A Practical Algorithm for the Greedy Triangulation of a Convex Polygon

Graham D. Finlayson and Binay K. Bhattacharya
School of Computing Science
Simon Fraser University
Vancouver. V5A 1S6
email: graham@cs.sfu.ca

Abstract

A *triangulation* of a convex polygon can be considered to be a set of non-intersecting chords (chords join two vertices) such that the polygon is partitioned into triangles. The *greedy triangulation* (GT) set is built by repeatedly adding the minimum length internal chord such that this chord does not intersect with any already in the set.

Recently Levecopolous and Lingas[6] presented an asymptotically optimal, $O(k^3n)$ ($k \geq 42$), algorithm for the greedy triangulation problem. In this paper we develop a new practical k-d tree based algorithm for the greedy triangulation problem with average case complexity of $O(n \log n)$. Every time the proposed algorithm adds a minimum length chord, it makes at most 6 searches of the k-d tree.

1 Introduction

A triangulation of a convex polygon can be considered as a set of non-intersecting chords such that the interior of the polygon is partitioned into triangles. A chord is a line segment joining non-adjacent vertices in the polygon boundary. It is well known that there are $\frac{2}{n-1}\binom{2n-5}{n-3}$ possible triangulations of a convex polygon with n vertices [2]. Let each chord in a triangulation be assigned a weight equal to it's length. Further let the sum of the weights of these chords define the weight of a triangulation. The minimum weight triangulation (MWT) of an n-vertex simple polygon can be determined in $O(n^3)$ worst case time[3]. However an algorithm which is guaranteed to generate a triangulation whose weight is close to optimal (the MWT) but which has lower computational costs is desirable. Levecopolous and Lingas[5] have shown that the Minimum Weight Triangulation of a convex polygon is approximated to within a constant factor by it's *greedy triangulation* whose procedure is shown in Figure 1.

In[6] Levecopolous and Lingas have shown that the greedy triangulation set can be built *locally*; and as such there is no necessity to maintain global minimum chord information. This observation led to their optimal linear time, $O(k^3n)$, algorithm for greedy triangulation. However $k \geq 42$ and this implies that algorithms whose asymptotic complexity is higher ($O(n \log n)$ or even $O(n^2)$) can perform better for convex polygons with large number of vertices. We expand upon this idea in designing a practical algorithm for the greedy triangulation problem.

Consider the algorithm GT for calculating the greedy triangulation of a convex polygon P. To find the minimum chord nearest neighbour information must be maintained at each of the vertices in the polygon. Each time the polygon is split we must update this nearest neighbour information. Specifically the nearest neighbour of a vertex must not lie on the opposite side of the minimum chord. Thus the complexity of algorithm GT is bounded by the cost of finding the minimum chord, updating neighbourhood information and splitting the

```
Procedure GT($P$);
{ $n$ =number of vertices in $P$;
  if ($n \leq 3$) then return $\emptyset$;
          else { $c$ =minimum chord in $P$;
          split $P$ along $c$ into $P_1$ and $P_2$;
          update nearest neighbours in $P_1$;
          update nearest neighbours in $P_2$;
          return $\{c\} \bigcup$GT($P_1$)$\bigcup$GT($P_2$);
          }
}
```

Figure 1: Greedy Triangulation Algorithm: GT

polygon. In[5] Levecopolous and Lingas have shown GT can be implemented with worst case complexity of $O(n^2)$.

In their algorithm as each minimum chord is added to the GT set the neighbourhood information at all remaining vertices is recalculated. In this paper we exploit geometric properties of convex polygons to reduce the cost of updating neighbourhood information in GT. This leads to two efficient implementations of GT. The first with complexity $3n^2 + o(n^2)$ will outperform the optimal algorithm for convex polygons for moderate n. The second with average case complexity $O(n \log n)$ performs better than optimal in almost all practical cases.

2 $O(n^2)$ Implementation of GT

A convex polygon P has n vertices $v_0, v_1, \cdots, v_{n-1}$ ordered in a clockwise direction. Each vertex has two adjacent neighbours—v_{i-1} and v_{i+1} are adjacent to v_i[1]. The nearest *non-adjacent* neighbour to a vertex v_i in P is denoted by $\mathcal{N}(v_i)$. The *chord* $[v_i, v_j]$ is a directed line segment joining v_i to v_j and has length $\|[v_i, v_j]\|$. If the *minimum chord* of a convex polygon is $[v_m, \mathcal{N}(v_m)]$ (the one chosen during greedy triangulation) then $\|[v_m, \mathcal{N}(v_m)]\| \leq \|[v_i, \mathcal{N}(v_i)]\|, i = 0, 1, \cdots, n-1$. Finally we denote the halfspace which includes all points closer to v_i than v_j as $\mathcal{H}(v_i, v_j)$, the halfspace to the right of $[v_i, v_j]$ as $\mathcal{H}([v_i, v_j])$ and the halfspace bounded by the perpendicular through v_i of $[v_i, v_j]$ which contains v_j as $\mathcal{H}([v_i, v_j]\perp)$.

We distinguish between two types of minimum chords. An *ear* is a chord which joins two vertices v_i and v_{i+2}. If a polygon is split on an ear then this is equivalent to deleting the vertex v_{i+1}. A *guard* describes all other minimum chords. Splitting a polygon on a guard will result in two non-trivial subpolygons—both will have to be greedy triangulated separately.

Clearly any algorithm for generating the greedy triangulation of a convex polygon must have some mechanism for choosing it's minimum chord. This implies that the $\mathcal{N}(v_i), i = 0, 1, \cdots, n-1$ needs to be calculated and maintained as each minimum chord is added to the triangulation. In the discussion which follows we will refer to vertices whose nearest neighbour information changes, on choosing a minimum chord, as *update* vertices.

Theorem 1 *When a minimum chord is added to the greedy triangulation of P the nearest non-adjacent neighbours of at most 6 vertices need to be recalculated.*

The general approach to proving theorem 1 involves trying to construct a polygon which when split along the minimum chord results in maximum update vertices.

Sketch of proof for ears: By definition an ear splits a polygon of n vertices into a triangle and a polygon with $n-1$ vertices. Consider the ear to be $[v_i, v_{i+2}]$ with $\|[v_i, v_{i+2}]\| = d$. Clearly v_{i+1} must lie to the left of $[v_i, v_{i+2}]$ and all other points to its right.

Lemma 1 *If $[v_i, v_i + 2]$ is the minimum chord and $\mathcal{N}(v_k) = v_{i+1}$, where $k \neq i-1$ and $k \neq i+3$, then $v_{i+1}, v_k \in \mathcal{H}([v_i, v_{i+2}]\perp) \bigcap \mathcal{H}([v_{i+2}, v_i]\perp)$.*

[1]all indices are modulo $n+1$

Clearly the vertex v_k must lie in $\mathcal{H}(v_{i+1}, v_i) \bigcap \mathcal{H}(v_{i+1}, v_{i+2})$. Given the constraints of Lemma 1 we maximize $\mathcal{H}(v_{i+1}, v_i) \bigcap \mathcal{H}(v_{i+1}, v_{i+2})$ (and our freedom in placing v_k) if v_{i+1} lies on $[v_i, v_{i+2}]$. Note in this case $\mathcal{H}(v_{i+1}, v_i) \bigcap \mathcal{H}(v_{i+1}, v_{i+2})$ lies between two parallel lines, perpendicular to $[v_i, v_{i+2}]$, $0.5 * \|[v_i, v_{i+2}]\|$ apart. It is easy to place 2 adjacent vertices, v_{k1} and v_{k2} in $\mathcal{H}(v_{i+1}, v_i) \bigcap \mathcal{H}(v_{i+1}, v_{i+2})$ such that $\mathcal{N}(v_{k1}) = \mathcal{N}(v_{k2}) = v_{i+1}$. However it is impossible to place 3 vertices such that all have v_{i+1} as their nearest neighbour. This follows directly from Lemma 2.

Lemma 2 *Consider a triangle $\mathcal{T}$ with verices v_a, v_b and v_c where $\|[v_a, v_b]\| < \|[v_b, v_c]\| < \|[v_c, v_a]\|$. If v_b has the least y-coordinate then it is impossible for $\mathcal{T}$ to lie between two parallel lines perpendecular to the x-axis which are $\frac{1}{2}\|[v_a, v_b]\|$ apart.*

This complete the sketch of proof for ears. There are at most 6 update vertices when an ear is chosen in the greedy triangulation: v_i and v_{i+2}, the 2 vertices in $\mathcal{H}(v_{i+1}, v_i) \bigcap \mathcal{H}(v_{i+1}, v_{i+2})$ and the 2 vertices adjacent to v_i and v_{i+2}.

Sketch of proof for guards: Let us define the lune of influence of a chord as lune($[v_i, v_j]$) = disk$(v_i, d) \bigcap$ disk(v_j, d); where $d = \|[v_i, v_j]\|$ and disk(v_i, r) is the space bounded by a circle centred at v_i with radius r.

Lemma 3 *If $[v_i, v_j]$ is a guard then lune($[v_i, v_j]$) must be empty.*

Proof: This follows immediately from the definitions of lune of influence and minimum chord.

Let us place a vertex v_k to the right of the guard $[v_i, v_j]$, where $v_k \notin$ lune($[v_i, v_j]$). Without loss of generality assume that $v_k \in \mathcal{H}(v_i, v_j)$. A vertex v_l to the left of $[v_i, v_j]$, where $i + 1 < l < j - 1$ (v_l is non-adjacent to v_i and v_j), can have $\mathcal{N}(v_l) = v_k$ if and only if $v_l \in \mathcal{H}([v_j, v_i]) \bigcap \mathcal{H}(v_k, v_i) \bigcap \mathcal{H}(v_j, v_k)$.

Lemma 4 *The guarding property states that: if $v_k \notin$ lune(v_i, v_j) then $\mathcal{H}([v_j, v_i]) \bigcap \mathcal{H}(v_k, v_i) \bigcap \mathcal{H}(v_j, v_k) = \emptyset$; that is, vertices non-adjacent to a guard cannot have their nearest neighbours on the opposite side of the guard.*

A consequence of Lemma 4 is that $\mathcal{N}(v_l)$ can be on the opposite side of $[v_i, v_j]$ if and only if v_l is adjacent to v_i or v_j. This implies that there can be at most 6 update vertices for guards—v_i, v_j and their adjacent vertices. This completes the proof of theorem 1.

Theorem 1 places a constant bound on the number of update vertices created each time a polygon is split. These update vertices can be found in $O(1)$. For guards, the update vertices are identified by examining adjacent vertices. However ears can create non-adjacent update vertices but these are taken care of by maintaining the update list $\mathcal{N}^{-1}(v_i)$ at each vertex.

A priori to calling algorithm GT we can initialize $\mathcal{N}(v_i)$ and $\mathcal{N}^{-1}(v_i)$ ($i = 0, 1, \cdots, n-1$) in $O(n^2)$ by a simple enumeration of the distances between pairs of vertices. The initialization cost can be reduced to $O(n)$ using Levecopolous and Lingas'[5] extension of Lee and Preparata's[4] all nearest neighbour algorithm[2]. The complexity of GT is bounded by the cost of the basic operations update_polygon, minimum_chord and split_polygon. These costs are dependent on the underlying data structures. Currently the only data structure is the linked list of vertices; hence minimum_chord requires n link traversals in the worst case and split_polygon takes $O(1)$ time. The nearest neighbour of each of the 6 update vertices takes n operations and hence updating the polygon can cost $6n + o(1)$. A greedy triangulation consists of $n-3$ chords—this gives a worst case complexity for GT of $3n^2 + o(n)$. Thus by characterizing the update vertices we have an algorithm which will outperform optimal linear algorithm for convex polygons of reasonable size.

3 An $O(n \log n)$ implementation of GT

Organizing the vertices of a convex polygon in a single linear data structure—a linked list—causes the complexity of the minimum_chord and update_polygon to be $O(n)$. In this section we introduce data structures which lead to an $O(n \log n)$ average case implementation of GT.

[2]The nearest neighbour of a vertex in Lee and Preparata's algorithm can be adjacent

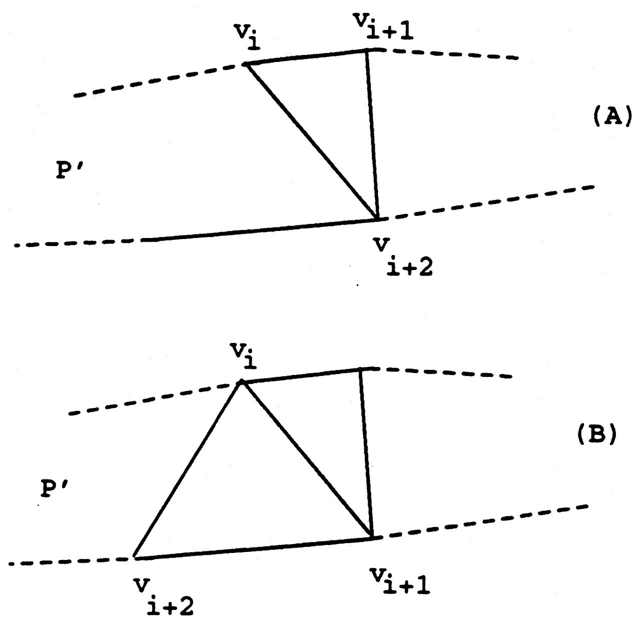

Figure 2: Guard-extensions

Let us store minimum chord information in a 2-3 tree where the leaves of the tree correspond to vertices in the polygon. The leaf node representing v_i stores the length of the chord $|[v_i, \mathcal{N}(v_i)|$. We use the 2-3 tree as a priority queue. Thus the minimum_chord operation costs $O(1)$. During a greedy triangulation the polygon must be split about the minimum chord. If the leaves of the tree are ordered by increasing vertex number then the corresponding splitting of the 2-3 tree costs $O(\log n)$ in the worst case. Thus the total cost of 2-3 tree operations for GT is $O(n \log n)$.

Similarly we would like to reduce the update complexity to $O(\log n)$ expected time. A k-d tree[1] is a distance encoding data structure which allows nearest neighbour querying in $O(\log n)$ average complexity. Further insertion and deletion also cost $O(\log n)$ expected time. The cost of splitting a k-d tree is bounded by the number of vertices in the smaller of the resulting subtrees. Let k be the size of the smaller subpolygon generated when a minimum chord is added to the GT set. The k-d tree is split by deleting these k vertices and reinserting them into a new (second) k-d tree which requires $O(k \log k + k \log n)$ operations. Thus the cost of splitting k-d trees, during the entire process of greedy triangulation, is $O(n \log^2 n)$ since k is atmost half of the size of the polygon being split. To achieve an $O(n \log n)$ expected bound we show that the k-d tree does not, in fact, have to be split.

Given a snapshot of the execution of GT, we define a ***guard-extension*** to be either:

- an ear $[v_i, v_{i+2}]$ where either $[v_i, v_{i+1}]$ or $[v_{i+1}, v_{i+2}]$ is a guard already in the GT set, example (A) in Figure 2.
- an ear $[v_i, v_{i+2}]$ where either $[v_i, v_{i+1}]$ or $[v_{i+1}, v_{i+2}]$ is a guard-extension already in the GT set, example (B) in Figure 2.

By definition guard-extensions are associated with a unique ***source*** guard.

Theorem 2 *A polygon P' generated from P during the execution of $GT(P)$ has at most 2 chords which are guards or guard-extensions.*

Proof sketch: Let us walk, clockwise, around the boundary of P'. At each vertex v_i we make a right turn of θ_i degrees. The total turn of the tour is $\sum_{i=0}^{n-1} \theta_i = 360$ degrees.

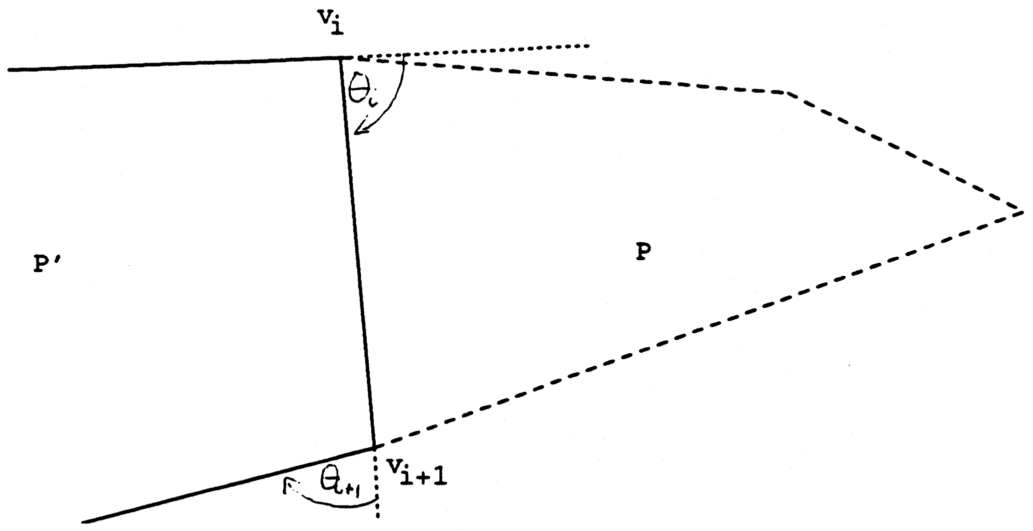

Figure 3: The turn of $[v_i, v_{i+1}]$ is equal to $\theta_i + \theta_{i+1}$.

Each guard edge $[v_i, v_{i+1}]$ of P' results in turns of $\theta_i + \theta_{i+1}$. The total turn from all guards must be less than or equal to 360 degrees. Let us calculate the turn of each guard of P' with respect to the convex chain to its left in P, see Figure 3.

By Lemma 3, the lune of influence of each guard must be empty. Hence it is straightforward to show that, assuming that all vertices are at unique locations, the total turn from each guard in P (and hence P') is strictly greater than 120 degrees. Theorem 2 follows immediately.

Guard-extensions exhibit the *guarding property* with respect to their sources. That is, vertices non-adjacent to a guard-extension cannot have their nearest neighbours on the opposite side of the corresponding source guard.

Theorem 3 *Let $[v_i, v_j]$ be a source guard and $[v_x, v_y]$ be an associated guard extension (the source of $[v_x, v_y]$ is $[v_i, v_j]$, where $i \le x < y \le j$). If the vertex v_k occurs to the left of $[v_x, v_y]$ where $x+1 < k < y-1$ (v_k is non-adjacent to $[v_x, v_y]$), $\mathcal{N}(v_k) \neq v_l, l = j+1, j+2, \cdots, i-2, i-1$.*

Proof Sketch: Since $[v_x, v_y]$ is a minimum length chord $v_k \notin \text{disk}(v_x, d) \bigcup \text{disk}(v_y, d)$ $(d = \|[v_x, v_y]\|)$. If $\mathcal{N}(v_k) = v_l$, where v_l lies to the right of $[v_x, v_y]$ then $\|[v_x, v_k]\| \ge \|[v_l, v_k]\|$ and $\|[v_y, v_k]\| \ge \|[v_l, v_k]\|$. In this case it is straight forward to show that $v_l \in \Delta v_x v_z v_y$ where $\angle v_x v_z v_y = 120$ and $\angle v_z v_x v_y, \angle v_z v_y v_x = 30$, see Figure 4.

Lemma 5 *Let us place points v_c and v_d on the line segment $[v_a, v_b]$. and a third point v_e to the right of $[v_a, v_b]$ such that $\angle v_c v_e v_d > 90$ then $\Delta v_d v_e v_c \in lune([v_a, v_b])$.*

The $\Delta v_x v_z v_y$ may contain no source guard vertices, one source guard vertex, or both source guard vertices. In each of these cases the space to the right of $[v_i, v_j]$ which can be closer to v_k than v_x or v_y lies in $\Delta v_d v_z v_c$ where v_d and v_c lie on $[v_i, v_j]$ and $\angle v_c v_z v_d > 90$. By Lemma 5 this space is contained within lune($[v_i, v_j]$) which by Lemma 3 is empty. This proves Theorem 3.

Theorem 4 *Let the vertices of P be split into (any) two disjoint sets S_1 and S_2. For each $v_i \in S_1$ the $\mathcal{N}(v_i) \in S_1 \bigcup S_2$ is known. For each $v_j \in S_2$ only the nearest neighbour of v_j in S_2 is known. The minimum chord in P can be found by examining $\mathcal{N}(v_i)$ for all vertices in P.*

Theorems 2, 3 and 4 are at the heart of the $O(n \log n)$ implementation of GT. Let P' denote a subpolygon generated during the execution of GT(P). We divide the vertices into 2 sets, S_1 contains vertices which are not part of, and are not adjacent to guards or guard-extensions; S_2 contains all other vertices. By theorem 2 the cardinality of S_2 is at most 8. For any update vertex $v_u \in S_2$ we find $\mathcal{N}(v_u) \in S_2$, by brute force evaluation of

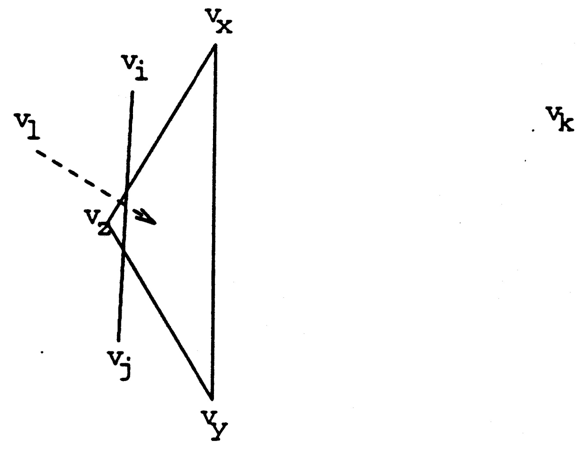

Figure 4: If $\mathcal{N}(v_k) = v_l$ then $v_l \in \Delta v_x v_y v_z$.

the distance between v_u an all other vertices in S_2—this costs $O(1)$. By Lemma 4 and Theorem 3 for $v_u \in S_1$, $\mathcal{N}(v_u) \in S_1 \bigcup S_2$ can be calculated by k-d tree querying in $O(\log n)$ time; since nearest neighbours of vertices in S_1 cannot lie on on the opposite sides of source guards. However the nearest neighbour of a vertex can lie on the opposite side of a guard extension. In particular a source guard vertex could be nearest. However deletion of source guard vertices from the k-d tree prior to querying will ensure correct nearest neighbour updates. Finally by theorem 4 we still maintain the minimum chord information.

Theorem 5 *The algorithm GT can be implemented in $O(n \log n)$ average case complexity with at most 6 queries of the k-d tree for every minimum chord choice.*

4 Conclusion

In this paper we have exploited geometric properties of convex polygons to design an $O(n \log n)$ average case algorithm for greedy triangulation. For all practical purposes this algorithm, on average, outperforms all other greedy triangulation methods (including the linear time optimal) for convex polygons.

References

[1] J.H. Friedman and J.L. Bentley. An algorithm for finding best matches in logarithmic expected time. In *TOMS*, volume 3, 1977.

[2] R. Honsberger. *Mathematical Gems 1.* The Mathematical Society of America, 1973.

[3] G.T. Klinscek. Minimal triangulations of polygonal domains. *Ann. Disc. Math*, 9:121–123, 1980.

[4] D.T. Lee and F.P. Preparata. The all-nearest-neighbour problem for convex polygons. *Inform. Process. Lett.*, 7:189–192, 1978.

[5] C. Levecopolous and A. Lingas. On approximation behaviour of greedy triangulation for convex polygons. *Algorithmica*, 2:175–193, 1987.

[6] C. Levecopolous and A. Lingas. Fast algorithms for greedy triangulation. In *Scandinavian Workshop on Algorithm Theory*, pages 238–249, June 1990.