Polygonal Path Approximation: A Query Based Approach

Ovidiu Daescu* and Mi Ningfang
Department of Computer Science
University of Texas at Dallas
Richardson, TX 75080, USA
{daescu,ningfangmi}@utdallas.edu

1 Introduction

A polygonal path or chain P, with n vertices, is defined as an ordered set (p_1, p_2, ..., p_n) of vertices, such that any two consecutive vertices p_i, p_{i+1} are connected by the line segment $\overline{p_i p_{i+1}}$, for $1 \le i < n$. The polygonal path approximation problem asks to approximate a general polygonal path P by another polygonal path P' whose vertices are constrained to form an ordered subset of the vertices of P. Polygonal path approximation appears as a subproblem in many applications in geographic information systems (GIS), cartography, computer graphics, medical imaging and data compression. Efficient solutions to this problem may lead to better/faster solutions for more complex problems. For example, with recent imaging technologies, medical and satellite images have become very reach in details. Analyzing such images or using them in interactive remote WWW-based applications may require a significant amount of time. Thus, path approximation algorithms can be used to simplify/compress those images, for the purpose of efficient storage or for progressive transmission to a remote site. Similar observations apply to map representation in geographic information systems, where in many situations (animations, etc.) a high level of detail is unnecessary or even unwanted.

1.1 Problem Definition

Given a polygonal path $P = (p_1, p_2, \ldots, p_n)$, the polygonal path approximation problem is to find a path $P' = (p_{i_1} = p_1, p_{i_2}, \ldots, p_{i_m} = p_n)$ such that, for each $j \in \{1, 2, \ldots, m-1\}$: (i) $i_j < i_{j+1}$, where $i_j, i_{j+1} \in \{1, 2, \ldots, n\}$; (ii) subpath $(p_{i_j}, p_{i_j+1}, \ldots, p_{i_{j+1}})$ of P is contained in some *error tolerance* region associated with the line segment $\overline{p_{i_j} p_{i_{j+1}}}$. We will also consider the case when the vertices of the approximation are not restricted to a subset of the original path.

The *error tolerance* region of a line segment $\overline{p_i p_j}$ is defined by an error measure criterion. A few error criteria have been used in solving various polygonal path approximation problems, not necessarily of the form above [1–3,5,8–10,12,13,16,17,21–28,30]. If not otherwise specified, we consider the error criterion used in [9, 16, 23, 30], called the *infinite beam* or *parallel-strip* criterion, with the Euclidean, L_2 measure of distance. With this criterion, the ϵ-tolerance

*Corresponding author.

region of a line segment $\overline{p_i p_j}$ is the set of points that are within distance ϵ from the line $L(\overline{p_i p_j})$ supporting the line segment $\overline{p_i p_j}$. If a subpath $P_{i_j,i_{j+1}} = (p_{i_j}, p_{i_j+1}, \ldots, p_{i_{j+1}})$ of P is contained in the ϵ-tolerance region of line segment $\overline{p_{i_j} p_{i_{j+1}}}$ of P', then $\overline{p_{i_j} p_{i_{j+1}}}$ is an ϵ-*approximating* segment for $P_{i_j,i_{j+1}}$. A path P' is an ϵ-*approximation* of P if each line segment $\overline{p_{i_j} p_{i_{j+1}}}$ of P', for $j = 1, 2, \ldots, m-1$, is an ϵ-*approximating* segment.

Other commonly used error criteria are the tolerance zone and uniform measure. With the *tolerance zone* criterion [21, 23, 26], the ϵ-tolerance region of a line segment $\overline{p_i p_j}$ is the region of space that is the union of all radius-ϵ disks centered at points along the segment $\overline{p_i p_j}$. With the uniform measure criterion [2], the error of a line segment $\overline{p_i p_j}$ is defined as $\max\{d(p_k, \overline{p_i p_j}) \mid i \leq k \leq j\}$, where $d(p_k, \overline{p_i p_j})$ denotes the vertical distance between p_k and the segment $\overline{p_i p_j}$.

In this paper, we consider the optimization version of the path approximation problem, called the **min-# problem**: Given a polygonal path P and a positive approximation error ϵ, find an ϵ-approximating path P' of P with the smallest number of vertices.

While the tolerance zone criterion would produce a compressed version that better captures the features of the original path, the motivation for studying the problem under infinite beam criterion is two fold. The first reason is that the best known solutions under tolerance zone compute approximating lines or semilines in order to obtain approximating segments. Thus, some of the results we develop may be useful for tolerance zone. The second reason is that infinite beam criterion gives a better degree of compression.

Lemma 1 *Let P' be an ϵ-approximating path of P with tolerance zone criterion. Then P' is an ϵ-approximating path of P with infinite beam criterion.*

1.2 Previous Work

Early results for the min-# problem, under various error criteria, were presented by Imai and Iri [21–23], Melkman and O'Rourke [26] and Toussaint [30]. In particular, Imai and Iri [23] have formulated the problem in terms of graph theory: construct a path approximation, unweighted directed acyclic graph and then use breadth first search to compute a shortest path in this graph. Their formalism have been later exploited by most of the algorithms devoted to the problem [2,5,8,9,16]. However, with the exception of Agarwal and Varadarajan algorithms [2], all the other algorithms have quadratic or superquadratic time. The algorithms in [2] combine the previous iterative graph based approach with divide and conquer and, using graph compression techniques and more complicated data structures, achieve $O(n^{4/3+\delta})$ time and space complexities, where $\delta > 0$ is an arbitrarily small constant. However, those algorithms work for L_1 distance metric and it has been left as an open problem in [2] to extend them to the more general, L_2 distance metric. Very recently, algorithms with running times that depend on the size of the output have been proposed in [11], by observing that only the edges of G that are needed by the shortest path computation in G need be computed. The most popular heuristic method that is used in path approximation, the recursive simplification heuristic of

Douglas and Peucker [14], can be implemented in $O(n \log^* n)$ time [19], but does not guarantee to find an optimal solution. If the vertices of the approximating path are not required to be a subset of the vertices of the input, then faster algorithms are possible [17, 18, 22]. In [1], near-linear time algorithms are proposed for computing an approximating path with vertices among those of P and size at most the size of a min-# approximating path with error $\epsilon/2$.

Solutions to polygonal subdivision simplification problem are also based on polygonal path simplification. In [12], polygonal path simplification has been used to approximate a subdivision S with N vertices and M extra points in $O(N(N+M)\log N)$ time. The simplification is only required to satisfy a set of constraints. If a minimum size simplification is sought, the problem becomes NP-hard [17] and, unless P=NP, one cannot obtain in polynomial time an approximation within a factor of $n^{1/5-\delta}$ of an optimal solution, for any $\delta > 0$ [15].

1.3 Our Results

Quadratic or near-quadratic time algorithms for min-# problem with L_2 distance metric have been developed almost two decades ago. However, no subquadratic time algorithms are known with L_2 metric and no nontrivial lower bound could be proved. In this paper we present a new, query based approach for solving the min-# problem that carries out the computation in a dual space. We give a few results related with this approach, some of more general interest, and propose a greedy heuristic to speed up the computation. In particular, our results imply an $O(n \log n)$ breadth first search procedure for some quadratic size graphs. The algorithms we propose are simple, based on standard geometric operations, and thus suitable for efficient implementation. We further give an $O(n \log n)$ time, factor 2 approximation algorithm with infinite beam criterion. Finally, we show that the dual space approach can be used to obtain a subquadratic time exact algorithm with infinite beam criterion and L_2 distance metric if some condition on the input path holds. Although in a special case, this is the first subquadratic result for path approximation with L_2 distance metric.

A basic version of our solution has been implemented and preliminary results are very promising when compared to previous algorithms.

2 Preliminaries

In this section, we briefly present the general structure of previous algorithms for solving the min-# problem. We exemplify the algorithms for the infinite beam criterion. To solve the min-# problem, those algorithms first build a path approximation, directed acyclic graph G. The vertices of G are associated with the vertices of P, and edges correspond to ϵ-approximating line segments for P. An optimal approximating path P' is obtained by a shortest path computation in G. In general, G has $O(n^2)$ complexity ($O(n^{4/3+\delta})$ in [2]). As observed in [9], in many cases one need not explicitly construct G: an optimal path can be computed while computing the ϵ-approximating segments. With this observation, the algorithms for computing the ϵ-approximating segments use an iterative, incremental approach similar to the following.

for $i = 1$ to $n - 1$ **do**
  while $D(p_{i+1}, \epsilon), D(p_{i+2}, \epsilon), \ldots, D(p_j, \epsilon)$ admit transversal through p_i **do**
    Check if $\overline{p_i p_j}$ is an ϵ-approximation; if yes, update path length at p_j (if needed).
    j=j+1.
  end while
end for

In this approach, the path length at a vertex of P could be updated multiple times, as the computation of ϵ-approximating edges proceeds. A more efficient approach [11] is to use a breadth first computation of the ϵ-approximating edges that simulates the breadth first shortest path computation in an ϵ-approximating graph G. The structure of the breath first traversal (BFT) approach is presented bellow, with *enqueue* and *dequeue* being the standard queue operations.

enqueue(1)
for $j = 2$ to n **do**
  $visit(j) = 0$.
end for
repeat
  i=dequeue().
  while $D(p_{i+1}, \epsilon), D(p_{i+2}, \epsilon), \ldots, D(p_j, \epsilon)$ admit line transversal through p_i **do**
    If $!visit(j)$, check if $\overline{p_i p_j}$ is an ϵ-approximating segment; if yes, $visit(j) = 1$, set path length at p_j and enqueue(j).
    j=j+1.
  end while
until p_n has been reached.

Lemma 2 *With the BFT approach, the shortest p_1-to-p_i path length at a vertex p_i of P is correctly computed when the vertex is enqueued and will not be updated by future computation.*

3 A Query Based Approach

From Lemma 2 above, it follows that with the BFT approach a vertex p_j of P that has been marked as visited need not be checked again in pairs (p_i, p_j), with $i < j$. Let p_i be the dequeued vertex by the BFT algorithm. Since it may happen that many vertices immediately following p_i have been already visited, we should avoid including those vertices in future computation associated with p_i. However, incremental approaches cannot avoid this computation. Then, it may be appropriate to combine the BFT algorithm with a query method for computing ϵ-approximating edges, rather than using the incremental method of computation. This may also be interesting for practical applications mentioned earlier, such as animation, where one may want to approximate various subpaths of P at different times. To this end, we divide the vertices of P in three categories: (1) *processed*: those visited by BFT that are no longer in the queue, (2) *active*: those visited by BFT that are in the queue and (3) *inactive*: those that have not yet been reached by the BFT. With a query method, one should be able to answer fast if $\overline{p_i p_j}$ is an ϵ-approximating segment, without first having to compute information for vertices p_k, where $i < k < j$. Specifically, the following two operations should be supported.

- $Span(i)$: compute the largest index j such that $\overline{p_i p_j}$ could be an ϵ-approximating segment. Index j is the largest index such that disks $D(p_{i+1}, \epsilon), D(p_{i+2}, \epsilon), \ldots, D(p_j, \epsilon)$ admit a line transversal through p_i.

- $Query(i, j)$: answer if $\overline{p_i p_j}$ is an ϵ-approximating segment.

Let V_1, V_2 and V_3 be the sets of processed, active and inactive vertices, and consider the BFT algorithm at some moment of time. The queue associated with the BFT contains a set of vertices such that the shortest path lengths of any two of them differ by at most one. In other words, all vertices can be reached from p_1 with $k-1$ or k links, for some integer $2 \le k \le n-2$. If p_1^i denotes the shortest p_1-to-p_i path length at p_i, then $p_1^i \le k$ if $p_i \in V_1 \cup V_2$ and $p_1^i \ge k$ if $p_i \in V_3$. We further augment the BFT algorithm with a greedy approach: vertices in V_2 are maintained in two priority queues having as keys the indices of vertices of P, such that a dequeue operation on each of the queues will return the largest index in the queue. The first priority queue corresponds to vertices V_2^1 that can be reached with $k-1$ links and the second to those vertices V_2^2 reachable with k links.

Lemma 3 *The overall time to maintain the two priority queues is $O(f(m)\log(f(m)))$, where $f(m)$ is the number of vertices of P that can be reached from p_1 with no more than $m-1$ ϵ-approximating segments and m is the number of vertices of a min-# approximating path.*

Observe that it is possible to have $p_i \in (V_1 \cup V_2)$ and $p_j \in V_3$ such that $i > j$. However, only pairs of the form (i, j), with $p_i \in V_2^1$, $p_j \in V_3$ and $i < j$ should be considered for $Query(i, j)$ operations. Thus, we need to maintain the set of inactive vertices such that, when a vertex p_i is dequeued from the BFT queue, the inactive vertices p_j, with $j > i$, are easily available.

Lemma 4 *Assuming $Span(i)$ is known, the inactive vertices for p_i can be found in $O(\log I + k)$ time, where I is the set of currently inactive vertices and k is the number of inactive vertices in I with indices between i and $Span(i)$. The set I of inactive vertices can be maintained in $O(n + f(m)\log n)$ time, using $O(n)$ space.*

Proof. Use a balanced binary search tree T_b on the inactive vertices. Initially, p_1 is in the queue and $p_2, p_3, \ldots, p_n$ are in T_b. The keys in T_b are the indices of the vertices in T_b. Assume that p_i is dequeued from the queue. We search T_b to find the first vertex with index larger than i. Then, the inactive vertices up to $Span(i)$ can be found by a simple traversal in T_b. If one of the visited vertices p_j becomes active (as a result of $Query(i, j)$), we remove the vertex from T_b. Then, we obtain the claimed time and space bounds. □

From Lemma 3 and Lemma 4 above it follows that the time complexity of the greedy BFT algorithm depends on the time complexities for performing $Span(\cdot)$ and $Query(\cdot,\cdot)$ operations and the number of failed $Query(\cdot,\cdot)$ operations. We can extend Lemma 4 as follows. Let G be an unweighted directed graph with n vertices, such that for each $v_i \in G$, there are edges $(v_i, v_{i_1}), (v_i, v_{i_1+1}), (v_i, v_{i_1+2}), \ldots, (v_i, v_{i_2}) \in G$. That is, G can be specified by its set of vertices and the index ranges (i_1, i_2) associated with each vertex $v_i \in G$. If a standard representation is used for storing G, $O(n^2)$ space is required, since G can have $O(n^2)$ edges. Computing a single source shortest path in G using standard breadth first search would take $O(n^2)$ time.

Lemma 5 *A single source shortest path tree in a graph G specified as above can be computed in $O(n \log n)$ time and $O(n)$ space.*

Let $d(p_i, p_j)$ denote the Euclidean distance between p_i and p_j. The following lemma bounds the time complexities of $Span(i)$ and $Query(i,j)$ operations.

Lemma 6 *For any pair (p_i, p_j), with $1 \le i < j \le n$, $Query(i,j)$ can be answered in $O(\log n)$ time. If $d(p_i, p_j) \notin [\epsilon, \epsilon\sqrt{2}]$, for $1 \le i < j \le n$, then $Span(i)$ can be answered in $O(\log n)$ time.*

To prove this lemma we first introduce some geometric structures. Let p_i and p_j, with $i < j$, be two vertices of P. For a vertex $p_k \in P$, with $i < k < j$, let σ_k denote the set of lines that are tangent to $D(p_k, \epsilon)$. Using a standard point-line duality transform, σ_k is mapped to a hyperbola H_k in the dual plane [20]. Any vertical line in the dual plane intersects each of the two branches of H_k exactly once; the upper branch H_k^a corresponds to tangents to the upper semicircle of $D(p_k, \epsilon)$ and the lower branch H_k^b corresponds to tangents to the lower semicircle. Each branch is an x-monotone and unbounded Jordan curve and $H_k^a \cap H_k^b = \Phi$. The dual of a line transversal of $D(p_k, \epsilon)$ corresponds to a point bellow H_k^a and above H_k^b. Observe that any pair (H_i^c, H_j^d) of hyperbolic branches intersects in at most one point, where $i \ne j$ and $c, d \in \{a, b\}$. This it true only for equal-radius disks; for different radii disks each pair can intersect in at most two points.

Let $\mathcal{L}_{ij}$ be the *lower envelope* of H_k^a and let $\mathcal{U}_{ij}$ be the *upper envelope* of H_k^b, for $k = i, i+1, \ldots, j$. Then, $L(\overline{p_i p_j})$ is a common transversal of $\{D(p_{i+1}, \epsilon), D(p_{i+2}, \epsilon), \ldots, D(p_{j-1}, \epsilon)\}$ (line segment $\overline{p_i p_j}$ is a valid approximation segment) iff its dual point lies between $\mathcal{L}_{ij}$ and $\mathcal{U}_{ij}$. Using the results in [29], it follows that the complexities of $\mathcal{L}_{ij}$, $\mathcal{U}_{ij}$ and the region $\mathcal{I}_{ij}$ sandwiched between them are $O(j-i)$ and that $\mathcal{I}_{ij}$ has at most one connected component. The last property is not true for different radii disks, in which case there could be $O(j-i)$ connected components. Since for a set of n equal-radius disks, the space of line transversals that are restricted to pass through a common point p can have $O(n)$ connected components [9,11], the line dual to p_i can have $O(j-i)$ disjoint segments in $\mathcal{I}_{ij}$. The observation also holds for L_1 distance metric and in general for any distance metric in which the resulting tolerance error regions of path vertices have the structure of congruent convex disks. This is a key property that makes the min-# problem with infinite beam criterion somehow harder than with other error criteria (such as tolerance zone and uniform measure).

If we use a vertical error measure however, where the error region of a point is a vertical line segment instead of a disk, the space of line transversals that are restricted to pass through a common point can be easily seen to have at most one connected component.

Lemma 7 *The min-# problem with infinite beam criterion and vertical error measure can be solved in* $O(n^{4/3+\delta})$ *time and space, where* $\delta > 0$ *is an arbitrarily small constant.*

Proof. The result can be obtained by a simple modification of Agarwal and Varadarajan [2] min-# algorithm for x-monotone paths with uniform metric. For infinite beam criterion and vertical distance metric, the difference is that right (or left) oriented rays are replaced by lines (the path P is not monotone) and thus wedges are replaced by double wedges. The only places where this plays a role are in the computation of the lower and upper convex hulls (of the upper and lower endpoints of the vertical error segments) and in the computation of the double cone at p_i. The double cone at p_i can be found by computing the largest separating double cone of the two convex hulls, if such a separation is possible. This reduces to computing the tangents from p_i to the two convex hulls. Note that the convex hulls are used only to help compute the double cone at p_i, since only the double cone at p_i is important for the outcome of the algorithm. Then, to perform the incremental updating of the convex hulls and to compute the tangents in altogether $O(\log n)$ time, as in [2], it suffices to use the solution in [16], based on the on-line convex hull algorithm of Avis *et al.* [4]. □

In what follows, we set up the data structures for computing $Span(i)$ and $Query(i,j)$. Let T be a complete binary tree such that the leaves of T are associated, in order, with the vertices of P. At each leaf l_i, we also store the hyperbolic branches H_i^a and H_i^b, the duals of the tangent lines to $D(p_i, \epsilon)$. For an internal node v in T, let i and j be the smallest and largest indices of the leaf descendants of v. We store at v the region $I(v) = \mathcal{I}_{ij}$ sandwiched between the lower envelope $\mathcal{L}_{ij}$ of $\{H_k^a \mid i \leq k \leq j\}$ and the upper envelope $\mathcal{U}_{ij}$ of $\{H_k^b \mid i \leq k \leq j\}$. Since any two hyperbolic branches intersect each other at most once, $\mathcal{L}_{ij}$ and $\mathcal{U}_{ij}$ have complexity $O(j-i)$ [29]. We compute $I(v)$ for all vertices v of T, using a divide-and-conquer method implemented by a bottom up traversal of T. The boundary of $I(v)$ is maintained as two monotone pieces, corresponding to the contributions of the lower and upper envelopes of the two sets of hyperbolic branches stored at the leaf descendants of v. Following similar arguments as in [5], $I(v)$ can be computed, for all vertices v in T, in a total of $O(n \log n)$ time. Note that in general the region $I(r)$ at the root r of T may be empty. We can use T to compute, in the same time bound, a family of trees $\mathcal{F} = \{T_1, T_2, \ldots, T_s\}$ such that T_1's leaves correspond to the longest possible prefix $D_1 = \{D(p_1,\epsilon), D(p_2,\epsilon), \ldots, D(p_{j_1},\epsilon)\}$ that admits a line transversal, T_2's leaves correspond to the longest possible prefix starting at p_{j_1+1}, and so on. This implies the following result.

Lemma 8 *$\mathcal{F}$ can be used to compute an approximating path with vertices inside the error tolerance regions of vertices of P and of size at most twice the size of an optimal approximation.*

Proof. For each $T_i \in \mathcal{F}$, pick a point in the intersection stored at the root of T_i and trim the dual line to a line segment with endpoints inside the first and last leaf of T_i, resulting in $|\mathcal{F}|$ line segments. Then join the resulting segments to form a path, which adds $|\mathcal{F}| - 1$ segments. Obviously, an optimal approximating path must have at least $|\mathcal{F}|$ vertices, from which the claim follows. Once $\mathcal{F}$ is available, the construction takes $O(|\mathcal{F}|)$ time. □

Note that our solution is different from greedy solutions in [1,17,26], which consider wedges instead of double wedges and/or may depend on some predefined stabbing order. Observe also that an $O(\log n)$ factor approximation can be obtained directly from T. Finally, it can be easily seen that the same results hold for other distance metrics, such as L_1 metric. We are now ready to give the proof for Lemma 6.

Proof (Lemma 6). Let p_{ij} be the dual of line $L(\overline{p_i p_j})$. To perform $Query(i,j)$ operation, we form a search path π_{ij} in $\mathcal{F}$, from leaf l_i to leaf l_j. Observe that l_i and l_j may be in the same tree or in different, adjacent trees of $\mathcal{F}$. We say that a node v of T is on the *right fringe* (*left fringe*) of π_{ij} if v is not on π_{ij} but is a right (left) child of a node on π_{ij}. Noting that the query is a decomposable search problem, we can answer $Query(i,j)$ by determining if p_{ij} is inside $I(v)$ for each v that is a right (left) fringe node of π_{ij} (answering "true" if and only if it is inside them all). Using the approach in [5], based on fractional cascading technique of Chazelle and Guibas [6, 7], the answer for $Query(i,j)$ can be obtained in $O(\log n)$ time. $Span(i)$ computation proceeds in a similar way. We form a search path π_i in $\mathcal{F}$ as follows. We start at l_i and go up the corresponding tree T_{l_i} storing l_i until we find a node with nonempty intersection on right fringe or we reach the root of T_{l_i}. If the root has been reached, we go on a downward path on T_{l_i+1} (if exists) until we find a node with nonempty intersection on left fringe. Let v be the root of that subtree (right or left fringe) and let $l(p_i)$ be the dual line of p_i. We then compute the intersection of $l(p_i)$ with $I(v)$. The condition in the lemma assures that $l(p_i) \cap I(v)$ has at most one connected component. If $v \in T_{l_i}$ and $l(p_i) \cap I(v) \neq \Phi$, set $l(p_i) = l(p_i) \cap I(v)$. Then go up T_{l_i} until a node w on right fringe of π_i with with $l(p_i) \cap I(v) = \Phi$ is found or the root of T_{l_i} has been reached, while updating $l(p_i)$ along the way. Then follow a downward path (possibly in T_{l_i+1}) until a node with nonempty left fringe is found. We then proceed down the tree as follows. Let v be the current node and let v_l and v_r be the left and right children of v. If $l(p_i) \cap I(v_l) \neq \Phi$ proceed on right fringe of v with $l(p_i) = l(p_i) \cap I(v_l)$, else proceed on left fringe of v with $l(p_i)$. The computation stops at some leaf l_j. Clearly, $Span(i) = j - 1$. The computation for $Span(i)$ at a node in the tree is similar to that for $Query(i,j)$, and thus $Span(i)$ can be answered in the same amount of time. □

The results in Lemma 3, Lemma 4 and Lemma 6 can be combined to obtain an efficient algorithm for solving the min-# problem. Alternatively, we have the following result.

Lemma 9 *Under the condition in Lemma 6, the min-# problem with infinite beam criterion and L_2 metric can be solved in $O(n^{4/3+\delta})$ time, where $\delta > 0$ is an arbitrarily small constant.*

Proof. We only sketch the proof for this lemma. Apply the divide and conquer procedure in [2], enhanced with the forest $\mathcal{F}$ and use $\mathcal{F}$ to compute the line segments (duals of double wedges) required by the procedure. This last computation is similar to that of $Span(i)$. If $p_{n/2}$ is in some $T_k \in \mathcal{F}$, then we only need consider vertices associated with T_k and the predecessor and successor of T_k in $\mathcal{F}$. $\square$

Although in a special case, this is the first subquadratic result for path approximation with L_2 distance metric. The result is useful especially when the algorithm is used to approximate borders of fat planar regions and terrains, and in medical imaging, where the condition stated in the lemma may be often satisfied. The result also holds for L_1 distance metric and in general for any distance metric in which the resulting tolerance error regions of path vertices have the structure of congruent convex disks.

The basic BFT algorithm has been implemented for tolerance zone error and preliminary experiments on randomly generated polygonal paths have been conducted. The experiments show BFT algorithm is significantly faster than the iterative one. The difference in running times increases as ϵ increases. This corresponds to the intuition that BFT should be much faster if coarser approximations are sought. Implementation of the query based, greedy BFT version described in this paper is under way. We plan to test that version on some benchmark data and compare with other heuristics [1, 14]. We conjecture that our results, under the condition of Lemma 6, can be extended to tolerance zone error. It remains an open problem to extend them for general paths while maintaining subquadratic time.

References

[1] P.K. Agarwal, S. Har-Peled, N. Mustafa, and Y. Wang. Near-linear time approximation algorithms for curve simplification. *Proc. of 10th Annual European Sympos. Algorithms*, pages 29–41, 2002.

[2] P.K. Agarwal and K.R. Varadarajan. Efficient algorithms for approximating polygonal chains. *Discrete Computational Geometry*, 23:273–291, 2000.

[3] H. Alt, J. Blomer, M. Godau, and H. Wagener. Approximation of convex polygons. *Proc. 10th Coll. on Autom., Lang., and Prog. (ICALP)*, pages 703–716, 1990.

[4] D. Avis, H. ElGindy, and R. Seidel. Simple on-line algorithms for convex polygons. *Computational Geometry*, G.T. Toussaint, Ed., 1985.

[5] G. Barequet, D.Z. Chen, O. Daescu, M.T. Goodrich, and J. Snoeyink. Efficiently approximating polygonal paths in three and higher dimensions. *Proc. of the 14th ACM Symp. on Comp. Geometry*, pages 317–326, 1998.

[6] B.Chazelle and L.J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1:133–162, 1986.

[7] B.Chazelle and L.J. Guibas. Fractional cascading: II. Applications. *Algorithmica*, 1:163–191, 1986.

[8] W.S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments or minimum error. *Intl. Journal of Computational Geometry and Applications*, 6(1):59–77, 1996.

[9] D.Z. Chen and O. Daescu. Space-efficient algorithms for approximating polygonal curves in two dimensional space. *Proc. of the 4th Annual International Computing and Combinatorics Conference*, pages 55–64, 1998.

[10] L.P. Cordella and G. Dettori. An $O(n)$ algorithm for polygonal approximation. *Pattern Recognition Letters*, 3:93–97, 1985.

[11] O. Daescu. New results on path approximation. *Algorithmica*, special issue on shape algorithmics. To appear.

[12] M. de Berg, M. van Kreveld, and S. Schirra. Topologically correct subdivision simplification using the bandwidth criterion. *Cartography and GIS*, 25:243–257, 1998.

[13] G. Dettori. An on-line algorithm for polygonal approximation of digitized plane curves. *Proc. 6th Int. Conf. on Pattern Recognition*, pages 18–22, Munich, Oct. 1982.

[14] 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, 1973.

[15] R. Estkowski and J.S.B. Mitchell. Simplifying a polygonal subdivision while keeping it simple. *Proc. 17th ACM Symposium on Computational Geometry*, pages 40–49, 2001.

[16] D. Eu and G.T. Toussaint. On approximation polygonal curves in two and three dimensions. *CVGIP: Graphical Models and Image Processing*, 56(3):231–246, 1994.

[17] L.J. Guibas, J.E. Hershberger, J.S.B. Mitchell, and J.S. Snoeyink. Approximating polygons and subdivisions with minimum link paths. *International Journal of Computational Geometry and Applications*, 3(4):383–415, 1993.

[18] S.L. Hakimi and E.F. Schmeichel. Fitting polygonal functions to a set of points in the plane. *CVGIP: Graphical Models and Image Processing*, 53(2):132–136, 1991.

[19] J. Hersberger and J. Snoeyink. Cartographic line simplification and polygon csg formulae in $O(n \log^* n)$ time. *Proc. 5th International Workshop on Algorithms and Data Structures*, pages 93–103, 1997.

[20] F. Hurtado, P. Ramos M. Noy, and C. Seara. Separating objects in the plane with wedges and strips. *Discrete Applied Mathematics*, 109:109–138, 2001.

[21] H. Imai and M. Iri. Computational-geometric methods for polygonal approximations of a curve. *Computer Vision, Graphics and Image Processing*, 36:31–41, 1986.

[22] H. Imai and M. Iri. An optimal algorithm for approximating a piecewise linear function. *Journal of Information Processing*, 9(3):159–162, 1986.

[23] H. Imai and M. Iri. Polygonal approximations of a curve-formulations and algorithms. *Computational Morphology*, pages 71–86, North–Holland, Amsterdam, 1988.

[24] Y. Kurozumi and W.A. Davis. Polygonal approximations by the minimax method. *Computer Graphics and Image Processing*, 19:248–264, 1982.

[25] M.K. Leung and Y.-H. Yang. Dynamic two-strip algorithm in curve fitting. *Pattern Recognition*, 23(1/2):69–79, 1990.

[26] A. Melkman and J. O'Rourke. On polygonal chain approximation. *Computational Morphology*, pages 87–95, North–Holland, Amsterdam, 1988.

[27] B.K. Natarajan. On comparing and compressing piecewise linear curves. Technical Report, Hewlett Packard, 1991.

[28] B.K. Natarajan and J. Ruppert. On sparse approximations of curves and functions. *Proc. 4th Canadian Conference on Computational Geometry*, pages 250–256, 1992.

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

[30] G.T. Toussaint. On the complexity of approximating polygonal curves in the plane. *Proc. IASTED International Symp. on Robotics and Automation*, Lugano, Switzerland, 1985.