NSF CAREER Proposal: Approximation Algorithms for Geometric Computing

1 Overview

Computational geometry is the branch of theoretical computer science devoted to the design, analysis, and implementation of geometric algorithms and data structures. Computational geometry has deep roots in reality: Geometric problems arise naturally in any computational field that simulates or interacts with the physical world—computer graphics, robotics, geographic information systems, computer aided-design, and molecular modeling, to name a few—as well as in more abstract domains such as combinatorial geometry and algebraic topology. Aside from their obvious practical significance, geometric algorithms and data structures enjoy a rich and satisfying mathematical structure, and their development often requires tools from mathematical disciplines such as combinatorics, topology, and algebraic geometry, as well as traditional computational tools.

The proposal outlines a challenging career development plan focusing on research in a broad cross-section of computational geometry, building on and significantly broadening the PI's successful work in the field over the last several years. Specific problem areas in which the PI plans to work include approximation algorithms, kinetic data structures, spatial and temporal databases, external memory computation, geometric optimization, and clustering. This classification is at best a rough guide, as many interesting geometric problems fall into more than one category. Furthermore, the PI plans to continue combining theory and empirical experimentation in his work, putting an emphasize on algorithms that perform well in practice.

The traditional approach in developing and analyzing algorithms for solving problems in computational geometry relies on several assumptions which are essentially false in practice. Issues involving the inherent inexactness of computation, degeneracies, memory limitations, noise in the input, performance for real inputs vs. worst-case analysis, implementation time of an algorithm, and the importance of an exact solutions to optimization problems, are all issues that have to be addressed and are usually ignored by the CS theory community. The PI strongly believes that for theory to remain relevant for practitioners, all these issues must be addressed. Addressing some of these issues in the context of geometric computing is the motivation for the PI's proposed research. This in turn requires a fresh and innovative approach to solving geometric problems. Such non-traditional approaches involve the development of new computation models that are more appropriate than the current ones, and extracting from the tasks at hand the critical components required to solve them efficiently.

Approximation algorithms emerged in the last decade as a class of algorithms that address some (and sometimes all) of the problems mentioned above related to the difficulty of implementing theoretical algorithms in practice. Approximation algorithms solve optimization problems by providing a solution which is close to optimal. In many cases, an approximate solution of a guaranteed quality is good enough in practice. Furthermore, the added flexibility in designing an approximation algorithm results in considerably better performance than the corresponding exact algorithms. For example, the Euclidean Traveling Salesman problem is NP-Hard, but it can be ε-approximated in near linear time [Aro98].

The proposal concentrates on three major themes: (i) the development of efficient approximation algorithms, (ii) understanding the combinatorial complexity and behavior of approximate geometric structures, and (iii) handling motion efficiently using approximation techniques.

The proposed research dealing with approximation algorithms is described in Section 2. Discussion of research related to handling motion is described in Section 3.

2 Approximation Algorithms

As mentioned above, the traditional assumptions on the computation model used are not realistic. In "real" life, the input is usually degenerate and often inexact, and computations, usually carried out using floating-point operations, are inherently inexact as well. Even if the input is exact, performing the computation in an exact manner is usually hard, prone to errors, and significantly slower than floating-point computations. This requires exact arithmetic [Gae98], filtering techniques [BBP98], and robust computations [Sch99].

These shortcomings make the implementation of geometric algorithms a non-trivial undertaking. It is often infeasible in practice to find the real optimal solution of a geometric optimization problem. This is because the optimum might typically be hard to compute, and, due to the inexactness of the input and the computations, might be indistinguishable from another close, but suboptimal, solution. This suggests that instead of insisting on computing the exact solution for such an optimization problem, one should be satisfied with a possibly suboptimal solution that approximates the optimum reasonably well.

The PI has done extensive work on such simple and practical approximation algorithms [AHSV97, Har99a, Har99b, BH01, AAHS00, AHK00, AH01, Har01b, Har01c, HV01, Har01a] and plans to continue his work on this topic.

2.1 Proximity

Proximity (or nearest-neighbor search) is a generic tool in handling data. Given a set of of items as input (i.e., records in a database), one can usually interpret each item as a generic point in the appropriate space along with a metric over the input items. One can now perform NN (nearest neighbor) search on the data, in a very generic way. Thus, NN search can be applicable to a wide range of data.

One way of facilitating such a NN search, is by using Voronoi diagrams. Voronoi diagrams are a fundamental structure in geometric computing. They are being widely used in clustering, learning, mesh generation, graphics, curve and surface reconstruction, and other applications, see [Aur91, For97, AK00b]. While Voronoi diagrams (and their dual structure Delaunay triangulations) are simple, elegant and can be constructed by (relatively) simple algorithms, in the worst case their complexity is $\Theta(n^{\lceil d/2 \rceil})$. Recently, there was effort to quantifying situations when the complexity of the Voronoi diagram in 3D has low complexity [AB01], and when it has high complexity [Eri01].

Those results are discouraging as far as using those structures for dimension $d > 2$. Fortunately, one can do better if one is satisfied with an approximation [AMN$^+$98, Cha98, IM98, Har01c]. Those works introduce data-structures that offer polylogarithmic query time using near linear space (ignoring the dependency of ε). They have exponential dependency on d, and although there is recent work [IM98] on data-structures with polynomial dependency on the dimension, it seems to be complicated and probably not practical. Furthermore, they are algorithmic and do not have natural geometric interpretation.

There are numerous open problems for further research. The PI next mentions several of the problems that he is interested in:

1. All known data-structures for ε-approximate NN in low-dimension have an *overall* dependency on ε by a factor of $1/\varepsilon^d$. Specifically: (i) [AMN$^+$98] presented a data-structure with a factor of $1/\varepsilon^d$ in the query time but with linear space. (ii) [Har01c] has logarithmic dependency on $1/\varepsilon$ in the query time, but the space has a factor of $1/\varepsilon^d$. and (iii) [Cha98] has factors of $1/\varepsilon^{(d-1)/2}$ both in the space and query time. It would be nice to break this limit, as there is no clear reason for its existence. In particular, the PI believes that it might be possible to do logarithmic query time using $O(n/\varepsilon^{(d-1)/2})$ space. It would be also interesting to achieve a continuous trade-off between the query time and the space used. Recently, there was work on answering approximate NN queries in time which is polynomial in d [IM98, KOR98]. The PI believes that bringing insights

from those techniques into computational geometry might result in better and faster algorithms for approximate NN search.

2. The complexity of the Voronoi diagram of n lines in 3D is still not known, with a cubic upper bound, and quadratic lower bound [AS00]. It would be interesting to close this gap.

3. Recently, the PI [Har01c] showed how to compute an approximate Voronoi diagram of points in $\mathbb{R}^d$ of near linear size. In particular, this yields a decomposition of $\mathbb{R}^d$ into a near linear number of cells, so that each cell c has an associated point $p_c \in P$, such that for any point in c the point p_c is an approximate NN in P. In the process, the PI had considerably simplified the reduction of Indyk and Motwani [IM98] between NN search and point-location queries among equal balls (PLEB). As another consequence, this yields the currently fastest data-structure for answering approximate NN queries in constant dimension (i.e., $O(\log(n/\varepsilon))$ query time). However, the space used by this data-structure has an exponential dependency on the dimension.

   This result hints on a broad class of questions: Can we replace exact combinatorial structures by an alternative "approximate" structures that have lower complexity and/or are easier to compute? For example, currently nothing is known about "approximate" Delaunay triangulations in three and higher dimensions.

   For example, if one is interested in an approximate Voronoi diagram of lines in 3D, Chew *et al.* [CKS+98] showed that under a polyhedral metric the complexity of this diagram is $O(n^2\alpha(n)\log n)$ (ignoring the dependency on ε). Since one can arbitrarily approximates the Euclidean metric by using a refined enough polyhedral approximation to the unit ball, it follows that there is near quadratic bound on the complexity of an approximate Voronoi diagram of lines in 3D. However, the examples requiring quadratic complexity are quite contrived. It would be interesting to come up with a space decomposition that approximates the Voronoi diagram of the lines, and has complexity close to the optimal for this specific input (i.e., for most inputs a linear complexity space decomposition should be feasible). Recently, there was work done on quantifying the situations where the complexity of the Voronoi diagram of points becomes quadratic in three dimensions [Eri01, AB01].

4. The question of answering approximate NN queries efficiently for a surface in 3D is still open. Namely, given a polyhedral surface $\mathcal{P}$, one would like to preprocess it so that given a query point q quickly compute the nearest point on $\mathcal{P}$ to q (or, of course, approximate NN). The best theoretical solutions are based on vertical ray-shooting on an appropriate lower-envelope in 4D. As such, these algorithms are complicated and far from practical. The only practical algorithm, for the case where all the queries are provided in advance, seems to be the algorithm of Zomorodian and Edelsbrunner [ZE00] that uses rectangles and range-trees to improve the running time (however, this is a heuristic without any guarantee on the resulting performance). A *practical* data-structure for the general problem would have applications in surface reconstruction and simplification [HG97, ACDL00, Ram01]. Such data-structures would enable one to quickly compute the quality of the approximated model compared to the original model. This question is especially interesting for an empirical study, where a fast implementation would be very useful. To handle huge graphic models, some additional ideas would be required. (This question is related to convex shape approximation, see below for details.)

5. How to maintain an *approximate* Voronoi diagram of moving points? In particular, it seems that maintaining the exact Voronoi diagram of moving points, requires a large number of events, as the structure of the Voronoi diagram "vibrates", when minor geometric violations happen; namely, some point enters an inscribing disk of a Delaunay triangle and leaves it almost immediately (i.e., the point moves close to a tangent to this inscribing disk). If one maintains the exact Voronoi diagram, at least two events must be handled. However, if one uses an approximation, these

two events might be ignored altogether. Thus, there is a clear potential here to do better (as far as number of events) by maintaining an approximate Voronoi diagram (or the dual Delaunay triangulation).

2.2 Clustering

Clustering is a centeral problem in computer-science. It is related to unsupervised learning, classification, databases, spatial range-searching, data-mining, etc. As such, it has received much attention in computer-science over the last twenty years. There is a large literature on this topic with numerous variants, see [BE97].

One of the natural definitions of clustering, is the *min-max radius clustering*, also known as the *k-center clustering*. Here, given a set of n points with an associated metric, one wishes to find k centers (i.e., points), so that the maximum distance of a point to a center is minimized. There is a very simple and natural algorithm that achieves a 2-approximation for this clustering [Gon85], which can be implemented in $O(nk)$ time. Feder and Greene [FG88] showed that if the points are taken from $\mathbb{R}^d$, one can improve the running time to $\Theta(n \log k)$, by a different implementation of the greedy algorithm mentioned above, that is optimal in the comparison model. They also showed that doing better than 1.822-approximation is NP-hard (under the L_2 metric). Recently, the PI [Har01a] showed how one can implement this greedy clustering algorithm in $O(n)$ expected time, for $k = O(n^{1/3}/\log n)$.

Intuitively, the new algorithm is based on the idea of first computing a good clustering of a random sample, and then by using all the points that violate this clustering, one can compute the required clustering.

An interesting question is whether one can cluster the points in an I/O efficient fashion. In particular, the new algorithm reads each point a constant number of times in an I/O efficient fashion (but the constant is exponential in d). It would be interesting to come up with a trade-off between the number of times one need to read a point, and the quality of clustering computed. In particular, if one is satisfied with $2k$ clusters instead of k clusters, the PI's result implies that one can read the points only once (except for a small subset of the points). This may be important in clustering very large data-sets, when reading the input even once is expensive.

Another direction for future research is maintaining the clustering under insertions and deletions. Although there are several recent results on this [CCFM97], the PI believes that one can do better by using the techniques of Callahan and Kosaraju [CK95] together with the clustering techniques of Feder and Greene [FG88].

The question of how to handle outliers is of critical importance. To a large extent, the success of learning techniques in clustering follows from their inherent ability to handle noise and outliers [DHS01, DV01]. However, the standard k-center clustering is extremely sensitive to noise, and although there was recent work in theory on doing clustering with noise [CKMN01, ADPR00], there is still a lot of ground for further research. For example, if one assumes the number of outliers is truly small (i.e., $< n/(k \log n)$), then it is clear that one can cluster almost all the points without clustering the outliers (just by picking a random sample and clustering it - with constant probability it would not contain an outlier). Now, one can insert the remaining points one by one into the existing clustering. If such an inserted point cause a substantial deterioration in the quality of the clustering, one might as well throw it out and consider it to be an outlier. The PI believes that this approach might lead to a simple definition and algorithms for removing outliers.

Also, the PI is interested in handling and clustering inputs from high-dimensional space. In particular, the PI is interested in deploying techniques from learning to handle such inputs. One such problem is the question of projective clustering [AP00] – where it seems that some of the learning/classification algorithms do much better than is predicted by theory, as the "real" dimension of the input is much smaller than the input dimension.

Formally, given a set P of n points in $\mathbb{R}^d$ (d is quite large in this case, and might be as large as n). We would like to find a k dimensional affine subspace of $\mathbb{R}^d$ that contains a (large) constant fraction of the points of P. Such a k-flat is known as a projective cluster, and projective clustering is the problem of finding such a cover of most of the input points using a small number of clusters.

Although this question seems to be NP-Hard (it depends on the exact variant – some are NP-Hard and the status of others is still open), there are empirical studies [JMN99, AP00] suggesting that it is solvable in practice. An interesting question is what reasonable assumptions can be made to render this problem solvable? Can one provide specific conditions and an algorithm, such that if the conditions hold the algorithm runs in polynomial time?

Finally, clustering is to some extent an amorphous question. The measures used to judge what are the best clustering are somewhat contrived, and do not necessarily have a clear relation to the application at hand. Although there is work trying to categorize what is a good/bad clustering [KVV00], this is still not well understood. While it is clear that no general technique can be develop for this, the PI believes that by using random sampling and other techniques, it might be possible to quickly cluster most of the points, so that only a small fraction of the points needs further and more careful clustering. In particular, it might be possible to capture very tight clusters in the beginning of the clustering process, so that the remaining points, either yield large clusters, or alternatively should be treated as outliers. Note, that the question of what is an outlier, is closely related to what is a good clustering.

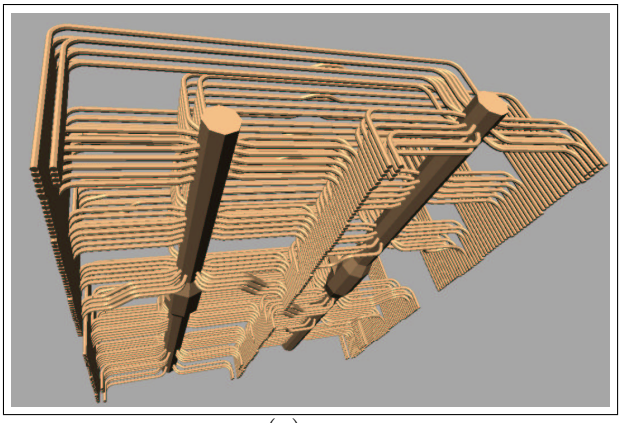
2.3 Shape Matching, Fitting, and Simplification

Recent progress in acquisition technologies, has supplied us with large amount of real world data of geometric models. Handling this data requires new techniques for understanding and manipulating the underlining model represented by such inputs. For example, given a densely sampled points on the surface of a sculpture [LPC+00], one would like to reconstruct the surface of the sculpture from those points (i.e., surface reconstruction [AK00a, ACDL00, AB98, Ram01]). Similarly, given a manufactured part (i.e., a pipe), one would like to perform quality assurance and verify that the part is close enough to the designed specifications. This involves *shape fitting* of the acquired data with the prespecified model, and computing the *matching distance* between the prespecified model and generated part. Some of the acquired models, being densely sampled, contain a lot of redundant geometric information. As such, one wishes to *simplify* them so that they preserve their geometric properties [HG97]. Furthermore, some of these models are huge, and handling them efficiently is very challenging, since they do not fit in main memory and I/O efficient algorithms are needed [Lin00, RL00, LPC+00]. Finally, the *core extraction* problem deals with computing the smallest subset of points from a model that represent the model according to a prespecified target function (see details below).

2.3.1 Shape Fitting

Given a set of points P in $\mathbb{R}^d$, *shape fitting* is the problem of finding the best shape (for example, hyperplane) that best fits the point-set. Shape fitting is a fundamental optimization problem and has numerous applications in graphics (shape simplification, collusion detection), learning, data-mining [FL95], databases [AWY+99] (projective clustering), metrology, compression, and geometric optimization.

A restricted variant of this problem is the case where the shape being fitted to the input is defined by a (small) constant number of parameters. Such problems fall into the realm of geometric optimization, and numerous variants have been solved [AAS97, AS96, AST94, AGSS89, EFNN89, EGS86, GLR97, LL91, MSY97, PS85, Riv79, RLW91, RZ92, SY95, SJ99, YC97, Cha00, AAHS00]. However, no unified theory has evolved, and solutions are usually based on a case by case analysis. Recently, the PI and Varadarajan [HV01] presented a general technique for approximate shape fitting for such variants.



(a)

Figure 1: A small part of the power-plant model `http://www.cs.unc.edu/~geom/Powerplant/|`.

If the shape being fitted is convex (i.e., find a cylinder that contains the point-set), then it is implied by recent work [BH01] that using convex shape approximation techniques one can first approximate the given input by a small point-set, and solve the optimization problem on this sampled subset. Since convex shape approximation is (relatively) well understood [Gru93], this results in fast and efficient approximation algorithms. However, if we are interested in a shape which is not convex (for example, we would like to cover a point-set by a minimum width annulus), then the problem becomes considerably harder. Although such problems have attracted a lot of research, most of the results rely on some additional assumptions about the input.

Following a sequence of papers [BH01, AH01, HV01] that showed the connection of convex shape fitting to other fitting problems (min width annulus, min volume bounding box, etc), the PI believes that now is the time to investigate the practicality of this technique, improve it, and improve existing algorithms for various special problems.

Recently, the PI presented a simple and practical algorithm [Har01b] for approximating the diameter of a point-set in 3d. The algorithm works in near linear time for real life inputs, while being quadratic in the worst case (when used to compute the exact diameter on synthetic inputs). The PI is interested in adapting similar approaches for solving similar other problems. The goal is to come up with *practical and fast* algorithms for computing the width, min-width cylinder, and min-volume bounding box of a point-set in three and higher dimensions (although an algorithm exists [BH01] for the minimum-volume bounding box, it is not completely satisfactory in practice). The PI believes that the insights and ideas that were useful in the case of the diameter should be useful also for these problems as well.

2.3.2 Model Simplification

Recently, there is a growing interest in handling huge scene models in graphics, both for real-time rendering and animation [SG01, RL00, Lin00, LPC$^+$00]. Shape simplification [HG97, SG01] has proved to be a critical technique in handling large models. However, some models are comprised of millions, and even billions, of triangles [LPC$^+$00]. This implies that the model is stored in an external memory (i.e., disk), and should be handled efficiently as far I/O operations are concerned. Although there is a large amount of research on theoretically I/O efficient algorithms [APR01], the graphical settings require efficient empirical performance.

Interesting questions that the PI plans to work on (cooperating with Michael Garland and other graphics people at UIUC) are: (i) Extracting a small and compact representation of a surface patch. Currently, the representation used by [SG01] relies on quadrics for this representation. However, this seems to be insufficient, as this representation might lose critical information about

the surface it represents. (ii) Extracting geometric information – the power plant model (available from `http://www.cs.unc.edu/~geom/Powerplant/`) is not only huge (13 million triangles), but is also very compact and can not be simplified directly. This model contains a large amount of pipes and similar geometric structures so that if one can extract them, then one can considerably compress the model, see Figure 1. For example, a pipe can be represented as a polygonal curve in space associated with a thickness parameter. How to identify those pipes and efficiently represent them is a question open for further research. This question is related to interesting questions about curve [DP73] and map simplifications.

2.3.3 Convex Shape Approximation

Given a set P of points in $\mathbb{R}^d$, compute a set $Q \subseteq P$ so that for any vector ν, the projection of P on the line spanned by ν (i.e., the smallest interval on this line that contains the projected points), and the projection of Q are the same up to a factor of $1+\varepsilon$. This problem is related to the problem of polytope/surface approximation. Here, we are given two convex polytopes P^-, P^+, and we need to find the smallest complexity polytope Q, such that $P^- \subseteq Q \subseteq P^+$. It is known that for the more general problem of finding the minimum complexity approximation for arbitrary polyhedral surface is NP-Hard [AS98]. For the convex case in 3D, Brönnimann and Goodrich [BG95] showed a constant factor approximation. For the general case in higher dimensions, Clarkson [Cla93] presented a $O(\log k)$ approximation, where k is the number of half-spaces needed in the optimal approximation (for a polytope with n vertices, a $O(\log n)$ is easy by stating the problem a set-cover problem).

Using those techniques, one can achieve a subset of P of cardinality $O(k \log k)$ that ε-approximates P, where k is the smallest such subset (however, these algorithms are complicated and their running time in d-dimensions is about $O(n^{\lfloor d/2 \rfloor})$, which might be too slow for $d \geq 4$). On the other hand, it is known that one can compute a subset of P of cardinality $O(1/\varepsilon^{(d-1)/2})$ that approximates P [HV01]. In particular, it would be interesting to develop a practical algorithm (and implement it) that can compute such a subset quickly for easy instances, that might deteriorate to the above stated bounds in the worst case. As the dimension increases, the hidden constants become worse, and an approach that can avoid an exponential constant for certain (easy) cases might be very useful in practice.

The PI next describe in more detail how the current convex shape approximation algorithms work. Given a point-set P one first computes an appropriate linear transformation T, and snaps the point-set $T(P)$ to an appropriate grid, while removing interior points [BH01] (intuitively, the linear transformation guarantees that the convex-hull of $T(P)$ is fat). This results in the set P_G that is of cardinality $O(1/\varepsilon^{d-1})$ (all this can be done in $O(n+1/\varepsilon^{d-1})$ time). The set P_G is a subset of grid with $O(1/\varepsilon)$ in each dimension. As such, its convex-hull $\mathcal{CH}_{grid} = \mathcal{CH}(P_G)$ has complexity $O(1/\varepsilon^{(d-1)d/(d+1)})$ [BL98]. This is a bound on the overall number k-faces of the CH of P_G, for $k = 0, \ldots, d-1$, and it is quite surprising as the standard bound $O(1/\varepsilon^{\lfloor (d-1)d/2 \rfloor})$ is much worse.

One can easily compute $\mathcal{CH}_{grid}$ in near-linear time in two and three dimensions. However, in higher dimensions the question of achieving a near-linear time algorithm for computing this convex-hull is still open. Since the point-set has so much underlining structure, it seems that a near-linear time algorithm should be achievable in this case. The fastest output-sensitive algorithm for computing convex-hull is due to Chan [Cha96], but its running time is near-quadratic; namely, slightly better than $O(N^2)$, where $N = O(1/\varepsilon^{d-1})$.

Anyway, to compute the approximating set Q to the set P_G, one uses Dudley's [Dud74] approximation technique. This involve spreading a spherical grid U of $O(1/\varepsilon^{(d-1)/2})$ points on a large enough sphere $\mathcal{S}$ around $\mathcal{CH}_{grid}$, and computing for each point of U, its closest point on $\mathcal{CH}_{grid}$. This can be done either in linear time by scanning the facets of the convex-hull, or alternatively, using the fact that this is problem is an LP-type problem, and can be solved in linear time in the

number of points of P_G, without computing $\mathcal{CH}_{grid}$ explicitly [Gär95]. Thus, each point of U, yields a new point on the boundary of $\mathcal{CH}_{grid}$; let Q denote the resulting set. One can show that the set Q ε-approximates the set P_G [Dud74], and as such the set $T^{-1}(Q)$ is an $O(\varepsilon)$-approximation the original point-set P. Furthermore, $|Q| = O(1/\varepsilon^{(d-1)/2})$ and the overall time to compute it is $O(n+N^{3/2}) = O(n+1/\varepsilon^{3(d-1)/2})$. Thus, the question is whether the running time of this algorithm can be improved to, say, $O(n + 1/\varepsilon^{d-1})$?

An interesting property of the representation of $\mathcal{CH}(Q)$, is that one can replace each point of Q by the hyperplane passing through it and tangent to $\mathcal{CH}_{grid}$. Taking the intersections of the induced half-spaces form an (outer) ε-approximation to $\mathcal{CH}_{grid}$. Let $\mathcal{C}_H$ denote the resulting Polytope. Although, the underlining polytope $\mathcal{CH}_{grid}$ has overall complexity $O(N)$, currently no similar bound is known for $\mathcal{C}_H$, and in fact the best bound currently known is (the standard) $O(1/\varepsilon^{\lfloor (d-1)d/2 \rfloor})$. However, the PI believes the real bound should be much smaller, and closer to $O(N)$.

Namely, one can currently have a compact implicit representation of $\mathcal{C}_H$ (as the collection H of $O(\sqrt{N})$ half-spaces), or an approximate representation as the convex-hull of $\mathcal{CH}(Q)$ (by the point-set Q), but no explicit compact representation of $\mathcal{C}_H$ is known. Another question, is whether one can do approximate point-location in $\mathcal{C}_H$ (or $\mathcal{CH}(Q)$) quickly. Indeed, assume that the diameter of $\mathcal{C}_H$ is $O(1)$. Given a query point q, we want to decide whether $q \in \mathcal{C}_H$, or outside $\mathcal{C}_H$. Clearly, this can be decided by scanning the halfspaces of H in $O(\sqrt{N}) = O(1/\varepsilon^{(d-1)/2})$ time. Furthermore, let assume that we are willing to accept an approximate result. Namely, if the distance of q from $\mathcal{C}_H$ is smaller than ε, then a positive answer is still valid. How fast can one do this approximate point-location? Well, one can certainly do it in $O(\sqrt{N})$ time as described above (using $O(\sqrt{N})$ space). One can also construct an appropriate ε-grid and mark all the cells of the grid that are intersecting $\mathcal{C}_H$. Now, performing an approximate point-location query would take $O(\log 1/\varepsilon)$ time, but the space required is $O(N)$. Can one do better?

Interestingly enough, as mentioned before, the same query time/space threshold holds here as in the approximate NN query problem. The reason is that the problems are related, as a fast approximate point-location in $\mathcal{C}_H$ would yield a faster algorithm for performing approximate NN query. To see that, observe that Clarkson's algorithm [Cla94] resolve the approximate NN query problem by performing a sequence of approximate point-location queries in a convex polytopes. As such, any progress on this problem would yield progress on the approximate NN search problem.

2.3.4 Matching Distance Between Curves

Given two (polygonal) curves α, β in the plane, one would like to match them up in an optimal way (i.e., find a continuous mapping from the points of α to the points of β, so that the mapping maps the endpoints of one curve to the endpoints of the other curve). Such a mapping is especially useful if it is associated with an appropriate metric between curves. Such a matching has various applications, from finding the best possible way to guard a polygon under the chain model [EGH$^+$00], to finding a (non self intersecting) morphing between two curves that minimizes a certain energy function [EGHM01], and for handwriting/signature recognition [MP99].

The Fréchet metric [AG95, Fré06] is one of the most natural measures of this type: It requires finding a mapping f between the curves so that $\mathcal{W}(f) = \max_{x \in \alpha} \|xf(x)\|$ is minimized. This is also known as the person-dog metric: imagine a person walking on α and a dog walking on β. The Fréchet distance between α and β is the shortest leash that enable both the person and the dog to travel along α and β with a leash connecting them.

Playing around with this metric, one realizes that because of the continuity requirement on the mapping f, this measure is in a lot of cases much better suited than the classical Hausdorff distance between the curves [HKK92]. Unfortunately, computing the Fréchet distance requires quadratic time in the complexity of the curves [AG95]. It is currently an open question if it is possible to do

better than quadratic even when one try to approximates this quantity within a constant multiplicative constant. (A subquadratic approximation algorithm is known for a somewhat restrictive case when the curves are globally well behaved. See [AKW01].) The PI is interested in resolving this question, even for the case when the two curves are disjoint, share the same x-extent and are both x-monotone.

Even for the following simple restricted case, the PI does not know a subquadratic solution: Let α, β be one dimensional curves limited to integer coordinates between 1 and k (i.e., α, β have a total of n vertices on the real line, where the vertices are restricted to the numbers $\{1, \ldots, k\}$. Namely, α, β are sequences of integer numbers.). Question: Is the Fréchet distance between α and β smaller than 2? The PI believes that solving this question would be the key to making progress for the above mentioned problems.

An interesting question is the extension of the notion of the Fréchet distance to surfaces in 3D. Such an extension would have wide applications in graphics, computational biology and modeling. In fact, the question of how to match two arbitrary surfaces in 3D (i.e., docking) is still not well understood.

2.3.5 Core-sets

The above discussion raises the following question: Given a set P of n points in $\mathbb{R}^d$, and a certain optimization problem $f(P)$ that one wants to solve on P (e.g., find the minimum radius ball that contains P). Can one find a small subset Q of points of P, so that computing $f(Q)$ results in an ε-approximate solution to $f(P)$? Such a set Q is called a *core-set* of P. The main question is bounding the size of such a core-set for various problems. Recently, [HV01] showed bounds on the size of the core-set for various problems. However, their bounds depend exponentially on d. It might be that at least in some cases, this dependency on d can be improved to be polynomial. As an example, consider the core-set for finding the min-radius cylinder that contains a given point-set. By [HV01], one can argue that a core-set of size $O(1/\varepsilon^{(d-1)/2})$ exists. However, if one is interested in a 2-approximation, 3 points are enough (for any dimension) – take the diametrical pair, together with the point furthest away from the line spanned by the diametrical pair. For cases where d is huge, the existence of a core-set that has sublinear dependency on d, might be quite useful. Note, that if one can extract such a core-set, then one can solve the optimization problem on the core-set, resulting in a considerably faster algorithm.

Even if one is satisfied with such a large core-set, the current algorithms are very slow. In particular, the fastest algorithm (it relies on the technique for convex shape approximation described above) for computing such a core-set of cardinality $O(1/\varepsilon^{(d-1)/2})$ (for this problem) is $O(n + 1/\varepsilon^{3(d-1)/2})$. On the other hand, if one is satisfied with a core-set of size $O(1/\varepsilon^{d-1})$ then it can be computed in $O(n + 1/\varepsilon^{d-1})$. Can this running time be improved?

This problem is under ongoing research by the PI together with Piotr Indyk and Mihai Badoiu.

3 Handling Motion

Motion appears everywhere in the world. As such, there is a large amount of research into how to manipulate, store, extract and simulate motion. Motion arises naturally in protein folding in biology, robotics, ad-hoc mobile networks, in systems using GPS (geographic positioning systems) to track their current position, and in simulating physical models. In the computational geometry community, work on moving points focused on bounding the number of changes in various geometric structures (e.g., convex hull, Delaunay triangulation) as the points move [Ata85, SA95]. Basch *et al.* [BGH97] introduced the notion of *kinetic data structures*. See [AEG98, Gui98] for relevant work.

A new exciting research topic is the understanding of geometry of motion, when one is interested in approximate notions. The PI believes that this approach is especially promising as it enables

one to compute the quantity of interest, without being limited by strict synthetic requirement that arise if one try to maintain a "rigid" structure/exact quantity over time. A striking example of this is the maintenance of the diameter of a linearly moving point-set: Agarwal *et al.* [AGHV97] showed that one has to handle quadratic number of events in the worst case. However, if one is interested in only maintaining an ε-approximate diameter (i.e., two points of the input so that the distance between them is an ε-approximation to the length of the diameter at this point in time) then the diameter changes only $O(1/\varepsilon)$ times (in the plane) and it can be computed and maintained in near linear time, as was recently shown by Agarwal and the PI [AH01].

Recently, Varadarajan and the PI had extended the work of [AH01], to other geometric optimization problems involving moving points. In particular, they show that various approximate geometric measures of moving points change only a small number of times, and the number of changes depend only the quality of approximation and the degree of the motion of the points, but not on the number of points involved. In particular, this was shown for the measures of: width, min-width cylinder, min-width annulus or shell, min-volume bounding box, and others. This is a very exciting development as it shed a light on an uncharted territory: Can one have substantial gains in handling motion by resorting to approximate approach? The PI plans extensive work on charting out this territory in the near future.

3.1 Clustering Motion

Let $P(t) = \{\mathbf{p}_1(t), ..., \mathbf{p}_n(t)\}$ be a set of n moving points in $\mathbb{R}^d$, with a degree of motion μ; namely, $\mathbf{p}_i(t) = (p_1^i(t), \ldots, p_d^i(t))$, where $p_j^i(t)$ is a polynomial of degree μ, and t is the time parameter. If one wish to answer spatial queries of the moving point-set $P(t)$, one need to construct a data-structure for that purpose. Most such data-structures for stationary points relies of space partition schemes, and while such partitions/clusterings are well understood for the case of stationary points, for the case of moving points almost nothing is known (see [AAE00, HV01, GGH$^+$01, AH01] for recent relevant results).

The hardness in maintaining and computing such clustering is the one underlining most kinetic data-structures: After the clusters are computed at a certain time, the quality of the clustering deteriorates as time progresses. To remain a competitive clustering (compared to the optimal clustering), one needs to maintain the clustering by either reclustering the points every once in a while, or alternatively, move points from one cluster to another. The number of such "maintenance" events dominates the overall running time of the algorithm, and usually the number of such events is prohibitively large. For example, it is easy to verify that a kinetic clustering of n linearly moving points must handle $\Omega(n^2)$ events in the worst case to remain competitive with the optimal k-clustering in any given time.

In [Har01a], the PI investigated a different approach, showing that if one is allowed to use $k^{\mu+1}$ clusters (instead of k), then one can compute a clustering which is (constant factor) competitive with the optimal k-clustering of the point-set in any point in time. This clustering is *static* and thus will not be required to handle any "maintenance" events. An efficient algorithm for computing this clustering was also presented.

To appreciate the above result, note that the motion of the points causes the distance between points to change continuously. Points which are at a certain time far, become close, and then far again. Although the motion is not chaotic, the underlining (changing!) structure of the clustering of the moving points can not be directly extracted from the points.

The above technique currently does not support insertions and deletions. The PI plans to extend this techniques to support such operations. Another interesting question is the question of extending this technique for the more traditional settings of kinetic data structures (KDS), where the motion is not prespecified in advance. In particular, in some situations the motion is not well behaved (i.e., protein folding), and handling it efficiently requires extending the above ideas.

For the larger picture, the problem of how to use such clustering for performing range searching and NN search still requires further research. Also, as the above result indicates, motion has a lot of internal structure. It would be interesting to extract this hierarchical information automatically. In particular, there is work in robotics on inverse kinematics [HKL97] and in computational biology [TPK01, SA01] (i.e., understanding how molecules move around). However, the problem is far from being well understood.

3.2 Querying Moving Points

Several areas such as digital battlefields, air-traffic control, mobile communication, navigation system, geographic information systems, call for indexing moving objects so that various queries on them can be answered efficiently; see [SJLL00, SWCD97] and the references therein. The techniques that have been successful for simulating some structures/phenomena over time are not always suitable for answering queries over moving objects. The queries might relate either to the current configuration of objects or to a configuration in the future — in the latter case, one should predict the behavior based on the current information. In either case, it is not advantageous to tracking the structure needlessly over periods when no queries are posed.

The continuously moving objects must be distinguished from discretely changing objects. It is relatively easy to keep track of the changes in the latter type of objects and to update the database of such objects as they change. Several indexing techniques already exist for storing and querying discretely moving objects; e.g., see the surveys [GG98] and the references therein. Continuously moving objects are, on the other hand, considerably harder to accommodate in databases because most existing database systems assume that the data is constant, unless it is explicitly modified. Such systems are not suitable for representing, storing, and querying continuously moving objects; either the database has to be continuously updated or a query output will be obsolete. A better approach is to represent the position of a moving object as a function $f(t)$ of time, so that changes in object position do not require any explicit change in the database system. With this representation, the database needs to be updated only when the function $f(t)$ changes, for example, when the velocity of an object changes.

The PI believes that new techniques introduced by [AH01, HV01, Har01a] should be useful in designing and developing such databases for moving points. In particular, in the context of GIS. Thus, the PI plans to continue working on extending his research into empirical studies of such databases. In particular, preliminary evidence [PAH01] shows that these techniques yield considerable improvement in practice over previous approaches.

3.3 KDS Extensions

Despite their great success, the PI believes that KDSs have several shortcomings: (i) Bounding the number of changes that a KDS goes through over time is notoriously hard. Even when a tight bound is known, it tends to be unacceptably large (i.e., quadratic or cubic). (ii) Working in vain – KDSs are usually being used as building block by higher level data-structures. Even if a KDS is never used, or is very rarely used, one still must spend a lot of time maintaining it without any benefit. (iii) In most cases, a KDS maintains a global heap of events – this induces hard numerical problems, as one has to resolve ordering on time stamps. Those stamps are the result of geometric computations of large depth, and are inaccurate, implying that one might need to resort to (expensive) exact arithmetic (which is not always possible). This also prevents the implementation of the algorithm in a distributed local fashion. Thus, the PI plans to research alternative approaches, and one such possible alternative computation model is outlined below.

One of the most challenging problems about KDS is to extend them so that they can run in a distributed fashion. For example, for a mobile ad-hoc network, where the geometric location of each node is locally known, one would like to maintain the connectivity of the network as nodes move around. Ultimately, one would like to perform the network connectivity in a local fashion,

where each node decides for itself what connections to maintain, and guarantee that the network remains connected. In particular, one would wish also to perform routing under this connectivity maintenance.

There has been some work on maintaining connectivity [HS99, HS01]. An alternative approach is to maintain a spanner of the nodes as they move around (a t-spanner is a sparse connected graph so that its shortest-path metric t-approximates the underlining euclidean metric). [KG01] showed that since Delaunay triangulation is a 6-spanner, one can use it to maintain a 6-spanner of the moving points. Those works involve a global approach, and they are somewhat complicated and are not completely satisfactory.

One possible approach, that has not been investigated yet, is to use the WSPD (well-separated pairs decomposition) [CK95] of the moving points. The nice property of this approach is that: (i) There is a natural $(1 + \varepsilon)$-spanner associated with a WSPD, (ii) a WSPD can deteriorate gracefully. Namely, if a pair is no longer well-separated, one can replace this pair by its children in the associated hierarchical decomposition. Of course, in the long run, this increases the number of active connections to be unacceptably high (i.e., each pair in the WSPD decomposition define one edges of the spanner, and as the number of pairs increases, so does the number of edges), but it enables one to delay the (global) update to a later stage. Thus, the PI believes that it is possible in this case to combine a global approach, together with a distributed approach, to maintain network connective efficiently for mobile ad-hoc networks.

An interesting problem, is how to use such a distributed spanner for performing routing in an ad-hoc network. The PI has done some work on routing in the recent past [BAH99].

3.3.1 Modifying KDS

The following is a list of several modifications to KDS that the PI plans to investigate:

1. **Removing the heap.** The usage of a heap, as mentioned above, causes computations of large (computational) depth to be carried out. This creates numerical problems when working with floating-point arithmetic. An interesting question is whether one can skip the heap all together. Alternatives, may include a lazy update approach, or maintaining different time-stamps in different part of the structure (i.e., similar in a sense to the topological sweep of [EG89]).
2. **Handling simultaneous events.** Current KDS has problems when handling simultaneous events (especially if they are close together). Developing a general technique for handling simultaneous events should be useful in practice.
3. **Weak Delaunay triangulation.** Can one maintain a lazy Delaunay triangulation? Namely, maintain a structure which is close to the Delaunay triangulation, but not necessarily identical. For example, what happens if one allows a weaker condition: Each inscribing disk of the triangles of the triangulations contain at most (say) 10 points. Of course, if one replicated every input point 11 times, then this triangulation would be identical to the Delaunay triangulation of the original set, so this seems a bit pointless. However, can one do better when the disk associated with each triangle contains at most $\sqrt{n}$ points? Can one maintain such a "flexible" triangulation more efficiently than a regular Delaunay triangulation?

   Another possible direction for further investigation is the following: Define the α-core of a disk D, as the disk of radius $\alpha \cdot \text{radius}(D)$ placed at the center of D. Now, one can ask for a triangulation, where a triangle is invalid if the α-core contains an input point (for $\alpha = 1$ this is just the regular Delaunay triangulation). Can one maintain such triangulation efficiently for moving points? Can such triangulation be computed efficiently in 3D? Does it have near linear complexity?

Another related question is to develop any KDS for maintaining a triangulation of a moving point-set that handles (provably) only quadratic number of events if the points move linearly in the plane.

3.3.2 The Jumping Frogs Model

The PI outline in this section a simple model that might serve as a replacement for KDS. The PI plans to continue working on formulating and using this model. The idea is to factor in lazy maintenance, and uncertainty about the locations of the moving points/objects in any given point in time. In the following, the model is considered in the context of answering approximate NN queries.

Assume that one is given a set of points P. Unlike KDS, assume that one does not know the exact location of the points. What is known is the previous last exact location of each point, and a (growing) region of where it might be (for example is a point is moving in a constant speed but in unknown direction, the region of uncertainty is a growing ball centered in the last known location). Assume that one has a probing oracle at their disposal, which can report the current exact location of a moving point. Thus, one should answer a NN query while minimizing the number of *probing queries*.

At this stage, the assumption about the availability of a probing oracle might seem unreasonable. However, this can be easily simulated by other models. Imagine for example that the moving objects update the system about their location every (say) minute. Instead of updating the data-structure directly every time such an update arrives, one can cache those updates and a probing query just returns the last location reported by the moving object.

To be more concrete, assume that the set of points are all moving with a maximum speed of one. Thus, if one knows the location of the point p to be $p(t_0) = (x_0, y_0)$ at time time t_0, then in time t the point $p(t)$ might be anywhere inside a disk of radius $t - t_0$ centered at $p(t_0)$. Namely, consider each point to be a growing ball as a function of time. Probing a point is no more than deflating the ball back to a point.

In the following, assume that the target is to minimize the number of probing queries, while completely ignoring the questions of processing time and space.

Thus, given a NN query point $q = (x_q, y_q)$ at time t, we first compute all the points that might be located at q at time t; namely, all the points that their corresponding disks contain q. Next, probe each one of those points, replacing their balls by balls of zero radius centered at their updated locations. Now, repeatedly find the closest ball to our query point, and update its location by probing. During this process, one maintains the closest (probed!) point p_{curr} found so far. One should stop as soon as $|qp_{curr}| \leq (1 + \varepsilon)r_{curr}$, where q is the query point, and r_{curr} is the distance to the closest disk that has not been probed yet.

Interestingly enough, if the disk of uncertainty of a point p being considered is smaller than εr_{curr} then one can return it as an approximate nearest-neighbor without performing any probing query.

At this point, it is interesting to compare this setting with the KDS setting for answering NN queries (or approximate NN queries). Currently it seems that the two settings are incomparable. Indeed, maintaining the Voronoi diagram of moving points might require a huge number of events, while our setting would be able to resolve the NN query without any probe. On the other hand, our setting might have a huge number of probes while the Voronoi diagram would not handle any events at all. (Imagine that all the points are moving in the same direction in the same speed. The combinatorial structure of the Voronoi diagram is not changing at all.) This follows from the fact that each model has different information at its disposal. One can extend this model so it incorporates also the KDS model inside it. However, understanding what is the right model to use requires further research.

Note, that for linearly moving points on the real line answering approximate NN queries, is as hard as the Hopcroft problem in the plane, as one can map each moving point to a line in the xt-plane. However, this is known to require $\Omega(n^{4/3})$ under a reasonable computation model [Eri95]. However, if one allows additive error, this lower bound no longer holds. Such additive error is a reasonable assumption as GPS chips compute its location up to a radius of ten feet.

Given a set L of n lines in the plane, and a set P of n points all in the unit square, and a parameter $\varepsilon > 0$, can one decide whether there is point of P at distance $\leq \varepsilon$ from a line of L? More formally, let G_ε be the partition of the unit square into cells of side length ε, and let S_ε be all the cells of G_ε that intersect the lines of L. The question now is whether there is any point of P inside the squares of S_ε. Can one design an algorithm that answers this question in near linear time in n, and polylogarithmic time in $1/\varepsilon$ (the interesting case is when $\varepsilon < 1/n^2$). If this is possible, than there is a possibility that one can handle approximate NN queries efficiently for linearly moving points.

4 Educational Activities

The PI believes that education is an exciting and integral component of his academic career, and he is firmly committed to excellence in teaching at all levels. As a member of the theory group at the University of Illinois, the PI will teach next year (2001/2) one of the standard undergraduate theoretical courses (CS373 combinatorial algorithms). This past year, the PI taught a course about geometric approximation algorithms and a course about randomized algorithms. The PI hopes that both courses become standard courses in the syllabus of the CS department at UIUC.

Course development The PI developed a course about approximation algorithms in geometry and has taught it twice (spring 2000 at Duke university, and fall 2001 at UIUC). The course covers material currently available only in research papers, and it is the intention of the PI to continue developing and updating this course. Teaching this course has helped the PI in formulating and understanding his research.

The PI provides, on his webpage, material concerned with the courses he taught (see: `http://www.uiuc.edu/~sariel/teach/`). Some of the class notes are being downloaded quite frequently.

Teaching Statement Teaching is important to me for several reasons: It is a way to provide a service to the community; it provides an opportunity to interact with intelligent people; it enables me to interest and expose students to my research topic; it enables me to achieve better understanding of the material taught; and it provides me with an opportunity to learn new material. This is why I am interested in teaching courses in a wide range of topics, and not only computational geometry. Overall, I enjoy teaching and respect its value and importance to society.

My teaching philosophy is that it should provide the students with the larger picture, i.e., when teaching a theoretical course, the students should be provided with a clear intuition and motivation behind what is being used and done in a formal (and sometime tedious) way. On the other hand, the real details should be covered, so the students will have enough understanding to apply and extend what they have learned. One should always consider the feasibility of what is being taught. "Can this be programmed? Can some similar ideas be used in practice? How fast will it be? What is the most natural thing to do in this case? Can one prove that it works? etc" – such questions (and others) are the key to successful research and software implementation.

Worthwhile research can be also be carried out directly by undergraduate students during a course. This can be done by presenting open questions in class, and discussing techniques and ideas that might lead to a solution for these problems. Another option, is to encourage and offer the students to implement a program related to the course, and carry out a competition between the implementations – grading their projects according to their quality. Such projects prepare the students for the work environment outside the university where they will need to cooperate with

others in a team, learn new technologies on their own, and implement software which is competitive. I had the pleasure of taking part in such a competition, carried out by Danny Halperin in his Computational Geometry course taught at Tel-Aviv university. The ideas underling the students programs were good enough to publish a paper about them [AHH+99], and the excellent practical results got me interested in the theoretical aspects of this problem, resulting in a paper [Har00].

When designing and giving a course it is important to provide some added value for the students, so even if they know all the material before hand, they would still gain some new insight and information which is not available to them by just reading the relevant books and papers. Such added value includes updating them with new results, pointing out connections to other problems and fields, and sharing insights gained by (sometimes painful) personal experience.

Overall, it is important to teach students to ask the right questions, as the current education concentrates on the art of answering, and less on the art of asking questions. Namely, asking questions is at the hart of creativity, intuition is at the core of answering those questions, and understanding the 'boring' details is a must to realizing the intuition into a new solution. And so, students should be supplied with tools and insights enabling them to play in all three fields simultaneously.

References

[AAE00] P. K. Agarwal, L. Arge, and J. Erickson. Indexing moving points. In *Proc. 19th ACM Sympos. Principles Database Syst.*, pages 175–186, 2000.

[AAHS00] P. K. Agarwal, B. Aronov, S. Har-Peled, and M. Sharir. Approximation and exact algorithms for minimum-width annuli and shells. *Discrete Comput. Geom.*, 24(4):687–705, 2000.

[AAS97] P. K. Agarwal, B. Aronov, and M. Sharir. Computing envelopes in four dimensions with applications. *SIAM J. Comput.*, 26:1714–1732, 1997.

[AB98] N. Amenta and M. Bern. Surface reconsruction by Voronoi filtering. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 39–48, 1998.

[AB01] D. Attali and J. D. Boissonnat. Complexity of the delaunay triangulation of points on a smooth surface. http://www-sop.inria.fr/prisme/personnel/boissonnat/papers.html, 2001.

[ACDL00] N. Amenta, S. Choi, T. Dey, and N. Leekha. A simple algorithm for homeomorphic surface reconstruction. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 213–222, 2000.

[ADPR00] N. Alon, S. Dar, M. Parnas, and D. Ron. Testing of clustering. In *Proc. 41th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 240–250, 2000.

[AEG98] P. K. Agarwal, J. Erickson, and L. J. Guibas. Kinetic BSPs for intersecting segments and disjoint triangles. In *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, pages 107–116, 1998.

[AG95] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *Internat. J. Comput. Geom. Appl.*, 5:75–91, 1995.

[AGHV97] P. K. Agarwal, L. J. Guibas, J. Hershberger, and E. Veach. Maintaining the extent of a moving point set. In *Proc. 5th Workshop Algorithms Data Struct.*, volume 1272 of *Lecture Notes Comput. Sci.*, pages 31–44. Springer-Verlag, 1997.

[AGSS89] A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete Comput. Geom.*, 4(6):591–604, 1989.

[AH01] P. K. Agarwal and S. Har-Peled. Maintaining the approximate extent measures of moving points. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, pages 148–157, 2001.

[AHH+99] Y. Aharoni, D. Halperin, I. Hanniel, S. Har-Peled, and C. Linhart. On-line zone construction in arrangements of lines in the plane. In *Proc. Workshop on Algorithm Engineering*, pages 139–153, 1999.

[AHK00] P. K. Agarwal, S. Har-Peled, and M. Karia. Computing approximate shortest paths on convex polytopes. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 270–279, 2000.

[AHSV97] P. K. Agarwal, S. Har-Peled, M. Sharir, and K. R. Varadarajan. Approximate shortest paths on a convex polytope in three dimensions. *J. Assoc. Comput. Mach.*, 44:567–584, 1997.

[AK00a] N. Amenta and R. Kolluri. Accurate and efficient unions of balls. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 119–128, 2000.

[AK00b] F. Aurenhammer and R. Klein. Voronoi diagrams. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 201–290. Elsevier Science Publishers B. V. North-Holland, Amsterdam, 2000.

[AKW01] H. Alt, C. Knauer, and C. Wenk. Bounding the Fréchet distance by the hausdorff distance. In *Proc. 17th European Workshop on Computational Geometry*, pages 166–169, 2001.

[AMN+98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. Assoc. Comput. Mach.*, 45(6), 1998.

[AP00] P. K. Agarwal and C. M. Procopiuc. Approximation algorithms for projective clustering. In *Proc. 11th ACM-SIAM Sympos. Discrete Algorithms*, pages 538–547, 2000.

[APR01] J. Abello, P. M. Pardalos, and M. G.C. Resende, editors. *Handbook of Massive Data Sets*. Kluwer Academic Publishers, 2001. to appear.

[Aro98] S. Arora. Polynomial time approximation schemes for euclidean tsp and other geometric problems. *J. Assoc. Comput. Mach.*, 45(5):753–782, Sep 1998.

[AS96] P. K. Agarwal and M. Sharir. Efficient randomized algorithms for some geometric optimization problems. *Discrete Comput. Geom.*, 16:317–337, 1996.

[AS98] P. K. Agarwal and S. Suri. Surface approximation and geometric partitions. *SIAM J. Comput.*, 27(4):1016–1035, 1998.

[AS00] P. K. Agarwal and M. Sharir. Pipes, cigars, and kreplach: The union of minkowski sums in three dimensions. *Discrete Comput. Geom.*, 24:645–657, 2000.

[AST94] P. K. Agarwal, M. Sharir, and S. Toledo. Applications of parametric searching in geometric optimization. *J. Algorithms*, 17:292–318, 1994.

[Ata85] M. J. Atallah. Some dynamic computational geometry problems. *Comput. Math. Appl.*, 11(12):1171–1181, 1985.

[Aur91] F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23:345–405, 1991.

[AWY+99] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park. Fast algorithms for projected clustering. In *SIGMOD 99*, pages 61–72, 1999.

[BAH99] A. Bremler-Barr, Y. Afek, and S. Har-Peled. Routing with a clue. In *Ann. Conf. on Data Comm. (SIGCOMM)*, pages 203–214, 1999.

[BBP98] H. Brönnimann, C. Burnikel, and S. Pion. Interval arithmetic yields efficient dynamic filters for computational geometry. In *Proc. 14th Annu. ACM Sympos. Comput. Geom.*, pages 165–174, 1998.

[BE97] M. Bern and D. Eppstein. Approximation algorithms for geometric problems. In D. S. Hochbaum, editor, *Approximationg algorithms for NP-Hard problems*, pages 296–345. PWS Publishing Company, 1997.

[BG95] H. Brönnimann and M. T. Goodrich. Almost optimal set covers in finite VC-dimension. *Discrete Comput. Geom.*, 14:263–279, 1995.

[BGH97] J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 747–756, 1997.

[BH01] G. Barequet and S. Har-Peled. Efficiently approximating the minimum-volume bounding box of a point set in three dimensions. *J. Algorithms*, 38:91–109, 2001.

[BL98] I. Bárány and D. G. Larman. The convex hull of the integer points of a large ball. *Math. Ann.*, 312(1):167–181, 1998.

[CCFM97] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *Proc. 29th Annu. ACM Sympos. Theory Comput.*, pages 626–635, 1997.

[Cha96] T. M. Chan. Output-sensitive results on convex hulls, extreme points, and related problems. *Discrete Comput. Geom.*, 16:369–387, 1996.

[Cha98] T. M. Chan. Approximate nearest neighbor queries revisited. *Discrete Comput. Geom.*, 20:359–373, 1998.

[Cha00] T. M. Chan. Approximating the diameter, width, smallest enclosing cylinder and minimum-width annulus. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 300–309, 2000.

[CK95] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. Assoc. Comput. Mach.*, 42:67–90, 1995.

[CKMN01] M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, pages 642–651, 2001.

[CKS+98] L. P. Chew, K. Kedem, M. Sharir, B. Tagansky, and E. Welzl. Voronoi diagrams of lines in 3-space under polyhedral convex distance functions. *J. Algorithms*, 29(2):238–255, 1998.

[Cla93] K. L. Clarkson. Algorithms for polytope covering and approximation. In *Proc. 3th Workshop Algorithms Data Struct.*, volume 709 of *Lecture Notes Comput. Sci.*, pages 246–252. Springer-Verlag, 1993.

[Cla94] K. L. Clarkson. An algorithm for approximate closest-point queries. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 160–164, 1994.

[DHS01] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, New York, 2nd edition, 2001.

[DP73] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer*, 10(2):112–122, December 1973.

[Dud74] R. M. Dudley. Metric entropy of some classes of sets with differentiable boundaries. *J. Approx. Theory*, 10(3):227–236, 1974.

[DV01] J. Dunagan and S. Vempala. Optimal outlier removal in high-dimensional spaces. In *Proc. 33rd Annu. ACM Sympos. Theory Comput.*, 2001. to appear.

[EFNN89] H. Ebara, N. Fukuyama, H. Nakano, and Y. Nakanishi. Roundness algorithms using the Voronoi diagrams. In *Proc. 1rd Canad. Conf. Comput. Geom.*, page 41, 1989.

[EG89] H. Edelsbrunner and L. J. Guibas. Topologically sweeping an arrangement. *J. Comput. Syst. Sci.*, 38:165–194, 1989. Corrigendum in 42 (1991), 249–251.

[EGH+00] A. Efrat, L. J. Guibas, S. Har-Peled, , D. C. Lin, J. S.B. Mitchell, and T. M. Murali. Sweeping simple polygons with a chain of guards. In *Proc. 11th ACM-SIAM Sympos. Discrete Algorithms*, pages 927–936, 2000.

[EGHM01] A. Efrat, L. J. Guibas, S. Har-Peled, and T. M. Murali. Morphing between polylines. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, pages 680–689, 2001.

[EGS86] H. Edelsbrunner, L. J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM J. Comput.*, 15(2):317–340, 1986.

[Eri95] J. Erickson. New lower bounds for Hopcroft's problem. In *Proc. 11th Annu. ACM Sympos. Comput. Geom.*, pages 127–137, 1995.

[Eri01] J. Erickson. Nice points sets can have nasty delaunay triangulations. In *Proc. 17th Annu. ACM Sympos. Comput. Geom.*, pages 96–105, 2001.

[FG88] T. Feder and D. H. Greene. Optimal algorithms for approximate clustering. In *Proc. 20th Annu. ACM Sympos. Theory Comput.*, pages 434–444, 1988.

[FL95] C. Faloutsos and K. I. Lin. FastMap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 24(2):163–174, June 1995.

[For97] S. Fortune. Voronoi diagrams and Delaunay triangulations. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 20. CRC Press LLC, Boca Raton, FL, 1997.

[Fré06] M. Fréchet. Sur quelques points du calcul fonctionnel. *Rendiconti del Circolo Matematico di Palermo*, 22:1–74, 1906.

[Gae98] B. Gaertner. Exact arithmetic at low cost - A case study in linear programming. In *Proc. 9th ACM-SIAM Sympos. Discrete Algorithms*, pages 157–166, 1998.

[Gär95] B. Gärtner. A subexponential algorithm for abstract optimization problems. *SIAM J. Comput.*, 24:1018–1035, 1995.

[GG98] V. Gaede and O. Günther. Multidimensional access methods. *ACM Comput. Surv.*, 30:170–231, 1998.

[GGH+01] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. In *Proc. 17th Annu. ACM Sympos. Comput. Geom.*, pages 188–196, 2001.

[GLR97] J. García-Lopez and P. Ramos. Fitting a set of points by a circle. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 139–146, 1997.

[Gon85] T. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.*, 38:293–306, 1985.

[Gru93] P. M. Gruber. Aspects of approximation of convex bodies. In Peter M. Gruber and J. M. Wills, editors, *Handbook of Convex Geometry*, volume A, chapter 1.10, pages 319–345. North-Holland, Amsterdam, Netherlands, 1993.

[Gui98] L. J. Guibas. Kinetic data structures — a state of the art report. In P. K. Agarwal, L. E. Kavraki, and M. Mason, editors, *Proc. 5th Workshop Algorithmic Found. Robot.*, pages 191–209. A. K. Peters, Wellesley, MA, 1998.

[Har99a] S. Har-Peled. Approximate shortest paths and geodesic diameters on convex polytopes in three dimensions. *Discrete Comput. Geom.*, 21:216–231, 1999.

[Har99b] S. Har-Peled. Constructing approximate shortest path maps in three dimensions. *SIAM J. Comput.*, 28(4):1182–1197, 1999.

[Har00] S. Har-Peled. Taking a walk in a planar arrangement. *SIAM J. Comput.*, 30(4):1341–1367, 2000.

[Har01a] S. Har-Peled. Clustering motion. In *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 84–93, 2001.

[Har01b] S. Har-Peled. A practical approach for computing the diameter of a point-set. In *Proc. 17th Annu. ACM Sympos. Comput. Geom.*, pages 177–186, 2001.

[Har01c] S. Har-Peled. A replacement for voronoi diagrams of near linear size. In *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 94–103, 2001.

[HG97] P. S. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. Technical report, CMU-CS, 1997. http://www.uiuc.edu/~garland/papers.html.

[HKK92] D. Huttenlocher, K. Kedem, and J. Kleinberg. On dynamic Voronoi diagrams and the minimum Hausdorff distance for point sets under Euclidean motion in the plane. In *Proc. 8th Annu. ACM Sympos. Comput. Geom.*, pages 110–119, 1992.

[HKL97] D. Halperin, L. E. Kavraki, and J.-C. Latombe. Robotics. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 41, pages 755–778. CRC Press LLC, Boca Raton, FL, 1997.

[HS99] J. Hershberger and S. Suri. Kinetic connectivity of rectangles. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 237–246, 1999.

[HS01] J. Hershberger and S. Suri. Simplified kinetic connectivity for rectangles and hypercubes. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, pages 158–167, 2001.

[HV01] S. Har-Peled and K. R. Varadarajan. Approximate shape fitting via linearization. In *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 66–73, 2001.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annu. ACM Sympos. Theory Comput.*, pages 604–613, 1998.

[JMN99] H. V. Jagadish, J. Madar, and R. T. Ng. Semantic compression and pattern extraction with fascicles. In M. P. Atkinson, M. E. Orlowska, P. Valduriez, S. B. Zdonik, and M. L. Brodie, editors, *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 186–198. Morgan Kaufmann, 1999.

[KG01] M. I. Karavelas and L. J. Guibas. Static and kinetic geometric spanners with applications. In *Proc. 12th ACM-SIAM Sympos. Discrete Algorithms*, pages 168–176, 2001.

[KOR98] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. In *Proc. 30th Annu. ACM Sympos. Theory Comput.*, pages 614–623, 1998.

[KVV00] R. Kannan, S. Vempala, and A. Vetta. On clusterings: good, bad and spectral. In *Proc. 41th Annu. IEEE Sympos. Found. Comput. Sci.*, 2000.

[Lin00] P. Lindstrom. Out-of-core simplification of large polygonal models. In Kurt Akeley, editor, *Proc. SIGGRAPH 2000*, pages 259–262. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[LL91] V. B. Le and D. T. Lee. Out-of-roundness problem revisited. *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-13(3):217–223, 1991.

[LPC+00] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3D scanning of large statues. In Kurt Akeley, editor, *Proc. SIGGRAPH 2000*, pages 131–144. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[MP99] M. E. Munich and P. Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In *Proc. of the 7th Inter. Conf. on Computer Vision*, pages 108–115, 1999.

[MSY97] K. Mehlhorn, T. C. Shermer, and C. K. Yap. A complete roundness classification procedure. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 129–138, 1997.

[PAH01] C. M. Procopiuc, P. K. Agarwal, and S. Har-Peled. STAR-Tree: An efficient self-adjusting index for moving objects. menuscript, 2001.

[PS85] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, NY, 1985.

[Ram01] E. Ramos. Smooth surface reconstruction in near linear time. menuscript, 2001.

[Riv79] T. J. Rivlin. Approximating by circles. *Computing*, 21:93–104, 1979.

[RL00] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. In Kurt Akeley, editor, *Proc. SIGGRAPH 2000*, pages 343–352. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[RLW91] U. Roy, C. R. Liu, and T. C. Woo. Review of dimensioning and tolerancing: Representation and processing. *Comput. Aided Design*, 23(7):466–483, 1991.

[RZ92] U. Roy and X. Zhang. Establishment of a pair of concentric circles with the minimum radial separation for assessing roundness error. *Comput. Aided Design*, 24(3):161–168, 1992.

[SA95] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, New York, 1995.

[SA01] G. Song and N. M. Amato. Using motion planning to study protein folding pathways. In *RECOMB 2001*, pages 287–296, 2001.

[Sch99] S. Schirra. Robustness issuess. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*. Elsevier Science Publishers B. V. North-Holland, Amsterdam, 1999. to appear.

[SG01] E. Shaffer and M. Garland. Efficient adaptive simplification of massive meshes. menuscript, 2001.

[SJ99] M. Smid and R. Janardan. On the width and roundness of a set of points in the plane. *Int. J. Comput. Geom. Appls.*, 9(1):97–108, 1999.

[SJLL00] S. Saltenis, C. S. Jensen, S. Leutenegger, and M. Lopez. Indexing the positions of continuously moving objects. In *ACM SIGMOD Int. Conf. on Manag. of Data*, pages 331–342, 2000.

[SWCD97] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and querying moving objects. In *Proc. Intl Conf. Data Engineering*, pages 422–432, 1997.

[SY95] T. C. Shermer and C. K. Yap. Probing for near centers and relative roundness. In *Proc. ASME Workshop on Tolerancing and Metrology*, 1995.

[TPK01] M. L. Teodoro, G. N. Phillips, and L. E. Kavraki. Molecular docking: A problem with thousands of degrees of freedom. In *Proc. 2001 IEEE Internat. Conf. Robot. Autom.*, Seoul, Korea, 2001.

[YC97] C. K. Yap and E.-C. Chang. Issues in the metrology of geometric tolerancing. In J.-P. Laumond and M. H. Overmars, editors, *Robotics Motion and Manipulation*, pages 393–400. A. K. Peters, 1997.

[ZE00] A. Zomorodian and H. Edelsbrunner. Fast software for box intersections. In *Proc. 16th Annu. ACM Sympos. Comput. Geom.*, pages 129–138, 2000.