutia points of discontinuity (stated in Sec. 8.1), $\{\mu_i\}_{i=1}^{n}$, as present in the input image $\mathcal{I}$, should remain unaltered. For removal of digital aberrations from an imperfect ridgeline, we have used a set of B-splines, which are cubic polynomial curves, as explained below:

(i) Polynomials with cubic degree are widely used in applications related to computer graphics and image processing for their simple but effective nature. The quadratic polynomial representation does not offer much flexibility, and for polynomials with higher degrees, there is always a trade-off between the complexity of the polynomial and the nature of the underlying digital curve.

(ii) Cubic B-splines possess C^0, C^1, and C^2 continuities[1], which ensure the smoothness and the optimal exactness of the fitted curve against the given set of control points.

[1]A curve that is only C^0 continuous may have a "kink" in the "knot point" between two consecutive segments. Hence a curve is made to be C^k continuous so that all k derivatives of the curve are continuous.

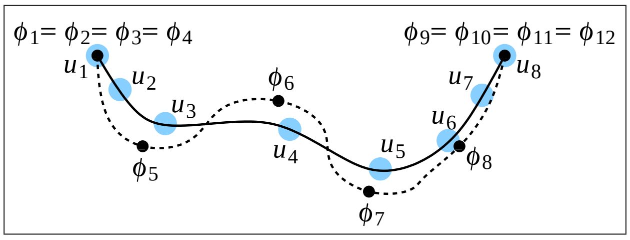

$\phi_1 = \phi_2 = \phi_3 = \phi_4$
$\phi_9 = \phi_{10} = \phi_{11} = \phi_{12}$
u_1
u_2
u_3
u_4
u_5
u_6
u_7
u_8
ϕ_5
ϕ_6
ϕ_7
ϕ_8

Figure 8.2: A set of B-spline segments (solid line), fitted against a set of control points $\{\phi_1, \phi_2, \ldots, \phi_{12}\}$, possesses lesser irregularity than the curve (dotted line) containing the control points, the "knot points" being $\{u_1, u_2, \ldots, u_8\}$.

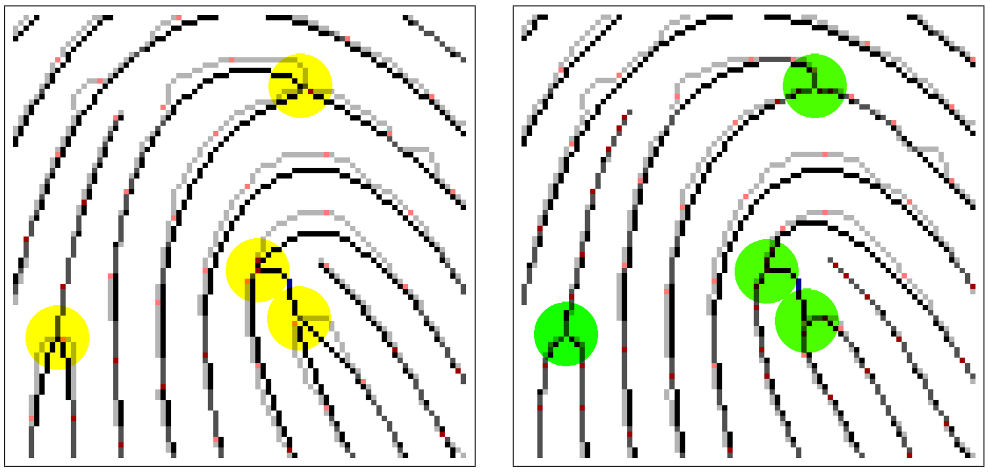
(iii) Since a set of B-splines is piecewise continuous, the order of the curve that interpolates a set of $m(\geq 4)$ control points is not dependent on m.

An example of a set of B-spline segments, fitted against an input set of control points $\{\phi_1, \phi_2, \ldots, \phi_{12}\}$, is shown in Fig. 8.2. It may be noted that, the start point of the B-spline coincides with the first four (coincident) control points, and its end point with the last four (coincident) control points, which have been intentionally done by making three copies of the first control point and three copies of the last control point. In Fig. 8.2, there exist seven B-spline segments of positive lengths – the first one being from knot point u_1 to knot point u_2, the second from u_2 to u_3, ..., and the seventh from u_7 to u_8; in addition, there also exist two B-splines of zero length each (degenerate case), one at u_1 and the other at u_8.

In order to extract the thinned binary image $\mathcal{I}$ representing the ridge topography from a gray-scale fingerprint image $\mathcal{F}$, we have used an earlier ridge-extraction algorithm [Bishnu *et al.* (2002)]. The set $\{\mu_i\}_{i=1}^{n}$, consisting of minutiae (bifurcation and termination) and points of discontinuity in the binary fingerprint image $\mathcal{I}$, have been extracted and stored in a list $\mathcal{L}$. Hence, each ridgeline has exactly two endpoints, μ_p and μ_q, $1 \leq p, q \leq n$, which are minutia(e) or non-minutia point(s) of discontinuity in the detected set $\{\mu_i\}_{i=1}^{n}$. Let $\rho(p, q)$ represent the ridgeline with endpoints μ_p and μ_q.

Now, for each ridgeline $\rho(p, q)$, an ordered set of control points, $\langle \phi_j(p, q) \rangle_{j=1}^{m}$, is defined starting from $\phi_1(p, q) = \mu_p$ and ending at $\phi_m(p, q) = \mu_q$, using the average inter-ridge distance λ (nearest integer value) corresponding to $\mathcal{I}$. It may be noted that, for the ridgeline $\rho(p, q)$, each control point $\phi_j(p, q)$, $1 \leq j \leq m$, lies on $\rho(p, q)$. Further, the control points are taken in a manner so as to minimize the deviation of the overall flow pattern of the corrected ridgeline $\tilde{\rho}(p, q)$ from the original (aberrated) ridgeline $\rho(p, q)$.

Since the curvature of the ridgeline is likely to be higher near either of the endpoints, namely $\phi_1(p,q)$ and $\phi_m(p,q)$, we have considered higher density of control points near $\phi_1(p,q)$ and $\phi_m(p,q)$, and fewer control points in the intermediate region of $\rho(p,q)$ (see Figs. 8.3 and 8.4). This strategy is based on the fact that authenticity of a minutia is adjudged by the ridge topography in the local neighborhood of the corresponding minutia [Bhowmick *et al.* (2005a)], and requires minimal digression of the ridge segments in that region.


(a) uniformly spaced control points

(b) control points located densely near a minutia and sparsely located away from it

Figure 8.3: Selecting control points (red pixels) with increasing density (i.e., decreasing separation) near a minutia ensures little or no deviation of the corrected ridgeline from its original location near the corresponding minutia. The figure has been shown with $\tau = 2$. See Fig. 8.4 and text for further explanations.

8.2.1 Formation of B-splines

Let $\Phi := [\phi_{s-1}\ \phi_s\ \phi_{s+1}\ \phi_{s+2}]'$ ($\phi_j := [x_j\ y_j]$, for $s-1 \leq j \leq s+2$) be the *geometry vector* defined by the set of 4 successive control points chosen from the ridgeline $\rho(p,q)$, required for each spline segment that would (piecewise) replace the ridge segment from ϕ_{s-1} to ϕ_{s+2}. Then the parametric form of a point $P := [x(u)\ y(u)]$ lying on the B-spline curve segment, defined by these four control points, considering u ($0 \leq u \leq 1$) as the corresponding parameter, can be expressed as:

$$P = UC, \tag{8.1}$$

using the *parameter vector*

$$U := [u^3 \; u^2 \; u \; 1], \tag{8.2}$$

and the *coefficient matrix*

$$C := \begin{bmatrix} c_{x3} & c_{y3} \\ c_{x2} & c_{y2} \\ c_{x1} & c_{y1} \\ c_{x0} & c_{y0} \end{bmatrix}. \tag{8.3}$$

Since a *basis matrix* $B := [b_{ij}]_{4\times4}$ is necessary to capture the underlying nature of a B-spline segment, the coefficient matrix is rewritten as

$$[C]_{4\times2} = [B]_{4\times4}[\Phi]_{4\times2}, \tag{8.4}$$

whence we get

$$P = UB\Phi. \tag{8.5}$$

Now, for uniform non-rational B-splines, the basis matrix [Foley *et al.* (1993)] is given by

$$B = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}, \tag{8.6}$$

which is obtained from the parametric continuity and the geometric continuity of B-splines, as mentioned earlier. Hence, from Eqn. 8.5, we get

$$\begin{aligned} x(u) &= \frac{1}{6}(-x_{s-1} + 3x_s - 3x_{s+1} + x_{s+2})u^3 + \frac{1}{2}(x_{s-1} - 2x_s + x_{s+1})u^2 + \\ &\quad \frac{1}{2}(-x_{s-1} + x_{s+1})u + \frac{1}{6}(x_{s-1} + 4x_s + x_{s+1}), \end{aligned} \tag{8.7}$$

$$\begin{aligned} y(u) &= \frac{1}{6}(-y_{s-1} + 3y_s - 3y_{s+1} + y_{s+2})u^3 + \frac{1}{2}(y_{s-1} - 2y_s + y_{s+1})u^2 + \\ &\quad \frac{1}{2}(-y_{s-1} + y_{s+1})u + \frac{1}{6}(y_{s-1} + 4y_s + y_{s+1}). \end{aligned} \tag{8.8}$$

8.2.2 Parameter Setting

It is important to note that the average inter-ridge distance λ plays a crucial role in our method. The maximum distance between two consecutive control points, $\phi_j(p, q)$ and $\phi_{j+1}(p, q)$, $1 \leq j \leq m-1$, for each ridgeline $\rho(p, q)$, has been considered to be $\tau\lambda$, where τ is defined as the *smoothness parameter*, the rationale being as follows. Since λ is the average inter-ridge distance of $\mathcal{I}$, the ridge topography of $\mathcal{I}$ largely depends on λ [Haralick (1983), Kovács-Vajna *et al.* (2000)] and the length of each ridgeline $\rho(p, q)$ also varies with λ for a particular finger depending on the resolution/scale of the acquired image. Further, the trade-off deciding the level of accuracy to which an aberrated ridgeline would be corrected versus the departure of the corrected ridgeline from the original (aberrated) one, would depend on the number of control points selected for the corresponding ridgeline. With fewer control points selected for a ridgeline $\rho(p, q)$, the corresponding output ridgeline $\tilde{\rho}(p, q)$ would be smoother, allowing higher deviation of $\tilde{\rho}(p, q)$ from $\rho(p, q)$. When more control points are chosen, the output ridgeline $\tilde{\rho}(p, q)$ would become less smoother because of its lesser deviation from $\rho(p, q)$.

Hence, a proper selection of τ is of primary concern in the proposed method. A change in the value of τ will have several significant consequences in the final output, some of which are as follows:

(i) A higher value of τ increases the separation between the control points, thereby improving the quality of corrected ridges and reducing the overall time requirement for smoothing the ridges in a given fingerprint image.

(ii) A lower value of τ reduces deviation from the original input image, i.e., the output image approximates the input image more accurately, although overall time requirement increases in this case.

An optimal value of τ should be chosen, therefore, depending upon the characteristics of the input fingerprint image and the subsequent purpose of use of the output image. It should be noted that, a value of τ in the range $[1, 2]$ produces desired results, as observed in our experiments with the four benchmark image databases we have considered.

Another parameter that defines a set of B-splines is the total number of control points given as input. Although four control points are sufficient to define a (single) B-spline, four closely located points on the discrete plane may not give rise to a digitally smooth B-spline segment. Hence, we have applied the smoothing procedure on a ridgeline $\rho(p, q)$ only if its digital length $L(\rho(p, q))$ exceeds a predefined threshold $L_{\min}$, which is defined as follows:

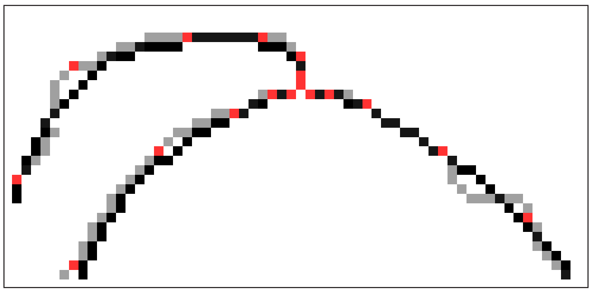


Figure 8.4: Corrected ridgelines (black) versus distorted ones (gray) for a sample fingerprint image. Red pixels indicate control points for defining the set of B-splines.

$$L_{\min} = \tau\lambda + 2\left(\lfloor \tau\lambda/2 \rfloor + \lfloor \tau\lambda/2^2 \rfloor + \cdots + 1\right) \tag{8.9}$$

The above equation owes from our consideration of densely located control points near the endpoints μ_p and μ_q, and sparsely located control points in the intermediate part of a ridgeline $\rho(p, q)$. We consider the second control point $\phi_2(p, q)$ at unit distance from the first control point $\mu_p = \phi_1(p, q)$, the third control point $\phi_3(p, q)$ at a distance of two units from the second, and so on, such that the maximum distance between two consecutive control points is $\tau\lambda$, as mentioned above. The trailing control points, $\phi_{m-1}(p, q), \phi_{m-2}(p, q), \ldots$, are also selected in a similar fashion with respect to the last control point $\phi_m(p, q)$. This is followed in order to impose some strictness of ridgeline characteristics near each minutia, failing which, the local ridge topography in and around the minutia may get digressed from its original one. Fig. 8.3 demonstrates its rationale. It is evident from Fig. 8.3(a) that a uniform spacing of control points along a ridge segment incident at a minutia, suffers from undesired deviations of the corrected ridge segments from the original one (the region centered around each (bifurcation) minutia highlighted in yellow); on the contrary, as shown in Fig. 8.3(b), selecting control points with increasing density (i.e., decreasing separation) towards a minutia ensures little or no deviation of the corrected ridgeline from its original location near the corresponding minutia (the region centered around each (bifurcation) minutia highlighted in green).

There exist some subtle characteristics of a B-spline and related issues on its implementation, which should be mentioned in this context. Since a B-spline not necessarily passes through its defining control points, the start point and the end point of the fitted B-spline segment often deviate from the desired points, i.e., the first and the last control points respectively. In order to overcome this limitation, for each ordered set of B-spline segments,

$\langle \mathcal{B}_k(p,q) \rangle_{k=1}^{m-3}$, defined by the ordered set of m control points, $\langle \phi_j(p,q) \rangle_{j=1}^{m}$, corresponding to the ridgeline $\rho(p,q)$, the start point of the first B-spline segment, $\mathcal{B}_1(p,q)$, is forced to coincide with the first control point, $\mu_p = \phi_1(p,q)$, and end point of the last B-spline segment, $\mathcal{B}_{m-3}(p,q)$, is forced to coincide with the last control point, $\mu_q = \phi_m(p,q)$. This is achieved by copying $\phi_1(p,q)$ three times, thereby creating four coincident control points at the start point μ_p of the ridgeline, and similarly creating four coincident control points at the end point μ_q. An example on fitted B-splines for (part of) three irregularly flowing ridges (gray pixels) incident at a bifurcation minutia has been shown in Fig. 8.4, in which the control points (shown in red) are densely placed near the minutia and sparsely away from the minutia, thereby producing the desired B-spline-corrected output (black pixels).

8.2.3 Proposed Algorithm

For obtaining the ternary fingerprint image consisting of ridges and valleys (against the background) from a gray-scale image, and subsequently extracting the minutiae and the noisy features (e.g., bridges, loops, and spurs), we have used an earlier algorithm [Acharya *et al.* (2006)]. The procedure for removal of digital aberrations from a skeletonized binary fingerprint image $\mathcal{I}$ starts with the extraction of minutiae and points of discontinuity, and storing these points in a list $\mathcal{L}$. For each point in $\mathcal{L}$, there is one or more ridge segments incident on it, which are obtained using the depth-first-search (DFS) procedure [Cormen *et al.* (2000)] initiated from the corresponding point. It may be noted that if μ_p is a point in $\mathcal{L}$, from which a DFS is invoked to traverse and obtain a ridge segment $\rho_k(\mu_p,\mu_q)$ ending at some other point μ_q in $\mathcal{L}$, then the ridge segment $\rho_k(\mu_q,\mu_p)$, being identical with $\rho_k(\mu_p,\mu_q)$, is not again traversed when μ_q is processed in its turn.

The process of defining the control points for each ridge segment ρ_k is executed, provided $L(\rho_k) \geq L_{\min}$, as explained in Sec. 8.2.2. If $L(\rho_k) < L_{\min}$, then the ridge segment ρ_k is left unaltered. Here $L(\rho_k)$ denotes the number of grid points (pixels) lying on the ridge ρ_k. The control points are used to draw a set of B-spline segments for each valid ridge segment ρ_k. If there are m control points, then $m-3$ B-spline segments are drawn to replace the ridge segment ρ_k. Finally, the corrected image $\mathcal{J}$ consists of the (spline-)corrected ridge segments whose lengths (in $\mathcal{I}$) are at least $L_{\min}$ and few uncorrected segments whose lengths are less than $L_{\min}$.

In Fig. 8.5, the proposed algorithm is presented in a concise form. For finding the average inter-ridge distance, λ, a procedure by Bhowmick *et al.* (2005a) is used. Steps 1 to 4 of the algorithm Correct-Ridges are needed for extracting the minutiae and points

```
Algorithm CORRECT-RIDGES ($image\ \mathcal{I}, int\ \lambda, float\ \tau, image\ \mathcal{J}$)
1.   $\mathcal{L} \leftarrow \emptyset$
2.   for each $(x, y) \in \mathcal{I}$
3.       if $|\{(x', y') : \max(|x' - x|, |y' - y|) = 1\}| \neq 2$
4.           $\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, y)\}$
5.   for each point $\mu_p \in \mathcal{L}$
6.       $\{\mu_q\} \leftarrow$ DEPTH-FIRST-SEARCH $(\mathcal{I}, \mu_p)$
7.           for each ridge segment $\rho_k(p, q)$ from $\mu_p$ to $\mu_q$
8.               if $L(\mu_p, \mu_q) \geq L_{\min}$
9.                   $\phi[1..m] \leftarrow$ CONTROL-POINTS $(\rho_k(p, q), \lceil \tau\lambda \rceil)$
10.                  for $j \leftarrow 1$ to $m - 3$
11.                      DRAW-B-SPLINE $(\phi[j..j + 3], \mathcal{J})$
```

Figure 8.5: Proposed algorithm for removal of digital aberrations in fingerprint ridgelines.

of discontinuity, and storing them in the list $\mathcal{L}$. In Step 5, each point μ_p in $\mathcal{L}$ is considered one by one to extract the ridge segments present in the image in the next step using the procedure of DEPTH-FIRST-SEARCH as discussed earlier. In Step 9, $m (\geq 4)$ control points, namely $\langle \phi_j(p, q) \rangle_{j=1}^{m}$, are defined (for each ridgeline having length at least $L_{\min}$ — verified in Step 8) from $\phi_1(p, q) = \mu_p$ to $\phi_m(p, q) = \mu_q$, using the average inter-ridge distance λ and smoothness parameter τ, which are used in steps 10 and 11 to draw the set of B-splines, one for every four successive points in $\langle \phi_j(p, q) \rangle_{j=1}^{m}$ (Sec. 8.2.1). In Step 9 of the procedure CONTROL-POINTS $(\rho_k(p, q), \lceil \tau\lambda \rceil)$, the argument $\lceil \tau\lambda \rceil$ is used to decide the maximum distance (measured by the number of pixels lying on the ridgeline) between two consecutive control points lying on the ridge $\rho_k(p, q)$.

8.2.4 Approximation Error

Let the input binary fingerprint image $\mathcal{I}$ consist of a set of r ridges, namely $\{\rho_k(u, v)\}_{k=1}^{r}$, where μ_u and μ_v are the uth and the vth point (minutia(e) or point(s) of discontinuity) in the set $\{\mu_i\}_{i=1}^{n}$. Let $L(\rho_k)$ be the digital length of (i.e., number of pixels on) the kth ridge segment ρ_k. Then the Hausdorff distance, denoted by $d_{\mathrm{H}}(\rho_k, \mathcal{B}_k)$, of the ridge segment ρ_k from its corresponding B-spline segment, namely $\mathcal{B}_k$, is a measure for *error of smoothing* (approximating) the ridge ρ_k (by $\mathcal{B}_k$)[1]. Since $d_{\mathrm{H}}(\rho_k, \mathcal{B}_k)$, in effect, is given by

[1]Instead of Hausdorff distance, other measures of correspondence can be used, since Hausdorff distance is noise-sensitive [Klette and Zamperoni (1987)].

the maximum over the distances measured (in terms of average inter-ridge distance, λ, of $\mathcal{I}$) from all points of ρ_k to the corresponding nearest points of $\mathcal{B}_k$, we get

$$d_{\mathrm{H}}\left(\rho_k, \mathcal{B}_k\right) = \frac{1}{\lambda}\left(\max\left\{\min\left\{d_\infty(p,p') : p' \in \mathcal{B}_k\right\} : p \in \rho_k\right\} \right), \tag{8.10}$$

where, the 8-distance (Minkowski norm L_1), $d_\infty(p,p')$, between two points $p := (x,y)$ and $p' := (x',y')$ is given by

$$d_\infty(p,p') = \max(|x-x'|, |y-y'|). \tag{8.11}$$

The maximum error, associated with the spline-correction of an image $\mathcal{I}$ to obtain the image $\mathcal{J}$ is, therefore, given by

$$\xi_{max}(\mathcal{I},\mathcal{J}) = \max_{1\le k\le r} \left\{ d_{\mathrm{H}}\left(\rho_k, \mathcal{B}_k\right) : \rho_k \subset \mathcal{I}, \mathcal{B}_k \subset \mathcal{J} \right\}, \tag{8.12}$$

and the corresponding average error $\xi_{avg}\left(\mathcal{I},\mathcal{J}\right)$ in the output image $\mathcal{J}$, due to approximation by B-spline, would be

$$\xi_{avg}\left(\mathcal{I},\mathcal{J}\right) = \frac{\sum\limits_{k=1}^{r} d_{\mathrm{H}}\left(\rho_k, \mathcal{B}_k\right)}{\sum\limits_{k=1}^{r} L(\rho_k)}. \tag{8.13}$$

For a given database $\mathcal{D}$, consisting of $|\mathcal{D}|$ images, the corresponding *overall maximum error*, $\overline{\xi}_{max}$, and the *overall average error*, $\overline{\xi}_{max}$, are therefore obtained as follows.

$$\overline{\xi}_{max} = \frac{1}{|\mathcal{D}|}\sum_{\mathcal{I}\in\mathcal{D}} \xi_{max}(\mathcal{I},\mathcal{J}) \tag{8.14}$$

$$\overline{\xi}_{avg} = \frac{1}{|\mathcal{D}|}\sum_{\mathcal{I}\in\mathcal{D}} \xi_{avg}(\mathcal{I},\mathcal{J}) \tag{8.15}$$

Using Eqns. 8.14 and 8.15, the standard deviation of maximum errors and that of average errors for the image database $\mathcal{D}$ are given by:

$$\sigma_{max} = \left[\frac{1}{|\mathcal{D}|}\sum_{\mathcal{I}\in\mathcal{D}} \left(\xi_{max}(\mathcal{I},\mathcal{J}) - \overline{\xi}_{max}\right)^2\right]^{\frac{1}{2}} \tag{8.16}$$

$$\sigma_{avg} = \left[\frac{1}{|\mathcal{D}|}\sum_{\mathcal{I}\in\mathcal{D}} \left(\xi_{avg}(\mathcal{I},\mathcal{J}) - \overline{\xi}_{avg}\right)^2\right]^{\frac{1}{2}} \tag{8.17}$$

Table 8.1: Experimental results for four databases of images for $\tau = 2$

Set	Database	$\|\mathcal{D}\|$	Image size	$\overline{n}$	$\overline{\xi}_{max}$	σ_{max}	$\overline{\xi}_{avg}$	σ_{avg}	T
i	NIST sdb-4	50	480×512	127	2.703	1.327	0.411	0.172	0.180
ii	NIST sdb-14	124	480×512	141	2.925	1.184	0.489	0.185	0.207
iii	FVC-2000 B:db-1	80	300×300	73	3.038	1.504	0.396	0.190	0.095
iv	FVC-2000 B:db-2	80	364×256	94	2.982	1.588	0.439	0.199	0.081

$\overline{n}$ = average number of points of discontinuity (minutiae and non-minutiae) for all images in $\mathcal{D}$
T = average CPU time in seconds for all images in $\mathcal{D}$

8.3 Experimental Results

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. We used the fingerprint images from (i) NIST Special Database 4 [Watson and Wilson (1992)], (ii) NIST Special Database 14 [Watson and Wilson (1992)], (iii) Database B1 of FVC2000 [FVC 2000 (2000)], and (iv) Database B2 of FVC2000 [FVC 2000 (2000)]. Each image in these four sets is an 8-bit gray-scale image, recorded at 500 dpi. The images are used after applying Wavelet Scalar Quantization [Bhowmick *et al.* (2005a)], followed by (thinned) ridge extraction proposed by Bishnu *et al.* (2002).

The experimental results on B-spline correction for these four databases are reported in Table 8.1 and some plots on errors associated with B-spline correction are shown in Fig. 8.6 and Fig. 8.7. In Table 8.1, column 3 indicates the number of images considered for generating the results shown. Column 5 indicates the average number of minutiae and non-minutia points of discontinuity for image set $\mathcal{D}$. Column 6 shows the *overall maximum error*, $\overline{\xi}_{max}$, whereas column 8 the *overall average error*, $\overline{\xi}_{avg}$, for the set of images in the corresponding database $\mathcal{D}$ containing $|\mathcal{D}| = N$ images, estimated from the maximum errors (Eqn. 8.12) and the average errors (Eqn. 8.13) of the individual images in $\mathcal{D}$, as given in Eqn. 8.14 and Eqn. 8.15 respectively. The respective standard deviations corresponding to maximum error and average error (Eqns. 8.16 and 8.17) have been shown in columns 7 and 9. The average CPU time for correction of the ridgelines per image of a database is given in column 10.

In Fig. 8.6, we have shown the maximum error $\overline{\xi}_{max}$, and the average error $\overline{\xi}_{avg}$, as explained in Sec. 8.2.4 (Eqns. 8.14 and 8.15). The errors have been plotted against τ (varying from 1/8 to 6) for the images in the databases. It is evident from these plots that

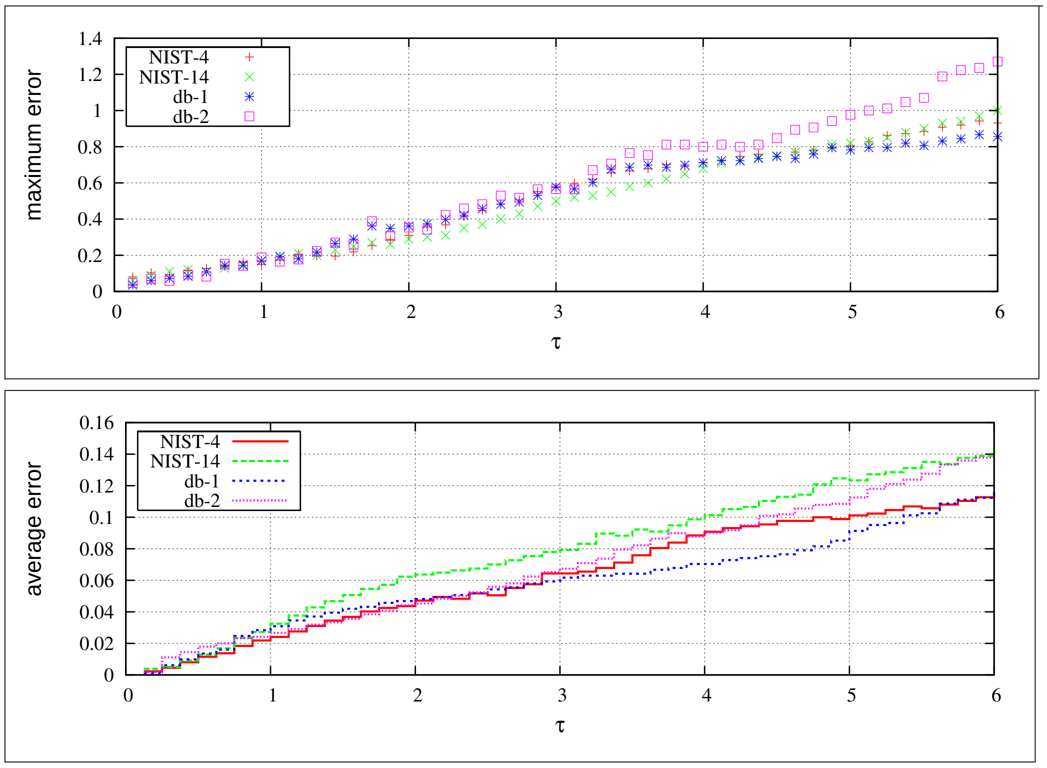
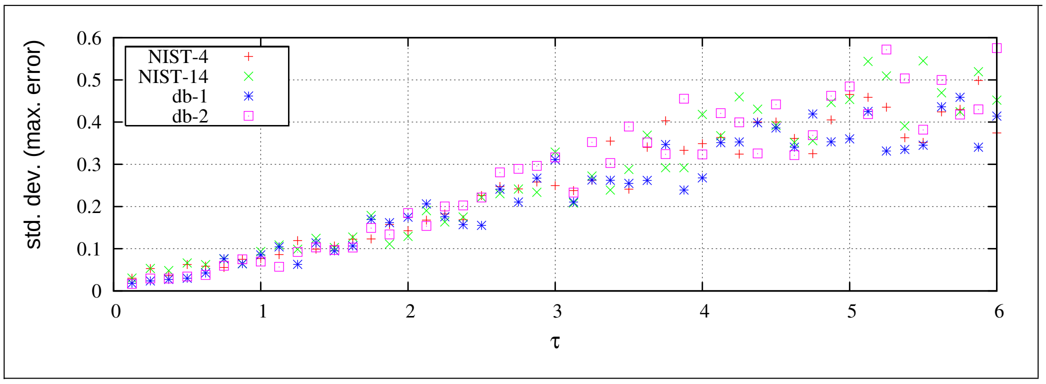


Figure 8.6: The maximum error ($\bar{\xi}_{max}$: Eqn. 8.14) and the average error ($\bar{\xi}_{avg}$: Eqn. 8.15), for the images in each of the four databases (sets (i)–(iv)) from $\tau = 1/8$ to $\tau = 6$. One unit of the vertical (error) axis equals to λ pixel units, where λ = average inter-ridge distance that lies in the interval $[7, 11]$; and a pixel unit means $\frac{1}{500}$th of an inch for each image (500 dpi). The horizontal axis corresponds to τ, and is, therefore, unit-free.

for τ in the range $[1, 2]$, the errors of the spline-corrected ridgelines is appreciably small, which justifies the usefulness of B-spline correction of the uneven ridgelines. Further, as shown Fig. 8.7, the standard deviations of both the maximum error and the average error for τ in the range $[1, 2]$ are also small, an observation that justifies the use of B-splines for smoothing aberrated ridgelines.

To further demonstrate the usefulness of B-spline correction of ridgelines in an AFIS, we have shown some Receiver Operating Characteristic (ROC) curves in Fig. 8.8 by plotting the Authentic Acceptance Rate (AAR) versus the False Acceptance Rate (FAR) for database NIST-4 and database fvc2000: db-1a. It may be observed from these ROC curves that, for an appropriate value of τ (e.g., for $\tau = 2$), the matching performance is better



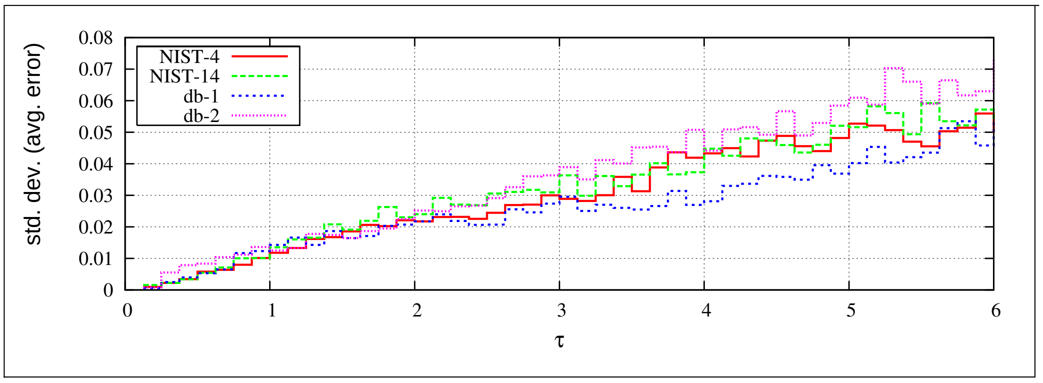
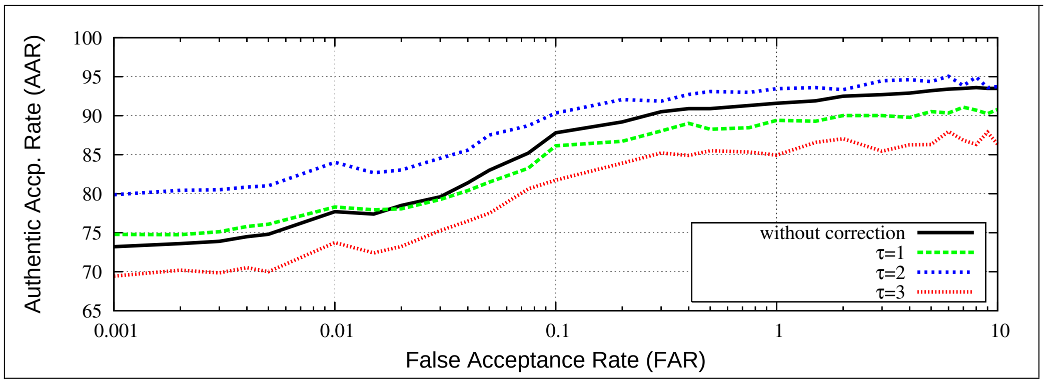


Figure 8.7: The standard deviations of maximum errors (σ_{max}: Eqn. 8.16) and of average errors (σ_{avg}: Eqn. 8.17) for the four databases ($\tau = 1/8 - 6$, axis units same as in Fig. 8.6).

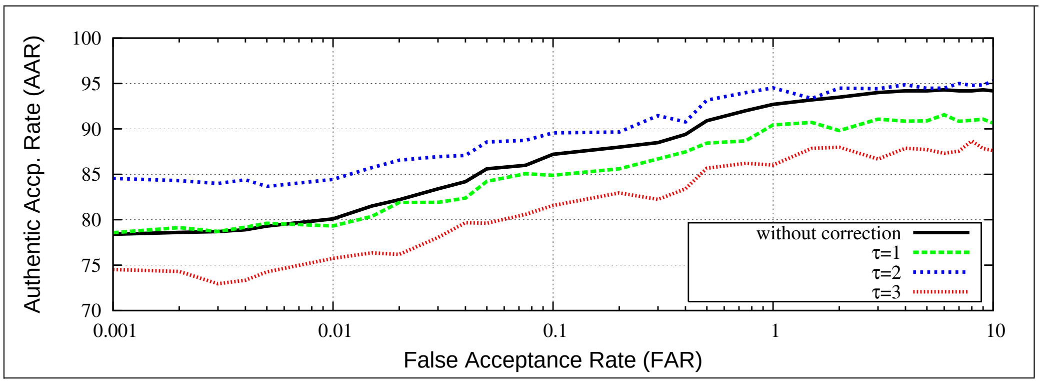
for the fingerprints that have undergone ridgeline corrections (using B-splines as proposed in this work) compared to the same database containing fingerprints with uncorrected ridgelines. Another advantage of the proposed spline correction is that, for a low value of FAR, the AAR is appreciably better than the AAR corresponding to the same FAR without spline correction; this effect is, however, prominent for τ nearing 2.

For τ nearing or exceeding 3, the matching result deteriorates. This is caused by excessive bending of the corrected ridgelines from their original positions, thereby creating false acceptance or false rejection. The fact that a high value of τ creates undue shifting of the ridgelines after B-spline correction, has been explained in Sec. 8.2.2. It is also apparent from the plots of errors and their standard deviations given in Figs. 8.6 and 8.7.

Results on a sample image (db2a_15_1.pgm) in set (iv) have been shown (cropped and magnified) in Fig. 8.9. The original gray-scale image and the subsequent enhanced image are shown in Fig. 8.9(a) and Fig. 8.9(b) respectively, from where the input image, shown in Fig. 8.9(c), is obtained using an earlier method [Bishnu *et al.* (2002)]. The extracted set



NIST-4



FVC2000: db-1a

Figure 8.8: ROC curves of fingerprint matching after B-spline correction with $\tau = 1, 2, 3$ versus those without correction justify the importance of the proposed scheme in an AFIS.

of ridgelines is shown in Fig. 8.9(d), the endpoints of ridgelines being colored in red. The set of control points, shown in red in Fig. 8.9(e), is used to finally obtain the corrected image, shown in Fig. 8.9(f), where the blue color of a ridgeline indicates that it is not corrected in the procedure, since its length does not exceed the requisite length $L_{\min}$ (see Eqn. 8.9).

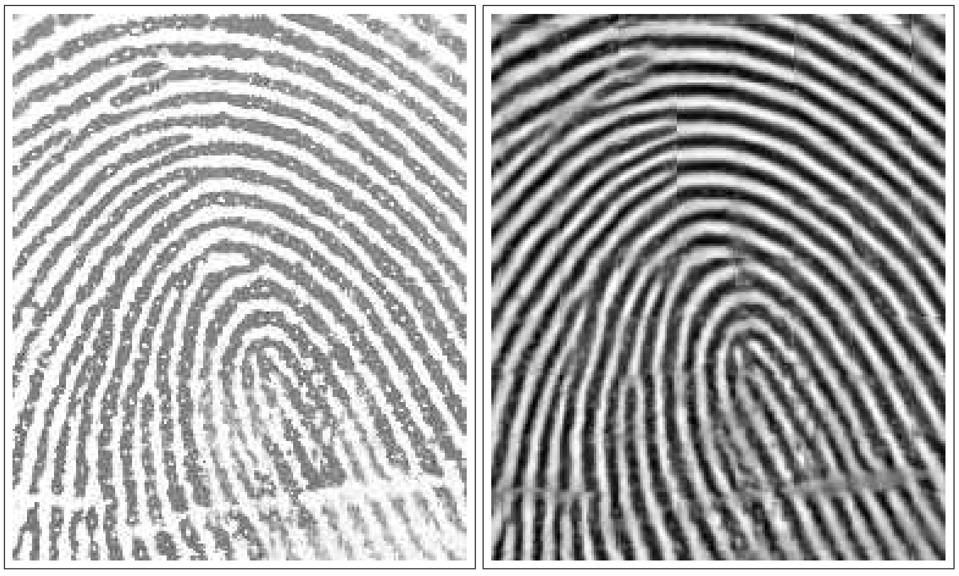
The effect of τ on the degree of smoothing versus the deviation of the corrected ridgelines w.r.t. their original positions has been illustrated by the results shown in Fig. 8.10. The set of B-spline segments (black) versus the corresponding ridge segments (gray) for $\tau = 1, 2, 3, 4$ indicates that a judicious selection of τ is an important prerequisite in the smoothing procedure. A value of τ in the order of $1 - 2$ is likely to produce the desired smoothness of ridgelines with minimal deviation from their original shape and position.

In Fig. 8.11, the binary skeleton before and that after the subsequent correction have been shown to demonstrate the effect of correction on determining the flow pattern of the ridgelines in a fingerprint image.

8.4 Conclusion

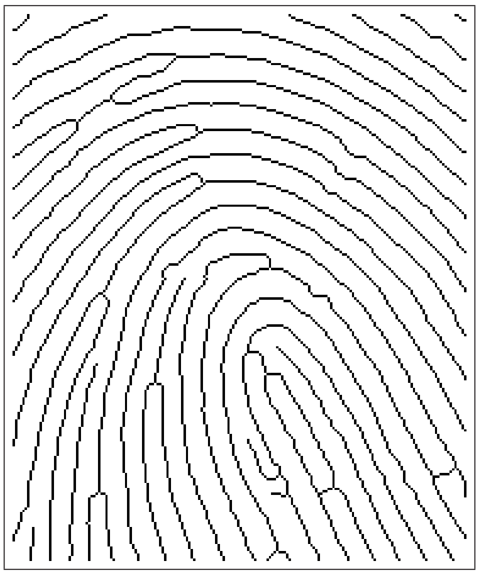
We have proposed here a new technique for removing arbitrary digital aberrations/deformations in the binary ridge topography of a fingerprint image. Previous methods dealing with distortions have sought to prevent distorted fingerprints from being captured or matched, or have allowed increasing tolerance, thereby compromising with the reliability and efficiency of the system. On the contrary, we have designed and tested a method that can actually reduce malformations in previously captured fingerprints, in an automatic unsupervised manner, based on certain logical assumptions about the undistorted fingerprints. The method is fast, robust, efficient, and easy to implement. The corrected ridgelines are smooth, continuous, and preserve the end points.

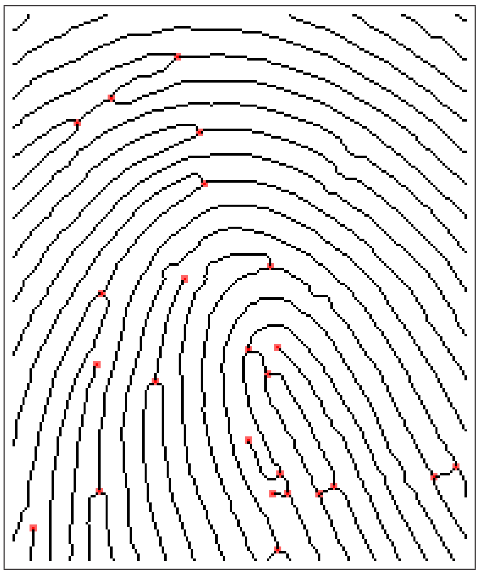
In this work, a set of geometrically and parametrically continuous set of uniform nonrational B-splines is fitted to each section of the digital curve (ridgeline) between two minutia(e) or point(s) of discontinuity. Further improvements of the proposed system may be obtained by inclusion of some other possible aspects, such as (i) correction of inter-ridge spacing; (ii) adaptivity of τ with the local curvature of each ridgeline; etc.

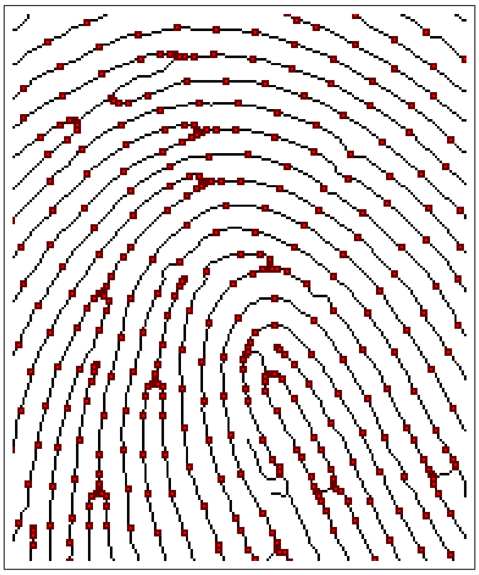


(a) gray-scale (b) enhanced

Figure 8.9: continued to next page.


(c) binary image (**input**)


(d) set of ridge segments


(e) control points

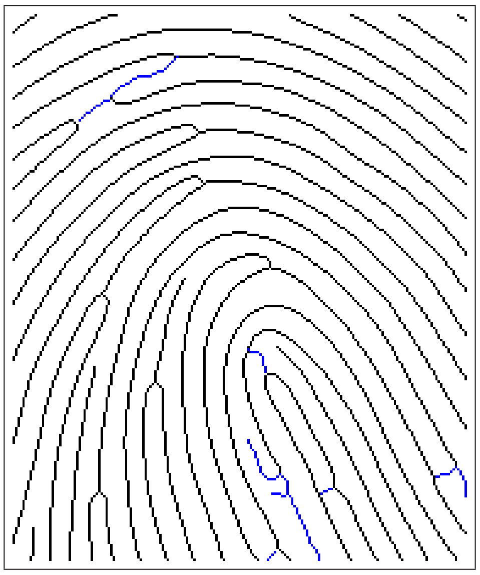
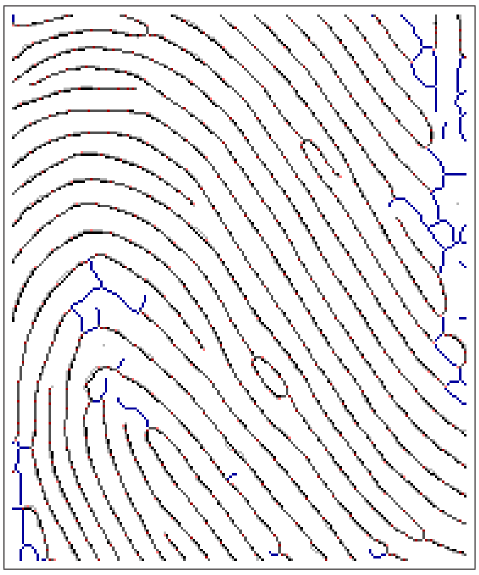
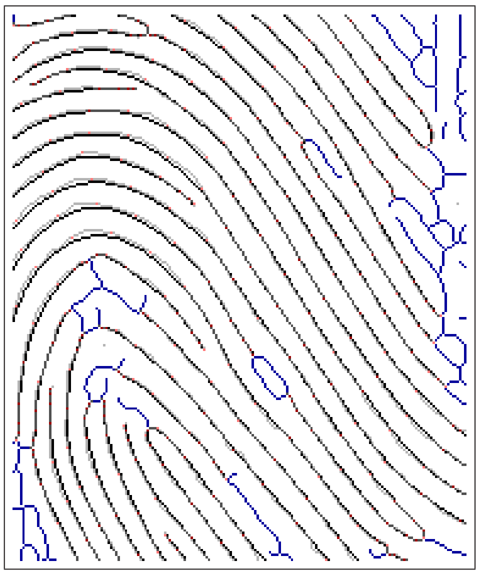
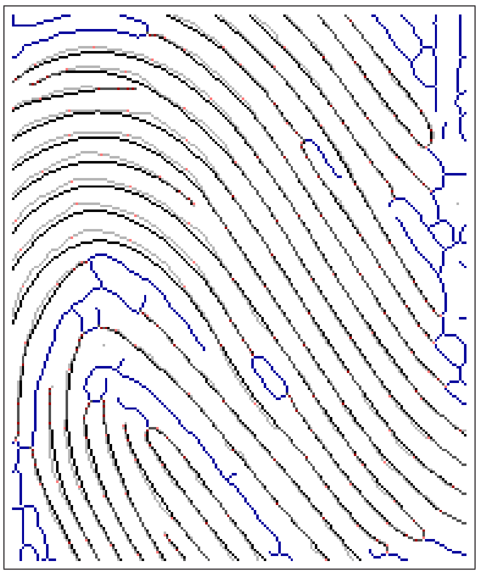

(f) corrected image

Figure 8.9: Results for a sample image (shown in (a)) from set (iv) after its necessary enhancement (shown in (b)). From the **input** binary skeleton (shown in (c)), the ridge segments (shown in (d)) are extracted to obtain the corresponding **output** set of B-spline segments (shown in (f)) using selected control points (shown in (e)) with $\tau = 2$.


(a) $\tau = 1$


(b) $\tau = 2$


(c) $\tau = 3$

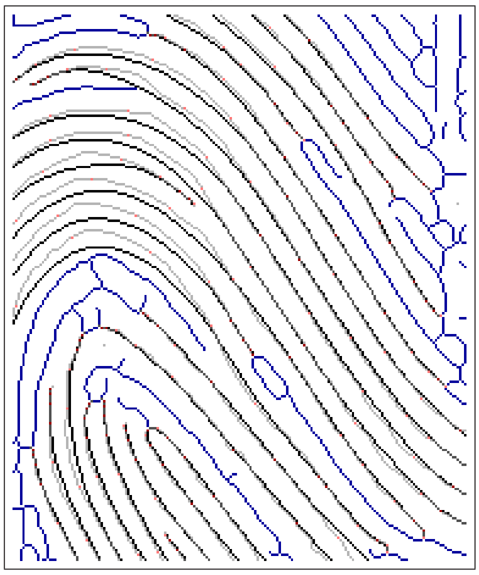
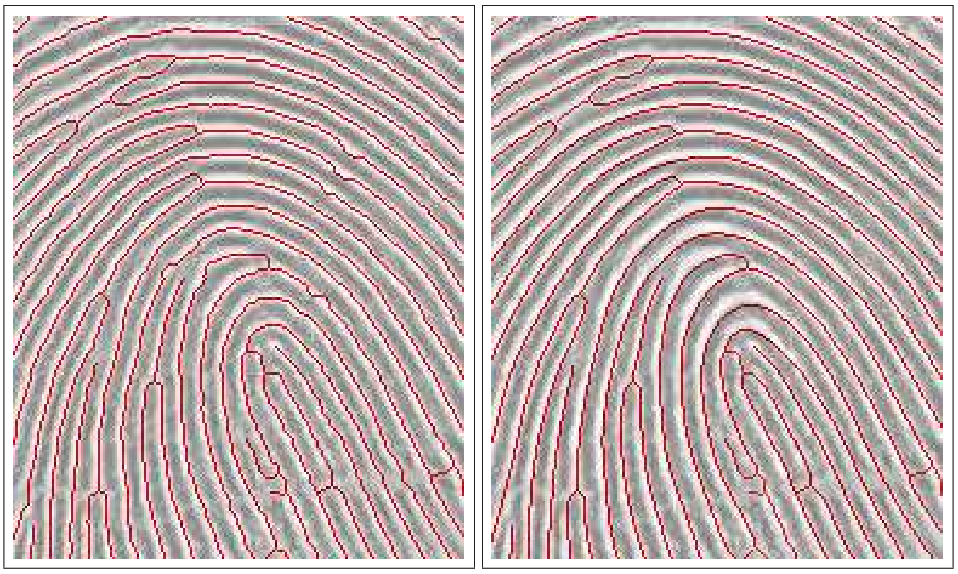

(d) $\tau = 4$

Figure 8.10: The set of B-spline segments (shown in black) versus the corresponding ridge segments (shown in gray) for different values of τ. It is evident from these figures that selection of τ in and around the interval $[1, 2]$ produces smoothly flowing ridgelines with minimal deviation from their original pattern.



(a) before correction (b) after correction

Figure 8.11: Binary skeleton (red) superimposed on the original gray-scale image (shown in Fig. 8.9) testifies that the flow patterns of ridgelines are smoother after correction.

Chapter 9

Conclusion

The whole is more than the sum of its parts.

Aristotle

Metaphysica

Some novel theoretical interpretations and related applications of low-order digital-geometric primitives, such as points, line segments, circles, and cubic curves, have been studied in this thesis. Comparative performances of existing approaches to solving some contemporary problems in the digital plane have been assessed. Appropriate modelings and reformulations have been done to conceive the problems in the digital plane, and some new techniques have been suggested to solve these problems using digital-geometric approaches. Diverse applications of these ideas to biometrics, geometric feature extraction, image matching, curve approximation, and computer graphics have been aptly demonstrated with both theoretical and experimental results.

Combining interdisciplinary paradigms like digital imaging, number theory, computational geometry in general, and digital geometry in particular, we have formulated and strengthened the proposed methods reported in this thesis. The emerging subject of digital geometry, with its immense potential in solving various geometric problems in the digital domain, has found many interesting applications to image analysis, for example, circular range query, approximate point set pattern matching, characterization and approximation of digital circles, approximate digital straight line segments, etc. We speculate that several other areas, for example, medical imaging and diagnostics, are yet to be looked at to enrich the theory and to explore future application domains in the digital space.

There are several open problems relevant to the work undertaken in this thesis. One is the precise formulation of "approximate straightness" w.r.t. a quantified approximation criterion, such as τ as mentioned in Chapter 5. In other words, how a set of properties (similar to [R1–R4] related with DSS) can be derived, which are necessary (and sufficient, preferably) to declare a digital curve as τ-straight in the sense that no grid point lying on that curve is at a distance exceeding τ from a real line that has a correspondence with the concerned curve. Another problem is extending the digital-geometric/number-theoretic interpretation of digital circles [Chapter 6] to explore further properties for extracting circular arcs from a digital image (e.g., an engineering drawing) using simple primitive operations in the digital space. We have already worked out and tested a few more interesting properties in this direction, which would be reported shortly.

Glossary

$*$	operation of convolution,
$\emptyset$	empty set,
$\triangleright$	indicates a comment in an algorithm,
α_r	angular tolerance, 4.2
a, b, a', b'	points in S, 2.4
A	discrete orthogonal domain (minimum-area finite region) containing all points in a set of points, 2.5
$\mathcal{A}$ ($\mathcal{A}_k$)	ordered list of end points of ADSS corresponding to a (kth) digital curve, 5.2
$\mathcal{A}_r^{(x,y)}$	ordered list of image gray values of the pixels lying on a digital circle of radius r centered at (x, y), 4.2
ADSS	approximate digital straight line segment, 5.1
AFIS	Automatic Fingerprint Identification Systems, 8.1
APSPM	Approximate Point Set Pattern Matching, 2.1
β	measure of accuracy of approximation in APSPM, 2.3
b_{ij}	an element of basis matrix $\mathbf{B}$, 8.2
$\mathbf{B}$	basis matrix defining a given B-spline segment, 8.2
c1, c2	conditions of an ADSS, 5.2
$c_{x(0-3)}, c_{y(0-3)}$	elements of coefficient matrix $\mathbf{C}$, 8.2
$\mathbf{C}$	coefficient matrix defining a given B-spline segment, 8.2
$\mathcal{C}_r^{(x,y)}$	ordered list of pixels of the digital circle of radius r centered at (x, y), 4.2
$\mathcal{C}_k$	kth digital curve in a given set, 5.3
$\mathcal{C}^{\mathbb{R}}(p, r)$	real circle with center p and radius r, 6.2, 7.2
$\mathrm{C}_{\max}$	maximum error criterion, 5.3
$\mathrm{C}_{\sum}$	cumulative error criterion, 5.3

$\mathcal{C}^{\mathbb{Z}}(p,r)$	(set of grid points/pixels of) digital circle with center at $p \in \mathbb{Z}^2$ and radius $r \in \mathbb{Z}^+$, 6.2, 7.2
$\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O},r)$	first octant of the full circle $\mathcal{C}^{\mathbb{Z}}(\mathrm{O},r)$, 6.2
CODE	algorithm for detection of COrners and Directions of incident Edges, 4.2
CR	compression ratio, 5.3
δ	distance of nearest grid point from a point on the grid line, 6.2
δ_μ	isothetic distance of grid point μ from $\mathcal{C}^{\mathbb{R}}(\alpha,\rho)$, 2.2, 7.2
d	allowance in variation of run-lengths of non-singular character `n` in an ADSS, 5.2
d	depth of a node in an angular tree, 2.4
$dev_\perp(p \to \widetilde{p})$	isothetic deviation from p to $\widetilde{p}$, 5.3
$d_{\min}$	minimum Euclidean distance between pairs of points in a set of points, 2.5
d_P	a distance (between two points) in P, 2.4
$d_{\perp(\max)}$	maximum isothetic deviation, 5.3
$d(p,q)$	Euclidean distance between points p and q, 2.3
$d_\top(p,q)$	maximum isothetic distance between two points p and q, 5.3
$\mathcal{D}^{\mathbb{R}}(q,r)$	disc lying in the real plane $\mathbb{R}^2$ with center $q \in \mathbb{R}^2$ and radius $r \in \mathbb{R}^+$, 7.2
$\mathcal{D}^{\mathbb{Z}}(\alpha,\rho)$	digital disc with center $\alpha \in \mathbb{Z}^2$ and radius $\rho \in \mathbb{Z}$, 7.2
$D(e_t,u)$	determinant to check the position of point u w.r.t. the directed edge e_t, 2.4
$\overline{D}(\underline{D})$	the set of distances/lengths greater (smaller) than L in S, 2.4
DC	Digital Curve, 5.1
DCR	algorithm to construct Digital Circle using Run length properties, 6.3
DCS	algorithm to construct Digital Circle using Square Numbers, 6.2
DSL	digital straight line/ray, 5.1
DSS	digital straight line segment, 5.1
$\varepsilon, \varepsilon^*$	measures of approximation of an APSPM algorithm, 2.2, 7.2

ε_μ	radial distance of grid point μ from $\mathcal{C}^{\mathbb{R}}(\alpha, \rho)$, 2.2, 7.2
$e(\mathbf{L})$	end point of the line $\mathbf{L}$, 5.3
e_t	edge directed from $v_t \in V_R$ to $v_{t+1} \in V_R$, 2.4
E	pixel east of current pixel, 6.2
E_i	end point of ith ADSS, 5.2
E_{approx}	error incurred by a suboptimal polygonal approximation algorithm, 5.3
E_{opt}	error incurred by an optimal polygonal approximation algorithm, 5.3
EPSPM	Exact Point Set Pattern Matching, 2.1
EXTRACT-ADSS	proposed algorithm for extraction of ADSS from a digital curve, 5.2
$f(\tau, d_\perp)$	see IEF, 5.3
$\mathcal{F}$	gray-scale fingerprint image, 8.2
$\mathcal{F}_r^{(x,y)}$	filtered list of gray values lying on a digital circle of radius r centered at (x, y), 4.2
$\widetilde{\mathcal{F}}_r^{(x,y)}$	reordered gray-values in $\mathcal{F}_r^{(x,y)}$ after cyclic shift φ subject to conditions **c1** and **c2**, 4.2
F1–F3	properties of DSS (Freeman (1961a)), 5.2
FIND-PARAMS	procedure to find the parameters (n, s, l) for an ADSS, 5.2
FOM	figure of merit, 5.3
γ_∞	sole parameter defined by user for adaptive brightness thresholding, 4.2
γ_r	adaptive brightness threshold for radius r, 4.2
$\mathit{\Gamma}$	space of transformations, 2.3
$G(\mathcal{C}^{\mathbb{Z}}(p, r), q)$	enumerated set of grid points lying on the digital circle $\mathcal{C}^{\mathbb{Z}}(p, r)$ with q as the point of reference, 6.2
$h(P, Q)$	Hausdorff distance *from P to Q*, 2.3
$\mathcal{I}$	image/set of digital curves, 5.3, 8.2
$I(i, j)$	gray-scale intensity at the point (i, j) in a digital image I, 4.2

$\nabla I(i,j)$	magnitude of gradient of intensity at a point (i,j) in an image I, 4.2
I_k	k-th interval (Eqn. 6.3), 6.2
IEF	isothetic error frequency (or, error frequency), 5.3
ISE	integral square error, 5.3
$\mathcal{J}$	output binary fingerprint image with smoothed ridgelines, 8.2
k	half the number of edges of a regular query polygon with even number of vertices, 2.4
K	number of digital curves in a given set, 5.3
λ	inter-ridge distance in a fingerprint image, 3.2, 8.2
λ_r	*floor* of half the size of mask $\mathcal{W}_r$, 4.2
λ_ϕ	median line during construction of angular tree, 2.4
$\lambda(j)$	run length of grid points of $\mathcal{C}^{\mathbb{Z},\mathrm{I}}(\mathrm{O},r)$ with ordinate j, 6.3
$\lambda(r-k)$	run length at ordinate $r-k$ in the first octant of digital circle with radius r, 6.2
l	length of leftmost run of the non-singular element in a DSS/ADSS, 5.2
l_k	length of interval I_k (Eqn. 6.4), 6.2
$\mathcal{L}$	list of minutiae/points of discontinuity, 8.2
$\widetilde{\mathbf{L}}$	ADSS formed after merging several ADSS, 5.3
$\mathbf{L}, \mathbf{L}'$	line segments in $\mathbb{R}^2$, 2.4
L, L'	lengths of $\mathbf{L}, \mathbf{L}'$, 2.4
$\mathbf{L}_i^{(k)}$	the ADSS produced by ith repetition of EXTRACT-ADSS algorithm on kth digital curve, 5.2
L_P	lexicographically sorted sequence – angle first, distance second – of points in P, 2.3
$\mathcal{L}(P)$	list containing distances in set of points, P non-increasing order, 2.4
μ	grid point on $\mathcal{C}^{\mathbb{Z}}(\alpha,\rho)$, 2.2, 7.2
μ_i	ith minutia/point of discontinuity, 8.2

$\mu(T(P), Q)$	number of matching points between $T(P)$ and Q, 2.4
m	slope of a line, 5.2
m	number of control points selected on a ridge segment to be smoothened, 8.2
m, n	number of rows and number of columns in a digital image, 4.2
M	total number of vertices in the approximate polygon(s), 5.1
MI	Matching Index, 3.3
M_k	number of vertices in the approximate polygon P_k, 5.3
M_{approx}	number of vertices in the suboptimal polygon(s), 5.3
M_{opt}	number of vertices in the optimal polygon(s), 5.3
MERGE-ADSS	proposed algorithm for polygonal approximation of all ADSS in a digital curve, 5.3
ν	grid point on $\mathcal{C}^{\mathbb{Z}}(\alpha, \rho)$, 2.2, 7.2
ν, ν_1, ν_2	parent node and its left and right child nodes in an angular tree, 2.4
$\mathfrak{n}$	non-singular element in a DSS/ADSS, 5.2
n	number of ADSS detected in a digital curve, 5.3
n	number of minutiae/points of discontinuity in a fingerprint image, 8.2
$\widehat{n}$	number of matching minutiae for two sets of minutiae, 3.3
$n_{\min}$	minimum number of matching vertices for a successful APSPM between sets of points, P and Q, 2.3
N	number of points in the input set of digital curves, 5.1
$N_{d_\perp}$	number of points with deviation $d_\perp$ due to polygonal approximation, 5.3
N_k	number of points/points defining the kth digital curve, 5.3
ω	weight assigned for matching minutiae, 3.3
O	origin of the rectangular coordinate system, 6.2, 7.2
ϕ	angle made by the splitting line with the horizontal line in an angular tree, 2.4
ϕ	angle of gradient of intensity at a point (i, j) in an image I, 4.2

ϕ_i	ith control point to fit B-spline(s), 8.2
ϕ_j	parameter used for error checking in recurrence Eqn. 5.9, 5.2
$\phi_j(p,q)$	jth control point on the ridge segment starting from $\phi_1(p,q) = \mu_p$ and ending at $\phi_m(p,q) = \mu_q$, 8.2
Φ	geometry vector, 8.2
p	lower limit of run-length interval to find an ADSS, 5.2
p_i	ith point in P, 2.3
$p, p', p'', p_1, \ldots$	points in the set P/P', 2.3, 3.2, 6.2
$\widetilde{p}$	deviation of point p due to polygonal approximation, 5.3
$\overline{pq}$	line segment in $\mathbb{R}^2$ joining points p and q, 5.2
P, P'	pattern set (of minutiae, corners, etc.), 2.1, 3.3, 4.2
P_k	the approximate polygon/polychain corresponding to kth digital curve, 5.3
$P^{R,T,S}$	transformed set obtained from set P after rotation (R), translation (T), and scaling (S), 3.3
q	upper limit of run-length interval to find an ADSS, 5.2
$q, q', q_1, \ldots$	points in set Q, 2.3 3.2
Q	database/background set (of minutiae), 2.1, 3.3
ρ	radius of a digital circle, 2.3
$\rho(p,q)$	ridgeline with endpoints μ_p and μ_q, 8.2
r	length of rightmost run of the non-singular element in a DSS/ADSS, 5.2
r	radius of a circle, 4.2 6.2
$r_{\max}$	maximum radius of an annular window, 4.2
$region(\nu)$	2D region corresponding to a node ν of an angular tree, 2.4
R	query (polygonal) region, 2.4
$(\mathbb{R}^2, d)$	2D Euclidean metric space, 2.4
R1–R4	properties of DSS (Rosenfeld (1974)), 5.2
$\Sigma d\theta(p)$	variance of edge directions for $1 \leq r \leq r_{\max}$, 4.2
$\mathtt{s}$	singular element in a DSS/ADSS, 5.2
s	score (in $[1-100]$) associated with a minutia, 3.3

$s(\mathbf{L})$	start point of the line $\mathbf{L}$, 5.3
$s_{\mathrm{r}}, s_{\mathrm{v}}, s_{\mathrm{n}}$	Score components of a minutia due to ridge flow, valley flow, and noise level respectively, 3.4
S	a finite subset of 2D Euclidean metric space, 2.4
S_i	start point of ith ADSS, 5.2
S_n	square number equal to n^2, 6.3
SE	pixel south-east of current pixel, 6.2
SNR	signal-to-noise ratio, 4.3
τ	approximation parameter or error tolerance, 5.3
θ	angle made by the tangent to the corresponding ridge at a minutia, 3.3
θ	internal angle of the query polygon corresponding to $\mathcal{T}_\theta(Q)$, 2.4
θ_u	direction of uth edge at a point p, 4.2
$\theta_{u,r}$	direction of uth edge at p as per the gray value analysis of the annular list with radius r centered at p, 4.2
t	type of a minutia (bifurcation or termination), 3.3
$\mathcal{T}_\theta(Q)$	angular tree defined on the set of points, Q, 2.4
T	transformation on a set of points, 2.3
$T(P)$	transformed set of points, P, 2.3
T, T_1, T_2	time complexities, 2.4
u	parameter defining a given B-spline segment, 8.2
u_i	ith knot point in a set of B-splines, 8.2
u_k	lower limit of interval I_k (Eqn. 6.6), 6.2
$\mathbf{U}$	parameter vector needed to define a B-spline segment, 8.2
ϱ	threshold parameter in the proposed APSPM algorithm, 2.4
v_k	upper limit of interval I_k (Eqn. 6.6), 6.2
v_t	tth vertex in V_R, 2.4
V_R	ordered set of vertices of query region R, 2.4
$\mathcal{W}_r$	mask of size $2\lambda_r + 1$, 4.2

(x, y)	coordinates of a point or minutia, 3.3, 5.2
x_j, y_j	coordinates of (the control point) ϕ_j, 8.2
(x_p, y_p)	coordinates of a point P, 5.2
$\mathbf{x}_{\mathsf{PQ}}$	projection of line segment $\overline{\mathsf{PQ}}$ on x-axis, 5.2
$\mathbf{y}_{\mathsf{PQ}}$	projection of line segment $\overline{\mathsf{PQ}}$ on y-axis, 5.2

Bibliography[*]

Acharya, T., Bhattacharya, B. B., Bhowmick, P., Bishnu, A., Biswas, A., Kundu, M. K., Murthy, C. A., Das, S. and Nandy, S. C. (2003a). Minutia matching using scoring techniques. US Patent pending (Publication Application # 20050129293, Dec 2003).

Acharya, T., Bhattacharya, B. B., Bhowmick, P., Bishnu, A., Dey, J., Kundu, M. K. and Murthy, C. A. (2003b). Method and apparatus to reduce false minutiae from a binary fingerprint image. US Patent pending (Publication Application # 20030063782, April 2003).

Acharya, T., Bhattacharya, B. B., Bhowmick, P., Bishnu, A., Dey, J., Kundu, M. K. and Murthy, C. A. (2004). Architecture for processing fingerprint images. US Patent No. 6795592.

Acharya, T., Bhattacharya, B. B., Bhowmick, P., Bishnu, A., Dey, J., Kundu, M. K. and Murthy, C. A. (2006). Method and apparatus for providing a binary fingerprint image. US Patent No. 7136515.

Agarwal, P. K. and Erickson, J. (1998). Geometric range searching and its relatives. In B. Chazelle, J. Goodman and R. Pollack, eds., *Advances in Discrete and Computational Geometry*, American Mathematical Society, Providence, 1–56.

Aken, J. R. V. and Novak, M. (1985). Curve-drawing algorithms for raster display. *ACM Trans. Graphics*, **4**, 147–169.

Alkaabi, S. and Deravi, F. (2004). Candidate pruning for fast corner detection. *Electronic Letters*, **40**, 18–19.

Alt, H. and Guibas, L. J. (1996). Discrete geometric shapes: Matching, interpolation, and approximation — A survey. Report B 96-11, Freie Universität, Berlin.

Alt, H., Mehlhorn, K., Wagener, H. and Welzel, E. (1988). Congruence, similarity and symmetries of geometric objects. *Discrete Comput. Geom.*, **3**, 237–256.

[*]**Reference Style:** Authors (year). Title. *Journal/Proceedings name*, volume, pages.

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 Costa, L. d. F. (1996). Multiscale shape analysis using the continuous wavelet transform. In *Proc. Intl. Conf. Image Processing (ICIP),* IEEE CS Press, 291–294.

Argyros, A. A., Bekris, K. E. and Orphanoudakis, S. C. (2001). Robot homing based on corner tracking in a sequence of panoramic images. In *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition (CVPR)*, 3–10.

Arkin, E. M., Kedmen, K., Mitchell, J. S. B., Sprinzak, J. and Werman (1992). Matching points into pair-wise disjoint noise regions: Combinatorials bounds and algorithms. *ORSA J. Comput.*, **4**, 375–386.

Asano, T. and Katoh, N. (1993). Number theory helps line detection in digital images. In *Proc. 4th Intl. Symposium on Algorithms and Computation (ISAAC)*, vol. 762 of *LNCS*, Springer, Berlin, 313–322.

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., Ito, H., Kimura, S. and Shimazu, S. (1998). Repairing flaws in a picture based on a geometric representation of a digital image. In *Proc. 9th Intl. Symposium on Algorithms and Computation (ISAAC)*, vol. 1533 of *LNCS*, Springer, Berlin, 149–158.

Asano, T., Kawamura, Y., Klette, R. and Obokkata, K. (2000). A new approximation scheme for digital objects and curve length estimations. In *Proc. Image Vision Computing New Zealand*, 26–31.

Asano, T., Kawamura, Y., Klette, R. and Obokata, K. (2001). Minimum-length polygons in approximation sausages. In *Proc. 4th Intl. Workshop on Visual Form*, vol. 2059 of *LNCS*, Springer, Berlin, 103–112.

Asano, T., Kawamura, Y., Klette, R. and Obokkata, K. (2003a). Digital curve approximation with length evaluation. *IEICE Trans. Fundamentals of Electronics, Communication and Computer Sciences*, **E86-A**, 987–994.

Asano, T., Klette, R. and Ronse, C., eds. (2003b). *Geometry, Morphology, and Computational Imaging*, vol. 2616 of *LNCS*. Springer, Berlin.

Atkinson, M. D. (1998). An optional algorithm for geometric congruence. *J. Algorithms*, **8**, 159–172.

Attneave, F. (1954). Some informational aspects of visual perception. *Psychological Review*, **61**, 183–193.

Badler, N. I. (1977). Disk generators for a raster display device. *Computer Graphics and Image Processing*, **6**, 589–593.

Baird, H. S. (1985). Model-based image matching using location. MIT Press, Distinguished Dissertation Series.

Banerjee, M., Kundu, M. K. and Mitra, P. (2004). Corner detection using support vector machine. In *Proc. 17th Intl. Conf. Pattern Recognition (ICPR),* IEEE CS Press, 819–822.

Banerjee, S., Mukherjee, D. P. and Dutta Majumder, D. (1995). Point landmarks for registration of CT and MR images. *Pattern Recognition Letters*, **16**, 1033–1042.

Barnard, S. T. and Thomson, W. B. (1980). Disparity analysis of images. *IEEE Trans. PAMI*, **2**, 333–340.

Bazen, A. M. and Gerez, S. H. (2003). Fingerprint matching by thin-plate spline modeling of elastic deformations. *Pattern Recognition*, **36**, 1859–1867.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, **18**, 509–517.

Bentley, J. L. (1979). Multidimensional binary search trees used in database applications. *IEEE Trans. Software Engineering*, **4**, 333–340.

Berg, M. D., Kreveld, M. V., Overmars, M. and Schwarzkopf, O. (2000). *Computational Geometry Algorithms and Applications*. Springer-Verlag, Berlin.

Bertrand, G., Imiya, A. and Klette, R., eds. (2001). *Digital and Image Geometry: Advanced Lectures*, vol. 2243 of *LNCS*. Springer, Berlin.

Beus, H. L. and Tiu, S. H. (1987). An improved corner detection algorithm based on chain-coded plane curves. *Pattern Recognition*, **20**, 291–296.

Bezdek, J. C. and Anderson, I. M. (1985). An application of the *c*-varieties clustering algorithms to polygonal curve fitting. *IEEE Trans. Sys., Man & Cybern.*, **15**, 637–641.

Bhanu, B. and Tan, X. (2003). Fingerprint indexing based on novel features of minutiae triplets. *IEEE Trans. PAMI*, **25**, 616–622.

Bhowmick, P. and Bhattacharya, B. B. (2004a). Approximate fingerprint matching using Kd-tree. In *Proc. 17th Intl. Conf. Pattern Recognition (ICPR),* IEEE CS Press, vol. 1, 544–547.

Bhowmick, P. and Bhattacharya, B. B. (2004b). **CODE**: An adaptive algorithm for detecting **Co**rners and **D**irections of incident **E**dges. In *Proc. Indian Conf. Computer Vision, Graphics and Image Processing (ICVGIP)*, Allied Publishers Pvt. Ltd., New Delhi, 509–515.

Bhowmick, P. and Bhattacharya, B. B. (2005a). Approximation of digital circles by regular polygons. In *Proc. Intl. Conf. Advances in Pattern Recognition (ICAPR)*, vol. 3686 of *LNCS*, Springer, Berlin, 257–267.

Bhowmick, P. and Bhattacharya, B. B. (2005b). Augmentation of corners by directions of incident edges. Indian Statistical Institute, Kolkata, India, Technical Report No. ACMU/BBB/2005/1.

Bhowmick, P. and Bhattacharya, B. B. (2006a). Number theoretic interpretation and construction of a digital circle. *Discrete Applied Mathematics* (under revision).

Bhowmick, P. and Bhattacharya, B. B. (2006b). Removal of digital aberrations in fingerprint ridgelines using B-splines. *Pattern Recognition* (under revision).

Bhowmick, P. and Bhattacharya, B. B. (2007a). Approximate matching of digital point sets using a novel angular tree. *IEEE Trans. PAMI* (doi.ieeecomputersociety.org/10.1109/TPAMI.2007.70812).

Bhowmick, P. and Bhattacharya, B. B. (2007b). Fast polygonal approximation of digital curves using relaxed straightness properties. *IEEE Trans. PAMI*, **29**, 1590–1602.

Bhowmick, P., Bishnu, A., Bhattacharya, B. B., Kundu, M. K., Murthy, C. A. and Acharya, T. (2002). Determination of minutiae scores for fingerprint image applications. In *Proc. Indian Conf. Computer Vision, Graphics and Image Processing (ICVGIP)*, Allied Publishers Pvt. Ltd., New Delhi, 463–468.

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 *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). ICE: the Isothetic Convex Envelope of a digital object. In *Proc. Intl. Conf. Computing: Theory and Applications (ICCTA)*, IEEE CS Press, 219–223.

Bhowmick, P., Biswas, A. and Bhattacharya, B. B. (2007c). Ranking of optical character prototypes using cellular lengths. In *Proc. Intl. Conf. Computing: Theory and Applications (ICCTA)*, IEEE CS Press, 422–426.

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., DAS, S., NANDY, S. C. AND BHATTACHARYA, B. B. (2003). An improved algorithm for point set pattern matching under rigid motion. In *Proc. 5th Italian Conf. Algorithms and Complexity*, vol. 2653 of *LNCS*, Springer, Berlin, 36–45.

BISHNU, A., BHOWMICK, P., BHATTACHARYA, B. B., KUNDU, M. K., MURTHY, C. A. AND ACHARYA, T. (2006a). A combinatorial approach to ridge extraction in a gray-scale fingerprint image and its hardware implementation. *ISI Platinum Jubilee Book (World Scientific)*.

BISHNU, A., DAS, S., NANDY, S. C. AND BHATTACHARYA, B. B. (2006b). 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). Reconstruction of torn documents using contour maps. In *Proc. Intl. Conf. Image Processing (ICIP),* IEEE CS Press, 517–520.

BISWAS, A., BHOWMICK, P. AND BHATTACHARYA, B. B. (2005c). **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, 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, S. N. and Chaudhuri, B. B. (1985). On the generation of discrete circular objects and their properties. *Computer Vision, Graphics, and Image Processing*, **32**, 158–170.

Blinn, J. F. (1987). How many ways can you draw a circle? *IEEE Computer Graphics and Applications*, **7**, 39–44.

Blisset, R. J. (1990). Retrieving 3D information from video for robot control and surveillance. *Electr. Commun. Eng. J.*, 155–163.

Bresenham, J. E. (1965). Algorithm for for computer control of a digital plotter. *IBM Systems Journal*, **4**, 25–30.

Bresenham, J. E. (1977). A linear algorithm for incremental digital display of circular arcs. *Communications of the ACM*, **20**, 100–106.

Bresenham, J. E. (1985). Run length slice algorithm for incremental lines. In R.A. Earnshaw, ed., *Fundamental Algorithms for Computer Graphics*, vol. F17 of *NATO ASI Series*, Springer-Verlag, New York, 59–104.

Burden, M. J. J. and Bell, M. G. H. (1997). Vehicle classification using stereo vision. In *Proc. 6th Intl. Conf. Image Processing and Its Applications*, vol. 2, 881–885.

Buvaneswari, A. and Naidu, P. S. (1998). Estimation of shape of binary polygonal object from scattered field. *IEEE Trans. Image Processing*, **7**, 253–257.

Candela, G. T., Grother, P. J., Watson, C. I., Wilkinson, R. A. and Wilson, C. L. (1995). *PCASYS — A Pattern-Level Classification Automation System for Fingerprints, NISTIR 5647*. National Institute of Standards and Technology.

Cecelja, F., Balachandran, W. and Jirawimut, R. (2003). A stereo vision system for pedestrian navigation. In *Proc. 17th Intl. Conf. Applied Electromagnetics and Communications*, 30–33.

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, 347–350.

Chan, Y. T. and Thomas, S. M. (1995). Cramer-Rao lower bounds for estimation of a circular arc center and its radius. *Graphical Models and Image Processing*, **57**, 527–532.

Chanda, B. and Dutta Majumder, D. (1999). *Digital Image Processing and Applications*. Prentice-Hall of India, New Delhi.

Chattopadhyay, S., Das, P. P. and Ghosh-Dastidar, D. (1994). Reconstruction of a digital circle. *Pattern Recognition*, **27**, 1663–1676.

Chen, T. C. and Chung, K. L. (2001a). An efficient randomized algorithm for detecting circles. *Computer Vision and Image Understanding*, **83**, 172–191.

Chen, T. C. and Chung, K. L. (2001b). A new randomized algorithm for detecting lines. *Real Time Imaging*, **7**, 473–481.

Chikkerur, S., Cartwright, A. N. and Govindaraju, V. (2007). Fingerprint enhancement using STFT analysis. *Pattern Recognition*, **40**, 198–211.

Chiu, S. H. and Liaw, J. J. (2005). An effective voting method for circle detection. *Pattern Recognition Letters*, **26**, 121–133.

Chung, W. L. (1977). On circle generation algorithms. *Computer Graphics and Image Processing*, **6**, 196–198.

Climer, S. and Bhatia, S. K. (2003). Local lines: A linear time line detector. *Pattern Recognition Letters*, **24**, 2291–2300.

Coeurjolly, D., Gérard, Y., Reveillès, J. P. and Tougne, L. (2004). An elementary algorithm for digital arc segmentation. *Discrete Applied Mathematics*, **139**, 31–50.

Coifman, B., Beymer, D., McLauchlan, P. and Malik, J. (1998). A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research: Part C*, **6**, 271–288.

Cormen, T. H., Leiserson, C. E. and Rivest, R. L. (2000). *Introduction to Algorithms*. Prentice Hall of India, New-Delhi.

Creutzburg, E., Hübler, A. and Wedler, V. (1982). On-line recognition of digital straight line segments. In *Proc. 2nd Intl. Conf. AI and Inf. Control Systems of Robots*, 42–46.

Danielsson, P. E. (1978). Comments on circle generator for display devices. *Computer Graphics and Image Processing*, **7**, 300–301.

Das, A. K. and Chanda, B. (2001). A fast algorithm for skew detection of document images using morphology. *IJDAR*, **4**, 109–114.

Davies, E. R. (1987). A high speed algorithm for circular object detection. *Pattern Recognition Letters*, **6**, 323–333.

Davies, E. R. (1988). A hybrid sequential-parallel approach to accurate circle centre location. *Pattern Recognition Letters*, **7**, 279–290.

Davis, L. S., Rosenfeld, A. and Agrawala, A. K. (1976). On models for line detection. *IEEE Trans. Sys., Man & Cybern.*, **6**, 127–133.

de Rezende, P. J. and Lee, D. T. (1995). Point set pattern matching in d-dimensions. *Algorithmica*, **13**, 387–404.

Debled-Rennesson, I. and Reveilles, J. P. (1995). A linear algorithm for segmentation of digital curves. *Intl. J. Patt. Rec. Artif. Intell.*, **9**, 635–662.

Debled-Rennesson, I., Jean-Luc, R. and Rouyer-Degli, J. (2003). Segmentation of discrete curves into fuzzy segments. In *Ninth Intl. Workshop on Combinatorial Image Analysis*, vol. 12 of *Electronic Notes in Discrete Mathematics*, 372–383.

Dorai, C., Ratha, N. and Bolle, R. (2000). Detecting dynamic behavior in compressed fingerprint videos. In *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition (CVPR)*, 320–326.

Doros, M. (1979). Algorithms for generation of discrete circles, rings, and disks. *Computer Graphics and Image Processing*, **10**, 366–371.

Doros, M. (1984). On some properties of the generation of discrete circular arcs on a square grid. *Computer Vision, Graphics, and Image Processing*, **28**, 377–383.

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.

Elias, R. and Laganiére, R. (2002). Cones: A new approach towards corner detection. In *Proc. IEEE Canadian Conf. Electrical & Computer Engg*, 912–916.

Erafat, A. and Itai, A. (1996). Improvements on bottleneck matching and related problems using geometry. In *Proc. 12th Annual ACM Symposium on Computational Geometry*, 301–310.

Eric, W. and Grimson, L. (1990). *Object Recognition by Computer: The Role of Geometric Constraints*. MIT Press.

Etou, Y., Sugiyama, T., Abe, K. and Abe, T. (2002). Corner detection using slit rotational edge-feature detector. In *Proc. Intl. Conf. Image Processing (ICIP),* IEEE CS Press, 797–800.

Faires, J. D. and Burden, R. (1998). *Numerical Methods*. Brooks/Cole Publishing Company, USA.

Farina, A., Kovács-Vajna, Z. M. and Leone, A. (1999). Fingerprint minutiae extraction from skeletonized binary images. *Pattern Recognition*, **32**, 877–889.

Field, D. (1986). Algorithms for drawing anti-aliased circles and ellipses. *Computer Vision, Graphics, and Image Processing*, **33**, 1–15.

Finn, P. W., Kavraki, L. E., Latombe, J., Motwani, R., Shelton, C. R., Venkatasubramanian, S. and Yao, A. (1997). RAPID: Randomized Pharmacophore Identification for Drug design. In *Symposium on Computational Geometry*, 324–333.

Fischler, M. A. and Wolf, H. C. (1994). Locating perceptually salient points on planar curves. *IEEE Trans. PAMI*, **16**, 113–129.

Foley, J. D., Dam, A. v., Feiner, S. K. and Hughes, J. F. (1993). *Computer Graphics — Principles and Practice*. Addison-Wesley, Reading (Mass.).

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.

Galton, F. (1892). *Fingerprints*. Macmillan, London.

Gao, Y. (2004). Single model face database retrieval by directional corner points. In *Proc. 10th Intl. Conf. Multimedia Modeling*, 10–16.

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.

Germain, R. S., Califano, A. and Colville, S. (1997). Fingerprint matching using transformation parameter clustering. *IEEE Computational Sc. and Engg.*, **4**, 42–49.

Gleicher, M. (1999). Animation from observation: Motion capture and motion editing. *Computer Graphics*, **33**, 51–55.

Gonzalez, R. C. and Woods, R. E. (1993). *Digital Image Processing*. Addison-Wesley, California.

Goodrich, M. T., Mitchel, J. B. and Orletsky, M. W. (1994). Practical methods for approximate geometric pattern matching under rigid motion. In *Proc. 10th Annual ACM Symposium on Computational Geometry*, 103–113.

Gu, Y. H. and Tjahjadi, T. (1999). Corner-based feature extraction for object retrieval. In *Proc. Intl. Conf. Image Processing (ICIP),* IEEE CS Press, 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.

Hagedoorn, M. and Veltkamp, R. C. (1999). Reliable and efficient pattern matching using an affine invariant metric. *Intl. J. Computer Vision*, **31**, 203–225.

Haralick, R. (1983). Ridges and valleys on digital images. *Computer Vision, Graphics, and Image Processing*, **22**, 28–38.

Haralick, R. M. (1974). A measure for circularity of digital figures. *IEEE Trans. Sys., Man & Cybern.*, **4**, 394–396.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Proc. 4th Alvey Vision Conf.*, 147–151.

He, Y., Tian, J., Luo, X. and Zhang, T. (2003). Image enhancement and minutiae matching in fingerprint verification. *Pattern Recognition Letters*, **24**, 1359–1370.

Heffernan, P. J. and Schirra, S. (1994). Approximate decision algorithms for point set congruence. *Computational Geometry: Theory and Applications*, **4**, 137–156.

Held, A., Abe, K. and Arcelli, C. (1994). Towards a hierarchical contour description via dominant point detection. *IEEE Trans. Sys., Man & Cybern.*, **24**, 942–949.

Ho, C. T. and Chen, L. H. (1995). A fast ellipse/circle detector using geometric symmetry. *Pattern Recognition*, **28**, 117–124.

Hoffmann, F., Kriegel, K. and Wenk, C. (1999). An applied point pattern matching problem: Comparing 2D patterns of protein spots. *Discrete Applied Mathematics*, **93**, 75–88.

Holm, L. and Sander, C. (1996). Mapping the protein universe. *Science*, **273**, 595–602.

Horn, B. K. P. (1976). Circle generators for display devices. *Computer Graphics and Image Processing*, **5**, 280–288.

Hosur, P. I. and Ma, K. K. (1999). A novel scheme for progressive polygon approximation of shape contours. In *Proc. IEEE 3rd Workshop on Multimedia Signal Processing*, 309–314.

Hrechak, A. K. and McHugh, J. (1990). Automated fingerprint recognition using structural matching. *Pattern Recognition*, **23**.

Hsieh, C. T., Lai, E. and Wang, Y. C. (2003). An effective algorithm for fingerprint image enhancement based on wavelet transform. *Pattern Recognition*, **36**, 303–312.

Hsu, S. Y., Chow, L. R. and Liu, C. H. (1993). A new approach for the generation of circles. *Computer Graphics Forum 12*, **2**, 105–109.

Hu, J. and Yan, H. (1997). Polygonal approximation of digital curves based on the principles of perceptual organization. *Pattern Recognition*, **30**, 701–718.

Hung, D. C. D. (1993). Enhancement and feature purification of fingerprint images. *Pattern Recognition*, **26**, 1661–1671.

Huttenlocher, D. P. and Rucklidge, W. T. (1993). A multi-resolution technique for comparing images using the Hausdorff distance. In *Proc. Intl. Conf. Computer Vision and Pattern Recognition (CVPR),* IEEE CS Press, 705–706.

Huttenlocher, D. P., Kedem, K. and Kleinberg, J. M. (1992). On dynamic Voronoi diagrams and the minimum Hausdorff distance for point sets under Euclidean motion in the plane. In *Proc. 8th Annual ACM Symposium on Computational Geometry*, 110–120.

Imai, H. and Iri, M. (1986). Computational geometric methods for polygonal approximations of a curve. *Computer Vision, Graphics, and Image Processing*, **36**, 31–41.

Indyk, P., Motwani, R. and Venkatasubramanian, S. (1999). Geometric matching under noise: Combinatorial bounds and algorithms. In *10th Annual SIAM-ACM Symposium of Discrete Algorithms (SODA)*, 457–465.

Jain, A. K., Hong, L. and Bolle, R. (1997). On-line fingerprint verification. *IEEE Trans. PAMI*, **19**, 302–313.

Jain, A. K., Prabhakar, S., Hong, L. and Pankanti, S. (2000). Filterbank-based fingerprint matching. *IEEE Trans. Image Processing*, **9**, 846–859.

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, 282–285.

Jiang, X. and Ser, W. (2002). Online fingerprint template improvement. *IEEE Trans. PAMI*, **24**, 1121–1126.

Jiang, X. and Yau, W. Y. (2000). Fingerprint minutiae matching based on the local and global structures. In *Proc. 15th Intl. Conf. Pattern Recognition (ICPR),* IEEE CS Press, 1042–1045.

Jiang, X., Yau, W. Y. and Ser, W. (1999). Minutiae extraction by adaptive tracing the gray level ridge of the fingerprint image. In *Proc. Intl. Conf. Image Processing (ICIP),* IEEE CS Press, 852–856.

Jobbágy, A., Furnée, E., Romhányi, B., L.Gyöngy and Soós, G. (1999). Resolution and accuracy of passive marker-based motion analysis. *Automatika*, **40**, 25–29.

Kawamura, Y. (2003). Approximation of planar objects in digital images with guaranteed accuracy. Japan Advanced Institute of Science and Technology, doctoral Thesis.

Kim, H. S. and Kim, J. H. (2001). A two-step circle detection algorithm from the intersecting chords. *Pattern Recognition Letters*, **22**, 787–798.

Kitchen, L. and Rosenfeld, A. (1982). Gray-level corner detection. *Pattern Recognition Letters*, **1**, 95–102.

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. (1983). On the approximation of convex hulls of finite grid point sets. *Pattern Recognition Letters*, **2**, 19–22.

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. (2004). Errata for Digital Geometry, first printing. http://www.citr.auckland.ac.nz/~rklette/Books/MK2004/ErrataText.htm.

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.

Klette, R. and Žunić, J. (1999). Digital approximation of moments of convex regions. *Graphical Models Image Processing*, **61**, 274–298.

Klette, R. and Žunić, J. (2000). Interactions between number theory and image analysis. In L.J. Latecki, D.M. Mount and A.Y. Wu, eds., *Proc. SPIE vol. 4117, Vision Geometry IX*, 210–221.

Klette, R. and Zamperoni, P. (1987). Measures of correspondence between binary patterns. *Image Vision Comput.*, **5**, 287–295.

Klette, R., Stojmenović, I. and Žunić, J. (1996). A parametrization of digital planes by least square fits and generalizations. *Graphical Models Image Processing*, **58**, 295–300.

Klette, R., Rosenfeld, A. and Sloboda, F., eds. (1998). *Advances in Digital and Computational Geometry*. Springer, Singapore.

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.

Koplowitz, J. and Plante, S. (1995). Corner detection for chain coded curves. *Pattern Recognition*, **28**, 843–852.

Koplowitz, J., Lindenbaum, M. and Bruckstein, A. (1990). The number of digital straight lines on an $n \times n$ grid. *IEEE Trans. Information Theory*, **36**, 192–197.

Kovács-Vajna, Z. M. (2000). A fingerprint verification system based on triangular matching and dynamic time warping. *IEEE Trans. PAMI*, **22**, 1266–1276.

Kovács-Vajna, Z. M., Rovatti, R. and Frazzoni, M. (2000). Fingerprint ridge distance computation methodologies. *Pattern Recognition*, **33**, 69–80.

Kovalevsky, V. A. (1990). New definition and fast recognition of digital straight segments and arcs. In *Proc. 10th Intl. Conf. Pattern Recognition (ICPR),* IEEE CS Press, 31–34.

Kulpa, Z. (1979). A note on "circle generator for display devices". *Computer Graphics and Image Processing*, **9**, 102–103.

Kulpa, Z. and Kruse, B. (1983). Algorithms for circular propagation in discrete images. *Computer Vision, Graphics, and Image Processing*, **24**, 305–328.

Langsam, Y., Augenstein, M. J. and Tenenbaum, A. M. (2000). *Data Structurs using C and C++*. Prentice-Hall of India, New Delhi.

Lüdtke, N., Luo, B., Hancock, E. and Wilson, R. C. (2002). Corner detection using a mixture model of orientation. In *Proc. 16th Intl. Conf. Pattern Recognition (ICPR),* IEEE CS Press, 574–577.

LEE, J. S., SUN, Y. N. AND CHEN, C. H. (1995). Multiscale corner detection by using wavelet transform. *IEEE Trans. Image Processing*, **4**, 100–104.

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.

LI, F. AND KLETTE, R. (2007). Analysis of the rubberband algorithm. *Image Vision Comput.*, **25**, 1588–1598.

LIANG, R., CHEN, C. AND BU, J. (2003). Real-time facial features tracker with motion estimation and feedback. In *Proc. IEEE Intl. Conf. Systems, Man and Cybernetics*, 3744–3749.

LIKAR, B. AND PERNUS, F. (1999). Automatic extraction of corresponding points for the registration of medical images. *Med. Phys.*, **26**, 1678–1686.

LIN, C. H., LIU, J. H., OSTENBURG, J. W. AND NICOL, J. D. (1982). Fingerprint comparison I: Similarity of fingerprints. *Journal of Forensic Sciences*, **27**, 290–304.

LINDEBERG, T. (1994). Junction detection with automatic selection of detection scales and localization scales. In *Proc. Intl. Conf. Image Processing (ICIP),* IEEE CS Press, 924–928.

LIU, H. C. AND SRINATH, M. D. (1990). Corner detection from chain-code. *Pattern Recognition*, **23**, 51–68.

LUO, X., TIAN, J. AND WU, Y. (2000). A minutia matching algorithm in fingerprint verification. In *Proc. 15th Intl. Conf. Pattern Recognition (ICPR),* IEEE CS Press, 833–836.

MAINTZ, J. AND VIERGEVER, M. (1998). A survey of medical image registration. *IEEE Engineering in Medicine and Biology Magazine*, **2**, 1–36.

MAIO, D. AND MALTONI, D. (1997). Direct gray-scale minutiae detection in fingerprints. *IEEE Trans. PAMI*, **19**, 27–39.

MALTONI, D., MAIO, D., JAIN, A. K. AND PRABHAKAR, S. (2003). *Handbook of Fingerprint Recognition.* Springer-Verlag, New York.

Manjunath, B. S., Shekhar, C., Chellappa, R. and von der Malsburg, C. (1992). A robust method for detecting image features with application to face recognition and motion correspondence. In *Proc. 11th Intl. Conf. Pattern Recognition Methodology and Systems*, 208–212.

McCane, B., Galvin, B. and Novins, K. (2002). Algorithmic fusion for more robust feature tracking. *Intl. Journal Computer Vision*, **49**, 79–89.

McIlroy, M. D. (1983). Best approximate circles on integer grids. *ACM Trans. Graphics*, **2**, 237–263.

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.

Mehtre, B. M. and Murthy, N. N. (1986). A minutia based fingerprint identification system. In *Proc. 2nd Intl. Conf. Advances in Patt. Recogn. and Digital Techniques, Calcutta*.

Mehtre, B. M., Murthy, N. N., Kapoor, S. and Chatterjee, B. (1987). Segmentation of fingerprint images using directional image. *Pattern Recognition*, **20**, 429–435.

Mieghem, J. A. V., Avi-Itzhak, H. I. and Melen, R. D. (1995). Straight line extraction using iterative total least squares methods. *J. Visual Commun. and Image Representation*, **6**, 59–68.

Mignosi, F. (1991). On the number of factors of Sturmian words. *Theoretical Computer Science*, **82**, 71–84.

Mohanna, E. and Mokhtarian, E. (2003). Robust corner tracking for unconstrained motions. In *IEEE Intl. Conf. Acoustics, Speech, and Signal Processing*, 804–807.

Mohanna, F. and Mokhtarian, F. (2002). Robust corner tracking for multimedia applications. In *Proc. Intl. Conf. Image Processing (ICIP),* IEEE CS Press, 945–948.

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. and Suomela, R. (1998). Robust image corner detection through curvature scale space. *IEEE Trans. PAMI*, **20**, 1376–1381.

Mount, D., Netanyahu, N. and Lemoigne, J. (1998). Improved algorithms for robust point pattern matching and applications to image registration. In *Proc. 14th Annual ACM Symposium on Computational Geometry*, 155–164.

Nagel, H. H. (1987). On the estimation of optical flow: Relations between different approaches and some new results. *Artificial Intelligence*, **33**, 298–324.

Nagy, B. (2004). Characterization of digital circles in triangular grid. *Pattern Recognition Letters*, **25**, 1231–1242.

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.

O'Gorman, L. and Nickerson, J. V. (1989). An approach to fingerprint filter design. *Pattern Recognition*, **22**, 29–38.

Panek, J. and Vohradsky, J. (1999). Point pattern matching in the analysis of two-dimensional gel electropherograms. *Electrophoresis*, **20**, 3483–3491.

Park, J. S. and Han, J. H. (1997). Estimating optical flow by tracking contours. *Pattern Recognition Letters*, **18**, 641–648.

Parui, S. K., Sarkar, S. and Chaudhuri, B. B. (1993). Computing the shape of a point set in digital images. *Pattern Recognition Letters*, **14**, 89–94.

Pavlidis, I., Singh, R. and Papanikolopoulos, N. P. (1997). An on-line handwritten note recognition method using shape metamorphosis. In *4th Intl. Conf. Document Analysis and Recognition (ICDAR)*, 914–918.

Pavlidis, T. (1977). *Structural Pattern Recognition*. Springer, New York.

Pavlidis, T. (1980). Algorithms for shape analysis and waveforms. *IEEE Trans. PAMI*, **2**, 301–312.

Perez, J. C. and Vidal, E. (1994). Optimum polygonal approximation of digitized curves. *Pattern Recognition Letters*, **15**, 743–750.

Pernus, F., Kovacic, S. and Gyergyek, L. (1980). Minutiae-based fingerprint recognition. In *Proc. 5th Intl. Conf. Pattern Recognition (ICPR),* IEEE CS Press, 1380–1382.

Pitrelli, J. F., Subrahmonia, J. and Perrone, M. P. (2006). Confidence modeling for handwriting recognition: Algorithms and applications. *IJDAR*, **8**, 35–46.

Pitteway, M. L. V. (1974). Integer circles, etc. — Some further thoughts. *Computer Graphics and Image Processing*, **3**, 262–265.

Pla, F. (1996). Recognition of partial circular shapes from segmented contours. *Computer Vision and Image Understanding*, **63**, 334–343.

Plamondon, R. and Srihari, S. N. (2000). On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. PAMI*, **22**, 63–84.

Povazan, I. and Uher, L. (1998). The structure of digital straight line segments and Euclid's algorithm. In *Proc. Spring Conf. Computer Graphics*, 205–209.

Prabhakar, S., Jain, A. K. and Pankanti, S. (2003). Learning fingerprint minutiae location and type. *Pattern Recognition*, **36**, 1847–1857.

Project Group: ISI-Intel (2002a). MinuBin: Minutiae extraction from a skeletonized BINary fingerprint image. Indian Statistical Institute, Kolkata, India, Technical Report No. ACMU/INTEL/2002/1.

Project Group: ISI-Intel (2002b). RiVEx: Extraction of RIdge and Valley skeletons from a gray-scale fingerprint image. Indian Statistical Institute, Kolkata, India, Technical Report No. ACMU/INTEL/2002/2.

Rangarajan, K., Shah, M. and Brackle, D. V. (1989). Optimal corner detection. *Computer Vision, Graphics, and Image Processing*, **48**, 230–245.

Ratha, N. K. and Bolle, R. N. (1998). Effect of control acquisition on fingerprint matching. In *Proc. 14th Intl. Conf. Pattern Recognition (ICPR),* IEEE CS Press, 1959–1961.

Ratha, N. K., Chen, S. and Jain, A. K. (1995). Adaptive flow orientation based feature extraction in fingerprint images. *Pattern Recognition*, **28**, 1657–1672.

Ratha, N. K., Karu, K., Chen, S. and Jain, A. K. (1996). A real time matching system for large fingerprint databases. *IEEE Trans. PAMI*, **18**, 799–813.

Richards, J. (1999). The measurement of human motion: A comparison of commercially available systems. *Human Movement Science*, **18**, 589–602.

Rocha, J. and Bernardino, R. (1998). Singularities and regularities on line pictures via symmetrical trapezoids. *IEEE Trans. PAMI*, **20**, 391–395.

Rockett, P. I. (2003). Performance assessment of feature detection algorithms: A methodology and case study on corner detectors. *IEEE Trans. Image Processing*, **12**, 1668–1676.

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. (2000). Shape partitioning by convexity. *IEEE Trans. Sys., Man & 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. (1988). Detection of circular arcs in images. In *Proc. 4th. Alvey Vision Conf., Manchester*, 259–263.

Rosin, P. L. and West, G. A. W. (1995). Non-parametric segmentation of curves into various representations. *IEEE Trans. PAMI*, **17**, 1140–1153.

Ross, A., Dass, S. C. and Jain, A. K. (2006). Fingerprint warping using ridge curve correspondences. *IEEE Trans. PAMI*, **28**, 19–30.

S.-Yuan, C. and W.-Hsiang, T. (1991). Determination of robot locations by common object shapes. *IEEE Trans. Robotics and Automation*, **7**, 149–156.

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, 811–814.

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.

Shanks, D. (1993). *Solved and Unsolved Problems in Number Theory*. AMS Chelsea Publishing, New York.

Shimizu, K. (1981). Algorithm for generating a digital circle on a triangular grid. *Computer Graphics and Image Processing*, **15**, 401–402.

Smeulders, A. W. M. and Dorst, L. (1991). Decomposition of discrete curves into piecewise segments in linear time. *Contemporary Math.*, **119**, 169–195.

Smith, S. M. and Brady, J. M. (1995). ASSET-2: Real-time motion segmentation and shape tracking. *IEEE Trans. PAMI*, **17**, 814–820.

Smith, S. M. and Brady, J. M. (1997). SUSAN — A new approach to low level image processing. *Intl. J. Comput. Vision*, **23**, 45–78.

Suenaga, Y., Kamae, T. and Kobayashi, T. (1979). A high speed algorithm for the generation of straight lines and circular arcs. *IEEE Trans. Comput.*, **28**, 728–736.

Szekely, I. (1997). Crossing numbers and hard Erdös problems in discrete geometry. *Combinatorics, Probability, and Computing*, **6**, 36–47.

Teh, C. H. and Chin, R. T. (1989). On the detection of dominant points on digital curves. *IEEE Trans. PAMI*, **2**, 859–872.

Thebaud, L. R. (1999). Systems and methods with identity verification by comparison and interpretation of skin patterns such as fingerprints. US Patent 5,909,501.

Thomas, S. M. and Chan, Y. T. (1989). A simple approach for the estimation of circular arc center and its radius. *Computer Vision, Graphics, and Image Processing*, **45**, 362–370.

Tian, G., Gledhill, D. and Taylor, D. (2003). Comprehensive interest points based imaging mosaic. *Pattern Recognition Letters*, **24**, 1171–1179.

Tsay, T. I. J., Hsu, M. S. and Lin, R. X. (2003). Development of a mobile robot for visually guided handling of material. In *Proc. IEEE Intl. Conf. Robotics and Automation*, 3397–3402.

Urdiales, C., Trazegnies, C., Bandera, A. and Sandoval, F. (2003). Corner detection based on adaptively filtered curvature function. *Electronic Letters*, **39**, 426–428.

Ventura, J. A. and Chen, J. M. (1992). Segmentation of two-dimensional curve contours. *Pattern Recognition*, **25**, 1129–1140.

Voss, K. (1991). Coding of digital straight lines by continued fractions. *Comput. Artif. Intelligence*, **10**, 75–80.

Wahab, A., Chin, S. H. and Tan, E. C. (1998). Novel approach to automated fingerprint recognition. *Vision, Image & Signal Processing*, **145**, 160–166.

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, Z., Rao, K. R. and Ben-Arie, J. (1996). Optimal ramp edge detection using expansion matching. *IEEE Trans. PAMI*, **18**, 1092–1097.

Watson, C. I. and Wilson, C. L. (1992). *Fingerprint Database, Special Database 4, FPDB*. National Institute of Standards and Technology.

Wegstein, J. H. (1982). *An Automated Fingerprint Identification System*. US Government Publication, Washington.

Williams, J. and Bennamoun, M. (2001). Simultaneous registration of multiple corresponding point sets. *Computer Vision and Image Understanding*, **81**, 117–142.

Willis, A. J. and Myers, L. (2001). A cost-effective fingerprint recognition system for use with low-quality prints and damaged fingertips. *Pattern Recognition*, **34**, 255–270.

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, vol. 4, 234–237.

Worring, M. and Smeulders, A. W. M. (1995). Digitized circular arcs: Characterization and parameter estimation. *IEEE Trans. PAMI*, **17**, 587–598.

Wright, W. E. (1990). Parallelization of Bresenham's line and circle algorithms. *IEEE Computer Graphics and Applications*, **10**, 60–67.

Wu, L. D. (1984). A piecewise linear approximation based on a statistical model. *IEEE Trans. PAMI*, **6**, 41–45.

Wu, X. and Rokne, J. G. (1987). Double-step incremental generation of lines and circles. *Computer Vision, Graphics, and Image Processing*, **37**, 331–344.

Wuescher, D. M. and Boyer, K. L. (1991). Robust contour decomposition using a constant curvature criterion. *IEEE Trans. PAMI*, **13**, 41–51.

Xiao, Q. and Raafat, H. (1991). Fingerprint image postprocessing: A combined statistical and structural approach. *Pattern Recognition*, **24**, 985–992.

Xie, Y. and Ji, Q. (2001). Effective line detection with error propagation. In *Proc. Intl. Conf. Image Processing (ICIP),* IEEE CS Press, 181–184.

Yang, J., Liu, L., Jiang, T. and Fan, Y. (2003). A modified Gabor filter design method for fingerprint image enhancement. *Pattern Recognition Letters*, **24**, 1805–1817.

Yao, C. and Rokne, J. G. (1995). Hybrid scan-conversion of circles. *IEEE Trans. Visualization and Computer Graphics*, **1**, 311–318.

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.

Yuen, P. C. and Feng, G. C. (1996). A novel method for parameter estimation of digital arc. *Pattern Recognition Letters*, **17**, 929–938.

Zang, Q. and Klette, R. (2003). Evaluation of an adaptive composite Gaussian model in video surveillance. In *Proc. 10th Intl. Conf. Computer Analysis of Images and Patterns (CAIP)*, 165–172.

Zhang, D., Peng, H., Zhou, J. and Pal, S. K. (2002). A novel face recognition system using hybrid neural and dual eigenfaces methods. *IEEE Trans. SMC (Pt. A)*, **32**, 787–793.

Zhang, D., Kong, W., You, J. and Wong, M. (2003). On-line palmprint identification. *IEEE Trans. PAMI*, **25**, 1041–1050.

Zhang, D., Lu, G., Kong, W. K. and Wong, M. (2005). Online palmprint authentication system for civil applications. *J. Computers Science and Technology*, **20**, 70–76.

Zheng, Z., Wang, H. and Teoh, E. K. (1999). Analysis of gray-level corner detection. *Pattern Recognition Letters*, **20**, 149–162.

Zhu, H., Tang, X. L. and Liu, P. (2006). An MLP-orthogonal Gaussian mixture model hybrid model for Chinese bank check printed numeral recognition. *IJDAR*, **8**, 27–34.

Author's Statement

The thesis is based on some of the patents and publications of Partha Bhowmick, which are listed below[a].

[a]**Reference Style:** Authors. Title. *Journal/Proceedings name*, volume (:number), pages, year.

Journal Publications

1. P. Bhowmick and B. B. Bhattacharya. Fast polygonal approximation of digital curves using relaxed straightness properties. *IEEE Trans. PAMI*, **29**:9, 1590–1602, 2007.

2. P. Bhowmick and B. B. Bhattacharya. Approximate matching of digital point sets using a novel angular tree. *IEEE Trans. PAMI* (accepted), doi: 10.1109/TPAMI.2007.70812, 2007.

3. P. Bhowmick and B. B. Bhattacharya. Number-theoretic interpretation and construction of a digital circle. *Discrete Applied Mathematics* (accepted), doi: 10.1016/j.dam.2007.10.022, 2007.

4. P. Bhowmick, A. Bishnu, B. B. Bhattacharya, M. K. Kundu, C. A. Murthy, and T. Acharya. Determination of minutiae scores for fingerprint image applications. *Intl. J. Image and Graphics*, **5**, 1–35, 2005.

5. P. Bhowmick and B. B. Bhattacharya. Removal of digital aberrations in fingerprint ridgelines using B-splines. *Pattern Recognition* (under revision), 2006.

6. A. Bishnu, P. Bhowmick, B. B. Bhattacharya, M. K. Kundu, C. A. Murthy, and T. Acharya. A combinatorial approach to ridge extraction in a gray-scale fingerprint image and its hardware implementation. (communicated, January 2007).

Conference Publications

1. 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.

2. 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.

3. 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, pages 219–223, 2007.

4. 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, pages 422–426, 2007.

5. 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, pages 590–594, 2007.

6. 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*, pages 299–310, 2006.

7. P. Bhowmick and B. B. Bhattacharya. Approximate fingerprint matching using Kd-tree. In *Proc. 17th Intl. Conf. Pattern Recognition (ICPR),* IEEE CS Press, volume 1, pages 544–547, 2004.

8. P. Bhowmick and B. B. Bhattacharya. **CODE**: An adaptive algorithm for detecting **Co**rners and **D**irections of incident **E**dges. In *Proc. Indian Conf. Computer Vision, Graphics and Image Processing (ICVGIP)*, Allied Publishers Pvt. Ltd., New Delhi, pages 509–515, 2004.

9. P. Bhowmick and B. B. Bhattacharya. Approximation of digital circles by regular polygons. In *Proc. Intl. Conf. Advances in Pattern Recognition (ICAPR)*, volume 3686 of *LNCS*, pages 257–267, 2005.

10. 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*, pages 407–412, 2005.

11. 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.

12. 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, pages 517–520, 2005.

13. 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*, pages 930–939, 2005.

14. 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.

15. P. Bhowmick, A. Bishnu, B. B. Bhattacharya, M. K. Kundu, C. A. Murthy, and T. Acharya. Determination of minutiae scores for fingerprint image applications. In *Proc. Indian Conf. Computer Vision, Graphics and Image Processing (ICVGIP)*, Allied Publishers Pvt. Ltd., New Delhi, pages 463–468, 2002.

16. A. Bishnu, P. Bhowmick, S. Dey, B. B. Bhattacharya, M. K. Kundu, C. A. Murthy, and T. Acharya. 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, pages 451–456, 2002.

Patents

1. T. Acharya, B. B. Bhattacharya, P. Bhowmick, A. Bishnu, J. Dey, M. K. Kundu, and C. A. Murthy. Method and apparatus for providing a binary fingerprint image. US Patent No. 7136515, 2006.

2. T. Acharya, B. B. Bhattacharya, P. Bhowmick, A. Bishnu, J. Dey, M. K. Kundu, and C. A. Murthy. Architecture for processing fingerprint images. US Patent No. 6795592, 2004.

3. 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. US Patent pending (Publication Application # 20050129293, December 2003).

4. T. Acharya, B. B. Bhattacharya, P. Bhowmick, A. Bishnu, J. Dey, M. K. Kundu, and C. A. Murthy. Method and apparatus to reduce false minutiae from a binary fingerprint image. US Patent pending (Publication Application # 20030063782, April 2003).

Technical Reports

1. P. Bhowmick and B. B. Bhattacharya. Augmentation of corners by directions of incident edges. Indian Statistical Institute, Kolkata, India, 2005. Technical Report No. ACMU/BBB/2005/1.

2. Project Group: ISI-Intel. MinuBin: Minutiae extraction from a skeletonized BINary fingerprint image. Indian Statistical Institute, Kolkata, India, 2002. Technical Report No. ACMU/INTEL/2002/1.

3. Project Group: ISI-Intel. RiVEx: EXtraction of RIdge and Valley skeletons from a gray-scale fingerprint image. Indian Statistical Institute, Kolkata, India, 2002. Technical Report No. ACMU/INTEL/2002/2.