Advances in Computational Geometry for Document Analysis

Godfried Toussaint *
School of Computer Science
McGill University
Montréal, Québec, Canada

Abstract

Many problems in document image analysis can be couched in geometric terms. We outline recent advances in computational geometry that contribute to many aspects of the document analysis process and we provide pointers to a selection of the computational geometry literature where the most relevant results can be found.

1 Introduction

Document image analysis (DIA) is concerned with the automatic transfer by machine of visual two-dimensional documents, most commonly consisting of printed pages from books, magazines or newspapers. Maps and engineering drawings constitute another class of common documents. The first class of problems have much in common with optical character recognition (OCR) and both with computer vision. On the other hand DIA is a special case of computer vision and therefore its special properties give rise to special sub-problems such as text-block isolation and textline-orientation inference. Furthermore these special properties allow the tailoring of more general computer vision strategies resulting in better performance. Not surprizingly computer vision has much to offer to document analysis. For an earlier collection of surveys concerning document analysis and OCR the reader is referred to the July 1992 special issue of the IEEE Proceedings. For a most useful cross-section of the evolution of DIA during the past twenty years see the survey by George Nagy [45]. Many problems in DIA have a geometric flavor and it is also not suprising that computational geometry has much to contribute to DIA. A survey of computational geometric techniques with promise for DIA was presented in 1994 in Las Vegas [60]. In this paper we concentrate on updating the most relevant developments in this area that have occurred since 1994. Accordingly, we assume readers have the Las

*This research was supported by NSERC and FCAR. e-mail: godfried@cs.mcgill.ca

Vegas [60] paper at their disposal and we minimize duplication of background material here.

The publication of material on computational geometry has finally mushroomed during the past decade. Here we briefly outline some of the most relevant sources where the newly developed tools of the trade can be obtained. The classic computational geometry text by Preparata and Shamos has a third edition [51]. While this book does not have the latest results, it is a useful reference for the standard basics. Since 1994 a string of new excellent books have been published. These will certainly be very useful to practitioners of ducument analysis research. There is the very readable introduction to the field by Joseph O'Rourke [46] which includes downloadable *C* code for many of the algorithms. For more advanced textbooks that emphasize applications see [15] and [5]. Janos Pach has edited a collection of chapters written by several different authors on the most esoteric new trends in computational geometry [49]. This book is quite theoretical however and it is not clear how much impact this material will have in the short run on the field of DIA. Another more recent collection of chapters which should be more useful for DIA was edited by Chazelle, Goodman and Pollack [10]. Two massive handbooks on computational geometry have also appeared which are excellent reference books to have on the shelf [52] and [31]. In recent years there has been an effort by the computational geometry community to concentrate more on *applications* of computational geometry instead of pure theory. As a result there are increasingly frequent conferences with resulting proceedings on applied topics [41]. The original *Algorithm Design Manual* of Steve Skiena with a CD included for demonstrations on many browsers also emphasises applications of computational geometry [55]. Finally we mention that since 1994 there has appeared a large collection of excellent surveys in both general and specific areas of computational geometry. For examples of general surveys see [40], and [44]. On the other hand, a survey of the most recent results in computational geometry most relevant for solving pattern recognition problems, which are themselves intimately related to DIA, has appeared as a chapter in one of the handbooks [48]. Another computational geometry topic most relevant to DIA is the *range searching* problem. This general problem includes, as a special case, nearest neighbor searching, which appears in a variety of document analysis problems such as nearest neighbor decision rules for character recognition. For a recent, detailed and exhaustive survey of this field see the survey paper by Pankaj Agarwal and Jeff Erickson [1].

2 Textline Orientation Estimation

When scanning a document for input into the computer, it is likely that the document is not perfectly aligned at its borders. The document may in fact be a skewed copy already. Furthermore the text may in fact have been written in an up-down fashion rather than a left-right manner [54]. The problem of text-line orientation consists of taking as input the connected components of the characters in a block of text (usually the output of the image segmentation stage) and determining the precise location of the lines of text. This information permits, at the next stage, the acquisition of the characters in the correct order for passing on to the character recognition program. The standard approach to solving this problem is to convert each connected component to a representative point such as the center of the smallest orthogonal rectangle that contains the connected component. The block of text is thus reduced to a set of points and it is desired to determine the subsets of almost-collinear points which will serve as estimators of the lines of text.

The traditional method of determing collinear sets of points in computer vision is via the Hough transform [20]. Duda and Hart [20] first proposed this method for scene analysis rather than document analysis and their histogram counts failed to take into account the underlying distribution of the data and noise. Cohen and Toussaint [13] subsequently showed how to optimize the Hough transform to take the window shape and noise into account. Since then hundreds of papers have appeared that propose variations and improvements for a variety of applications. Most of the approaches are semi-brute force in the sense that they perform histogram counts and then detect histogram peaks in some parameter space. However, completely different computational geometric approaches based on progressive peeling of convex hulls have also appeared [16], [17]. Several new more original variants of the Hough transform have recently been proposed by Asano and Katoh [4]. While these techniques may prove useful for some problems in DIA it seems doubtful that they will prove useful for text-line orientation. Most likely they will not compete with the new computationally efficient and robust *proximity graph* approach introduced by Ittner [35]. The fundamental property of text exploited by Ittner's approach is that two adjacent characters are closer together when they belong to the same line of text than if they come from different lines of text. The exploitation of this property is achieved by computing the *minimum spanning tree* of the points that represent the connected components and then estimating the average orientation of the edges in this tree. This orientation is then the estimated orientation of the text line. Several improvements on the computational complexity and the resulting average running time of Ittner's algorithm were presented in [60]. Since then there have also been improvements on the robustness of the procedure through several modifications of Ittner's algorithm. It must be stated that Ittner's algorithm is extremely robust and hard to improve. However, one weakness in the algorithm is that since the proximity graph computed on the points is a tree (minimum spanning tree), and a tree is a connected graph, it folows by necessity that there will be some edges between characters in different lines of text. For example, in this PostScript document the first character of a sentence is often closer to the character directly above it than to the previous character. These spurious edges will have orientations which are completely off the text line direction and therefore are a form of noise adversely affecting the robustness of the procedure. The new methods address this issue by using different proximity graphs that avoid edges between the lines of text. Pateras and Toussaint [50] proposed a proximity graph they dubbed a *bamboo-field*. A bamboo-field is obtained from the minimum spanning tree by removing all vertices (and their incident edges) of degree three or higher. This leaves a collection of essentially parallel chains (bamboo field). These chains are subsequently used to estimate the text-line orientation. Experiments have demonstrated that this vertex-removal operation deletes almost all spurious edges from the minimum spanning tree. In fact each chain corresponds pretty much to one line of text. Experiments comparing this approach to Ittner's method showed that using bamboo-fields improves the robustness of the technique [50]. More recently this approach was improved even further. The bamboo-field method does not prune all unwanted edges from the minimum spanning tree. Some edges remain, especially at the beginnings and ends of the lines of text. Furthermore, by deleting all the edges incident to vertices of degree three or higher, some good edges are also discarded along with the bad. Hence, in [50] there is a balance that has to be maintained. Bose, Caron and Ghoudi [6] proposed a method to prevent the good edges from being discarded during the pruning of the minimum spanning tree. They discard all pairs of adjacent edges in the tree that have an internal angle less than some critical angle. Experimentally they determined this angle to be 170 degrees. Furthermore, extensive experimentation showed that their technique is superior to the bamboo-field method. Their algorithm appears to be the best all-around algorithm available to the practicioner of DIA.

3 Polygonal Approximation

In the vectorized approach to shape analysis for optical character recognition (OCR) we obtain the polygonal boundary of the pixel-based connected components in the image and simplify it by removing as many vertices of the polygon as possible with as little distortion of the shape as possible. The idea is to reduce the size of the input to more complicated algorithms that will be applied later on. Here again is an area where computational geometry has great potential and is playing an ever increasing role. Given a polygonal planar curve $P = (p_1, p_2, ..., p_n)$ the polygonal approximation problem can be cast in many different molds. One such version for example calls for determining a new curve $P' = (p'_1, p'_2, ..., p'_m)$ such that, (1) $m < n$, (2) the p'_i are a subset of the p_i, and (3) any line segment $[p'_j, p'_{j+1}]$ which substitutes the chain corresponding to $[p_r, ..., p_s]$ in P is such that the distance between every p_k for k between r and s and the approximating line segment $[p'_j, p'_{j+1}]$ is less than some prespecified error tolerance.

One of the oldest algorithms used for solving this problem is the Douglas-Peucker-Ramer algorithm also known as *iterated-endpoints-fit* [20]. This method is a greedy algorithm that starts by approximating the entire polygonal chain by one line segment connecting p_1 to p_n. Then it finds the vertex p_i that is furthest from the segment $[p_1, p_n]$ and if the distance is greater than the error tolerance the segment is divided into two segments $[p_1, p_i]$ and $[p_i, p_n]$. The procedure then recurses on the two resulting subchains. This algorithm is simple but has a worst-case complexity of $O(n^2)$ time and may not yield optimal approximations. However, the computational complexity can be improved using computational geometric techniques to $O(n \log n)$ time [32].

In 1985 Iri and Imai [33] proposed an elegant algorithm that finds the approximation that minimizes m subject to the two other constraints. Their approach is very general and elegant and can be applied in a wide variety of contexts. Their paper is still the best place to start for an introduction to this problem and its tutorial value is as relevant today as it was in 1985. The power of the technique lies in computing a weighted graph on the vertices of the polygon and then finding the maximum weighted path in this graph. The weight of an edge is the number of vertices of P that the edge skips. Thus the maximum-weight path contains the fewest vertices possible under the constraints. The different methods of approximation determine the way in which the graph is computed while the rest remains the same. It is an elegant and convenient method for obtaining constrained optimal approximations. These may be either to minimize the number of vertices in the approximation while respecting a prespecified error tolerance, or to minimize the approximation error with a prespecified upper bound on the number of vertices to be used in the approximation. For more recent results that use this general graph-theoretical framework with these and other approximation criteria see Eu and Toussaint [24] and Chan and Chin [8]. In particular, Chan and Chin [8] have improved the complexities of the algorithms of Iri and Imai. They reduce the complexity of the minimum-segment approximation algorithm for an input polygon of n vertices from $O(n^2 \log n)$ to $O(n^2)$ time. They also reduce the complexity of the minimum error approximation algorithm from $O(n^2 \log^2 n)$ to $O(n^2 \log n)$ time.

4 Shape Analysis

4.1 The medial axis

The two fundamental problems in OCR are feature extraction (shape analysis) and classification (decision rules). One of the most general and well studied techniques of shape analysis is via the medial axis (or skeleton) of a shape. Skeletonization and/or thinning algorithms have a long history [20]. There are two fundamental approaches: pixel-based methods and vectorized approaches. For the latter, where shapes are represented as polygons, computational geometry has great potential. The first $O(n \log n)$ time algorithm for computing the medial axis of a simple polygon with n vertices is due to D. T. Lee [39] in 1982. Since then it was a long-standing open problem whether this complexity could be improved. It was finally shown in 1995 that the problem could be solved in $O(n)$ time [11]. For practical applications like OCR in DIA, however, it is not yet clear how useful these results will be. More recently a very simple algorithm due to Evans, Middleditch and Miles [25] has been proposed which may be preferable in practice. Although the complexity of this algorithm is $O(n^2)$ it uses simple data structures and is also robust under certain degeneracies. Clearly in a field like DIA degeneracies cannot be ignored. In the early days of computational geometry many assumptions of general position were made to simplify the design of the algorithms. This area is now receiving considerable attention in the computational geometry community. However, we defer discussion of this topic to a later section.

4.2 Proximity graphs

In some contexts such as the textline inference problem in DIA described earlier the input consists of a set of disconnected distinct dots (points). There the use of proximity graphs such as bamboo-fields proved to be a very powerful tool for solving the problem of line detection. However, proximity graphs are very powerful tools for shape analysis in OCR and pattern recognition in general. Let $S = \{x_1, x_2, ..., x_n\}$ be a finite set of points in the plane. The problem of extracting the perceptual structure from S is a long standing problem and there are several versions of it depending on whether one is interested in the "inner" structure (skeleton) or the "outer" structure (shape hull). Early approaches to this problem employed *ad hoc* heuristics. The first attempt to put such heuristics on a firm geometrical foundation was what I called the *Gabriel hull* in 1980 [58]. The Gabriel hull of S is the boundary of the outer face of the Gabriel graph of S [36]. Since then there have been a score of improvements of this idea. A proximity graph on a set of points is a graph obtained by connecting two points in the set by an edge if the two points are close, in some sense, to each other. The minimum spanning tree (MST), the relative neighborhood graph (RNG), the Gabriel graph, and the β-skeletons are proximity graphs that have been well investigated in this context [36]. Recently there has appeared a new proximity graph called the *crust* with some very interesting properties for shape analysis that should make it useful for application in practice [2], [26]. For the most recent update of the most recent advances in this area see the column by O'Rourke [47].

In 1988 I proposed a graph I called the *sphere-of-influence* graph intended to capture the low-level perceptual structure of visual scenes consisting of dot-patterns (point-sets) [59]. Avis and Horton [36] showed in 1985 that the number of edges in the sphere-of-influence graph was bounded above by $29n$. The best upper bound until recently remained fixed at $17.5n$. Finally in 1999 Michael Soss [57] brought this bound down to $15n$. David Avis has conjectured that the correct upper bound is $9n$ and has found examples that require $9n$ edges, so the problem is still open, although this bound is of pure theoretical interest. The fact that the sphere-of-influence graph contains at most a linear number of edges suffices for its efficient computation. Furthermore, from the shape measurement point of view, the graph suffers from almost none of the drawbacks of previous methods and for a dot pattern consisting of n dots can be computed efficiently in $O(n \log n)$ time. Soss has also given results on the number of edges for metrics other than Euclidean [56]. Finally, Dwyer [21] has some results on the expected number of edges in the sphere-of-influence graph.

4.3 The line-sweep technique

One of the most useful paradigms for efficient algorithm design to emerge from research in computational geometry is the *line sweep* technique [51]. In this method of processing geometric data, a line (typically vertical) is swept across the plane (typically from left to right) through a presorted set of vertical slabs determined by relevant "events". In the case of polygons these events are the polygons' vertices. Efficiency is usually obtained by maintaining a balanced tree at the position of the line. As the line sweeps through the events, these are inserted and deleted as appropriate from the balanced tree. The first application of this paradigm to shape analysis through thinning is due to Chang, Lu and Pavlidis [9]. The success of their approach will no doubt result in greater application of this elegant computational geometric technique to other problems in DIA.

4.4 Implicit representation of polygons

One class of documents quite different from text consists of maps. Such documents are the *sine qua non* in geographic imformation systems (GIS) for example. In many applications dealing with spatial data, whether they be map processing in GIS or shape analysis in DIA, linear constraints are considered to be a good representation of geometric objects because it is practically viable in terms of storage and efficiency of operations. In this representation, queries on geometric objects are expressed as operations on linear inequalities. Simple and fast algorithms for many fundamental operations on spatial data have been recently developed and can be found in [27] and [28]. In this model, the input is a convex polygon given as a set of linear constraints, i.e., by the intersection of a possibly redundant set of halfplanes. Given a set $H = \{H_1, \ldots, H_n\}$ of halfplanes, a variety of computation and optimization problems exists for the convex polygon $P(H) = \cap_{i=1}^{n} H_i$. Two such examples are: computing the two supporting lines of $P(H)$ from an external point, and computing the minimum distance to the polygon from an external point. There also exist problems for which the input consists of two convex polygons. Some examples of these are: finding the two separating tangents of two disjoint polygons, as well as their two common external tangents, and computing the minimum distance between two disjoint polygons.

Obviously, one approach to solving these problems is to "construct" the polygon; that is, obtain the ordered list of the vertices of the polygon by intersecting the halfplanes that define it, and then use existing algorithms for "ordered" polygons to solve the problems. However, computing halfplane intersections implies that such a procedure will always be $\Omega(n \log n)$ [51]. The more interesting question which [27] and [28] address is to determine which problems can be solved in optimal $\Theta(n)$ time, and which have an inherent complexity $\Omega(n \log n)$. They show, for example, that computing the two supporting lines from an external point, or computing the minimum distance between two polygons takes linear time, whereas computing a diametral pair of points, or the minimum enclosing circle of $P(H)$ are $\Omega(n \log n)$.

The linear time algorithms in [27] and [28] are based on a prune-and-search strategy. Each of these algorithms applies an oracle (the search step) that allows it to discard a constant fraction of the halfplanes H_i in linear time (the prune step). Recursively applying the prune-and-search strategy leads to a linear-time algorithm due to the properties of the geometric series. In order to guarantee that a constant fraction of the input halfplanes is eliminated at each step, it is usually necessary to compute a median value, which can be obtained in linear time [51]. The linear time search procedure is specific to each problem, whereas the prune step is always the same. Identifying linear time search procedures for various fundamental geometric problems is the main task of the design process for this class of problems. When such a search procedure cannot be found, $\Omega(n \log n)$ lower bounds usually result.

5 Decision Rules

In the non-parametric classification problem [20], [18] we have available a set of n feature (shape measurement) vectors taken from a collected data set of n objects (patterns) denoted by $\{X, Q\} = \{(X_1, q_1), (X_2, q_2), ..., (X_n, q_n)\}$, where X_i and q_i denote, respectively, the feature vector on the ith object and the class label of that object. One of the most attractive decision procedures is the nearest-neighbor rule (NN-rule) [14], [19]. Let Y be a character (feature vector) to be classified and let X_k^*be the feature vector in $\{X, Q\} = \{(X_1, q_1), (X_2, q_2), ..., (X_n, q_n)\}$ closest to Y. The nearest neighbor decision rule classifies the unknown object Y into class q_k^*. In the past many DIA practicioners have resisted using the NN-rule on the grounds of the mistaken assumptions that (1) all the data X, Q must be stored in order to implement such a rule and (2) to determine the nearest neighbor of a character to be classified, distances must be computed between the unknown vector Y and all members of $\{X, Q\} = \{(X_1, q_1), (X_2, q_2), ..., (X_n, q_n)\}$. As we shall see below, computational geometric progress in the 1980's and 1990's along with faster and cheaper hardware has made the NN-rule a practical reality for DIA in the 21st Century.

5.1 Editing training data

In 1979 a method was discovered [62] to edit (delete) "redundant" members of $\{X, Q\}$ in order to obtain a subset of $\{X, Q\}$ that implements *exactly* the same decision boundary as would be obtained using all of $\{X, Q\}$. This approach was later generalized in [61].

Such methods depend on the computation of Voronoi diagrams and other proximity graphs that are subgraphs of the Delaunay triangulation, such as the Gabriel graph [61]. For example, with Voronoi editing every point (feature vector) that has all its Voronoi neighbors labelled the same way (object class), is discarded from the set. Voronoi editing does not change the error rate of the resulting decision rule because the nearest nighbor decision boundary with the reduced set is identical to that obtained by using the entire set. While this approach to editing sometimes does not discard a large fraction of the training data (say 90 percent), that information in itself is extremely important to the OCR designer because the fraction of the data discarded is a measure of the resulting reliability of the decision rule. If few vectors are discarded it means that the feature space is relatively empty and more training data are urgently needed to be able to obtain reliable and robust estimates of the future performance of the rule.

The classical approaches to k-NN decision rules obtain the k nearest neighbors of Y based purely on distance information without regard to how these neighbors are distributed around Y. Recently new definitions of neighborhoods have been proposed and new k-NN decision rules based on proximity graphs [36] have been investigated [53].

5.2 Searching nearest neighbors

Another potential problem in the implementation of nearest-neighbor decision rules, whether editing has or has not been performed on the training set, concerns the efficient search for the nearest neighbor of an unknown vector in order to classify it. Various methods exist for computing a nearest neighbor without computing distances to all the candidates [1]. In fact, the region point-location techniques find the nearest neighbor without computing any distances at all. The most promising algorithms for this problem, difficult when the dimension is high, use either bucketing techniques or k-d trees. Many algorithms have a complexity that depends exponentially on the dimension d and so for dimensions higher than about 8 they become prohibitive in practice. Thus for OCR applications where a feature vector has 80 moment features the situation is hopeless. On the other hand an 80-dimensional feature space is preposterous given the emptiness of the space. So perhaps the approach should be to find say 5 good features for the problem. Indeed, in a new approach that uses a genetic algorithm, editing the training data *and* reducing the number of features used is carried out simultaneously [38].

One of the fastest nearest neighbor search algorithms is due to Meiser [42] and runs in time $O(d^3 \log n)$, where d is the dimension. More recent algorithms which achieve efficiency at the expense of knowing the distribution of query points in advance have been found by Clarkson [12]. One approach to practical applications is of course to sacrifice finding the *exact* nearest neighbor. If we are satisfied with finding approximate nearest neighbors then more efficient algorithms are available [3], [34], [37]. For apllication results of some new efficient nearest neighbor searching algorithms in handwritten character recognition see [43]. For additional references to the latest results concerning nearest neighbor search in arbitrary dimensions and for pointers to many other key recent results the reader is referred to [1].

5.3 Estimation of misclassification

A crucial problem in OCR and pattern recognition in general is the estimation of the performance of a decision rule [18]. Many geometric problems occur here also, where compuational geometry offers elegant and efficient solutions. For example, a good method of estimating the performance of the NN-rule is to delete each member of $\{X, Q\} = \{(X_1, q_1), (X_2, q_2), ..., (X_n, q_n)\}$ in turn and classify it with the remaining set. Geometrically this problem reduces to computing for a given set of points in d-space the nearest neighbor of each (the all-nearest-neighbors-problem). Vaidya [63] gives an $O(n \log n)$ time algorithm to solve this problem.

6 Handling Degeneracies

When application to practice is not their main destiny, algorithms in computational geometry are usually designed for the real RAM (random access machine) assuming that the input is in *general position*. More specifically, the general position assumption implies that the input to an algorithm for solving a specific problem is free of certain degeneracies. Yap [64] has distinguished between intrinsic or *problem-induced* and extrinsic or *algorithm-induced* degeneracies. For example, in computing the convex hull of a set of points in the plane, where "left" turns and "right" turns are fundamental primitives, three collinear points constitute a problem-induced degeneracy. On the other hand, for certain vertical line-sweep algorithms such as that used for shape thinning [9], two points with the same x-coordinate constitute an algorithm-induced degeneracy. Computational geometers make these assumptions because doing so makes it not only much easier to design algorithms but often yields algorithms with reduced worst-case complexities. On the other hand, to the implementers of geometric algorithms such as DIA practicioners these assumptions are frustrating. Programmers would like the algorithms to work for any input that they may encounter in practice, regardless of the degeneracies that such an input may contain.

Due to the practical importance of having algorithms work correctly for degenerate input, there has recently been a flurry of activity on this problem in the computational geometry literature. In one method to remove degeneracies for solving a problem, the approach is to compute some *approximation* to the exact solution of the problem when a degenerate input is encountered. The other well known class of methods for handling degeneracies is via *perturbation*. Here the input is perturbed in some way by an infinitesimal amount so that the degeneracies are no longer present in the perturbed input. For a review of perturbation schemes, see Emiris and Canny [22], Yap [64], and Emiris, Canny and Seidel [23].

The above methods give the implementer two rather unsatisfactory choices: find an *approximate* solution to the *original* problem given, or find an *exact* solution to an *approximation* of the original problem. Sometimes it may be possible to convert the approximate solution obtained from perturbation methods to the exact solution by using some kind of *post-processing* step but this step may be complicated [7].

One may wonder if one can have one's cake and eat it too. Can we find the *exact* solution for the *original* problem with simple algorithms that handle degeneracies? The answer depends on the type of cake we want to eat. Gomez et. al., [30, 29] address an issue concerning the assumption of non-degeneracies which has received very little attention in the past. Often a typical computational geometry paper will make a non-degeneracy assumption that can in fact be removed (*without* perturbing the input) by a global rigid transformation of the input (such as a rotation, for example). Once the solution is obtained on the transformed non-degenerate input, the solution can be transformed back trivially (by an inverse rotation) to yield the solution to the original problem. In these situations, by applying suitable *pre-* and *post-* processing steps, the *exact* solution to the *original* problem is obtained using an algorithm that assumes a non-degenerate input, even when that input is in fact degenerate. This approach not only handles algorithm-induced degeneracies via orthogonal projections but some problem-induced degeneracies as well with the aid of perspective projections [29].

Gomez et al. [30] consider several non-degeneracy assumptions that are typically made in the literature, propose efficient algorithms for performing the pre- and post-processing steps (suitable rotations) that remove these degeneracies, analyze their complexity in the real RAM model of computation and, for some of these problems, give lower bounds on their worst-case complexity. The assumptions considered in [30] include:

1. no two points in the plane have the same x and y-coordinates,
2. no two points in space lie on a vertical line (regular orthogonal projection),
3. no two points in space have the same x, y and z-coordinates,
4. no three points in space lie on a vertical plane,
5. no two line segments lie on a vertical plane.

Incorporating the algorithms in [30] and [29] with those in the literature that make these non-degeneracy assumptions allows those algorithms to work even when the degeneracies are present, albeit at the cost of the increased complexity of computing the required rotations and the position of the center of projection in the case of perspective projections.

References

[1] Pankaj K. Agarwal and Jeff Erickson. Geometric range searching and its relatives. In J. E. Goodman B. Chazelle and R. Pollack, editors, *Advances in Discrete and Computational Geometry*. AMS Press, Providence, RI, 1998.

[2] Nina Amenta, Marshall Bern, and David Eppstein. The crust and the β-skeleton: Combinatorial curve reconstruction. Research Report, Xerox PARC, 1997.

[3] S. Arya, D. M. Mount, and O. Narayan. Accounting for boundary effects in nearest-neighbor searching. *Discrete and Computional Geometry*, 16:155–176, 1996.

[4] Tetsuo Asano and Naoki Katoh. Variants for the Hough transform for line detection. *Computational Geometry: Theory and Applications*, 6:231–252, 1996.

[5] Jean-Daniel Boissonat and Mariette Yvinec. *Algorithmic Geometry.* Cambridge University Press, 1998.

[6] Prosenjit Bose, Jean-Denis Caron, and Kilani Ghoudi. Detection of text-line orientation. In *Proceedings Tenth Canadian Conference on Computational Geometry*, pages 98–99, Montreal, Quebec, Canada, 1998.

[7] C. Burnikel, K. Mehlhorn, and S. Schirra. On degeneracy in geometric computations. In *Proc. 5th ACM-SIAM Symposium on Discrete Algorithms*, pages 16–23, 1994.

[8] W. S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments or minimum error. *International J. Computational Geometry and Applications*, 6:59–77, 1996.

[9] Fu Chang, Ya-Ching Lu, and Theo Pavlidis. Feature analysis using line sweep thinning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:145–158, February 1999.

[10] Bernard Chazelle, Jacob E. Goodman, and Richard Pollack, editors. *Advances in Discrete and Computational Geometry.* AMS Press, Providence, RI, 1998.

[11] Francis Chin, Jack Snoeyink, and Cao-An Wang. Finding the medial axis of a simple polygon in linear time. In *Proc. 6th Annu. Internat. Sympos. Algorithms Comput.*, volume 1004 of *Lecture Notes Comput. Sci.*, pages 382–391. Springer-Verlag, 1995.

[12] Ken L. Clarkson. Nearest neighbor queries in metric spaces. In *Proc. 29th Annual ACM Symposium on the Theory of Computing*, pages 609–617, 1997.

[13] Melvin Cohen and Godfried T. Toussaint. On the detection of structures in noisy pictures. *Pattern Recognition*, 9:95–98, 1977.

[14] Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[15] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, Berlin, 1997.

[16] Frank Dehne and L. Ficocelli. An efficient computational geometry method for detecting dotted lines in noisy images. In *Proceedings Vision Interface '89*, pages 88–93, 1989.

[17] Frank Dehne and L. Ficocelli. Detecting dotted lines in noisy images. *The Computer Journal*, 33(5):424–428, 1990.

[18] Luc Devroye, Laslo Gyorfy, and Gabor Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer Verlag, 1996.

[19] Luc P. Devroye. On the inequality of Cover and Hart in nearest neighbor discrimination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 75–78, 1981.

[20] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* Wiley-Interscience, New York, 1973.

[21] Rex Dwyer. The expected size of the sphere-of-influence graph. *Computational Geometry: Theory and Applications*, 5:155–164, 1995.

[22] I. Emiris and J. Canny. A general approach to removing degeneracies. *SIAM Journal on Computing*, 24:650–664, 1995.

[23] I. Z. Emiris, J. F. Canny, and R. Seidel. Efficient perturbations for handling geometric degeneracies. *Algorithmica*, 19(1–2):219–242, September 1997.

[24] David Eu and Godfried T. Toussaint. On approximating polygonal curves in two and three dimensions. *CVGIP: Graphical Models and Image Processing*, 56(3):231–246, May 1994.

[25] G. Evans, A. Middleditch, and N. Miles. Stable computation of the 2d medial axis transform. *International Journal of Computational Geometry and Applications*, 8:577–598, 1998.

[26] Christopher Gold. Crust and anti-crust: A one-step boundary and skeleton extraction algorithm. In *Proc. 15th Annual ACM Symposium on Computational Geometry*, pages 189–196, 1999.

[27] Francisco Gómez, F. Hurtado, Suneeta Ramaswami, Vera Sacristán, and Godfried T. Toussaint. Implicit convex polygons. Technical Report SOCS-00.5, School of Computer Science, McGill University, July 2000.

[28] Francisco Gómez, Ferran Hurtado, Suneeta Ramaswami, Vera Sacristán, and Godfried Toussaint. Implicit convex polygons. In *Abstracts 14th European Workshop on Computational Geometry*, pages 83–85, 1998.

[29] Francisco Gómez, Ferran Hurtado, Antoni A. Sellarès, and Godfried T. Toussaint. Perspective projections and removal of degeneracies. In *Proc. of the Tenth Canadian Conference on Computational Geometry*, pages 100–101, 1998.

[30] Francisco Gomez, Suneeta Ramaswami, and Godfried T. Toussaint. On removing non-degeneracy assumptions in computational geometry. In *Algorithms and Complexity (Proc. CIAC' 97)*, Lecture Notes in Computer Science, pages 52–63. Springer-Verlag, 1997.

[31] Jacob E. Goodman and Joseph O'Rourke, editors. *Handbook of Discrete and Computational Geometry.* CRC Press LLC, Boca Raton, FL, 1997.

[32] John Hershberger and Jack Snoeyink. An $O(n \log n)$ implementation of the Douglas-Peucker algorithm for line simplification. In *Proc. 10th Annual ACM Symposium on Computational Geometry*, pages 383–384, 1994.

[33] Hiroshi Imai and Masao Iri. Polygonal approximations of a curve: formulations and algorithms. In Godfried T. Toussaint, editor, *Computational Morphology*, pages 71–86. North-Holland, Amsterdam, Netherlands, 1988.

[34] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annual ACM Symposium on the Theory of Computing*, 1998.

[35] D. J. Ittner. Automatic inference of textline orientation. In *Second Annual Symposium on Document Analysis and Information Retrieval*, pages 123–133, Las Vegas, 1993.

[36] Jerzy W. Jaromczyk and Godfried T. Toussaint. Relative neighborhood graphs and their relatives. *Proc. IEEE*, 80(9):1502–1517, September 1992.

[37] J. Kleinberg. Two algorithms for nearest-neighbor search in high dimension. In *Proc. 29th Annual ACM Symposium on the Theory of Computing*, pages 599–608, 1997.

[38] Ludmila I. Kuncheva and Lakhmi C. Jain. Nearest neighbor classifier: Simultaneous editing and feature selection. *Pattern Recognition Letters*, 20:1149–1156, 1999.

[39] Der-Tsai Lee. Medial axis transformation of a planar shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(4):363–369, 1982.

[40] Der-Tsai Lee. Computational geometry. In Jr. Allen B. Tucker, editor, *The Computer Science and Engineering Handbook*, chapter 6. CRC Press, Boca Raton, FL, 1996.

[41] M. C. Lin and D. Manocha, editors. *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science.* Springer-Verlag, 1996.

[42] S. Meiser. Point location in arrangements of hyperplanes. *Inform. Comput.*, 106:286–303, 1993.

[43] Luisa Micó and Jose Oncina. Comparison of fast nearest neighbor classifiers for handwritten character recognition. *Pattern Recognition Letters*, 19:351–365, 1998.

[44] Joseph S. B. Mitchell and Subhash Suri. A survey of computational geometry. In M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser, editors, *Network Models*, Handbook of Operations Research/Management Science, pages 425–479. Elsevier Science, Amsterdam, 1995.

[45] George Nagy. Twenty years of document analysis in PAMI. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:38–62, 2000.

[46] Joseph O'Rourke. *Computational Geometry in C (Second Edition).* Cambridge University Press, 1998.

[47] Joseph O'Rourke. Computational geometry column 38. *International Journal of Computational Geometry and Applications*, 10:221–223, 2000.

[48] Joseph O'Rourke and Godfried T. Toussaint. Pattern recognition. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 43, pages 797–814. CRC Press LLC, Boca Raton, FL, 1997.

[49] Janos Pach, editor. *New Trends in Discrete and Computational Geometry*, volume 10 of *Algorithms and Combinatorics.* Springer-Verlag, 1993.

[50] Claudia Pateras and Godfried T. Toussaint. Bamboo fields: A new proximity graph and its application to text-line orientation estimation in document analysis. In *Proceedings Third Workshop on Proximity Graphs*, pages 73–120, Mississippi State University, Starkville, Mississippi, 1994.

[51] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, 3rd edition, October 1990.

[52] Jorg-R. Sack and Jorge Urrutia, editors. *Handbook of Computational Geometry.* North-Holland, 2000.

[53] J. S. Sánchez, Filiberto Pla, and F. J. Ferri. Improving the k-NCN classification rule through heuristic modifications. *Pattern Recognition Letters*, 19:1165–1170, 1998.

[54] Minako Sawaki and Norihiro Hagita. Text-line extraction and character recognition of document headlines with graphical designs using complementary similarity measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1103–1109, October 1998.

[55] Steven S. Skiena. *The Algorithm Design Manual.* TELOS Springer Verlag, 1997.

[56] Michael Soss. The size of the open sphere of influence graph in L_∞ metric spaces. In *Proceedings Tenth Canadian Conference on Computational Geometry*, pages 108–109, Montreal, Quebec, Canada, 1998.

[57] Michael Soss. On the size of the Euclidean sphere of influence graph. In *Proceedings Eleventh Canadian Conference on Computational Geometry*, pages 43–46, Vancouver, British Columbia, Canada, 1999.

[58] Godfried T. Toussaint. Pattern recognition and geometrical complexity. In *Fifth International Conference on Pattern Recognition*, pages 1324–1347, Miami, December 1980.

[59] Godfried T. Toussaint. A graph-theoretical primal sketch. In Godfried T. Toussaint, editor, *Computational Morphology*, pages 229–260. North-Holland, Amsterdam, Netherlands, 1988.

[60] Godfried T. Toussaint. Computational geometry for document analysis. In *Third Annual Symposium on Document Analysis and Information Retrieval*, pages 23–42, Las Vegas, 1994.

[61] Godfried T. Toussaint, Binay K. Bhattacharya, and Ronald S. Poulsen. The application of Voronoi diagrams to nonparametric decision rules. In *Computer Science and Statistics: The Interface*, pages 97–108, 1985.

[62] Godfried T. Toussaint and Ronald S. Poulsen. Some new algorithms and software implementation methods for pattern recognition research. In *Proc. IEEE International Compututer Software Applications Conference*, pages 55–63, 1979.

[63] P. M. Vaidya. An O(n log n) algorithm for the all-nearest-neighbors problem. *Discrete and Computational Geometry*, 4:101–115, 1989.

[64] Chee K. Yap. Symbolic treatment of geometric degeneracies. *Journal of Symbolic Computation*, 10:349–370, 1990.