Quadtree Decomposition, Steiner Triangulation, and Ray shooting*

Siu-Wing Cheng Kam-Hing Lee

Department of Computer Science, HKUST, Clear Water Bay, Hong Kong.

Abstract. We present a new quadtree-based decomposition of a polygon possibly with holes. For a polygon of n vertices, a truncated decomposition can be computed in $O(n \log n)$ time which yields a Steiner triangulation of the interior of the polygon that has $O(n \log n)$ size and approximates the minimum weight Steiner triangulation (MWST) to within a constant factor. An approximate MWST is good for ray shooting in the average case as defined by Aronov and Fortune. The untruncated decomposition also yields an approximate MWST. Moreover, we show that this triangulation supports query-sensitive ray shooting as defined by Mitchell, Mount, and Suri. Hence, there exists a Steiner triangulation that is simultaneously good for ray shooting in the query-sensitive sense and in the average case.

1 Introduction

Triangulation is a popular research topic because many problems call for a decomposition of a scene into simple elements that facilitate processing. In the plane, the focus has been on optimizing or approximating some measure. Steiner points are allowed and the triangulation obtained must respect the given line segments (i.e., a given line segment must be equal to the union of some edges in the triangulation). One natural measure is the total length of edges in the triangulation. This is called the *weight* of the triangulation. No algorithm is known that computes the *minimum weight Steiner triangulation* (MWST for short) of a point set or a polygon. Eppstein [5] presented an $O(n \log n)$-time algorithm to approximate the MWST of a point set to within a constant factor. A variant of the method also works for a convex polygon. One question raised in [5] is how to compute an approximate MWST for a general polygon.

The MWST problem, which has been of theoretical interest so far, is recently related to the *ray shooting* problem by Aronov and Fortune [1]. The ray shooting problem is to report the first obstacle hit by a query ray. In this paper, we assume that the scene is a polygon possibly with holes. Query light source will fall into this polygon, and the boundary of the polygon acts as obstacles. The usual scene of a collection of simple polygons can be modeled by enclosing them in a square and treating the given polygons as holes.

A simple approach for the ray shooting problem is to decompose the polygon into a *planar subdivision* so that each face is of constant complexity and has

* Research supported in part by RGC CERG HKUST 650/95E.

at most a constant number of adjacent faces. Then the ray shooting query is answered by walking from face to face until the first obstacle is hit. Aronov and Fortune [1] showed that a Steiner triangulation of small weight is good for ray shooting in the average case. The average is taken over all random choices of light source on polygon boundary and over all random choices of ray direction. In 2D, the average number of triangulation edges visited is equal to the weight of Steiner triangulation divided by the boundary length of the polygon. Aronov and Fortune also claimed a polynomial-time algorithm for approximating the MWSTs of a set of line segments enclosed within their convex hull in 2D.

Mitchell, Mount, and Suri [7] proposed a notion of *C-complexity* for measuring the complexities of the scene and a ray shooting query. In the plane, a *C-ball* is a connected component of the intersection the scene with a disk. The center and radius of the C-ball are taken to be the center and radius of the defining disk. Given a C-ball B, if we expand its defining disk by a factor of $1+\epsilon$, then there is one connected component in this expanded disk that contains B. We denote this component by $(1+\epsilon)B$. A C-ball is *simple* if it does not intersect more than two edges. A C-ball B is ϵ-strongly simple if both B and $(1+\epsilon)B$ are simple. For a fixed ϵ, the C-complexity of a polygon is the minimum number of ϵ-strongly simple C-balls that cover the interior of the polygon. Mitchell *et al* presented an algorithm to decompose the polygon interior into convex cells so that given a query ray, the number of cells visited is proportional to the number of ϵ-strongly simple C-balls that cover the ray up to the first intersection. This is called *query-sensitive ray shooting* in [7].

The main contribution of our paper is a new quadtree-based decomposition of a polygon possibly with holes which is inspired by the quadtree decomposition developed for multiple-tool milling [2]. We prove that triangulating the decomposition yields an approximate MWST of the polygon. For a polygon of n vertices, a truncated decomposition has $O(n \log n)$ size and can be computed in $O(n \log n)$ time. Triangulating the truncated decomposition also produces an approximate MWST of size $O(n \log n)$. The boundary length of the polygon is included in the weight of the triangulation. No result is known that excludes the boundary length. Our result is different from that in [1] since they require a convex outer boundary and they compatibly triangulate the holes too. In general, compatible triangulation of the holes produces more triangles (possibly $\Omega(\log n)$ more, see Section 5).[1] We also show that the triangulation of the untruncated decomposition supports query-sensitive ray shooting. Since this triangulation also approximates the MWST, this demonstrates the existence of a Steiner triangulation that is simultaneously good for ray shooting in the query-sensitive sense and in the average case.

The rest of the paper is organized as follows. Section 2 provides the basic definitions. Section 3 describes our quadtree-based decomposition. In Section 4, we discuss how to use the decomposition to obtain approximate MWSTs. Section 5 discusses the application in ray shooting.

[1] In other words, we assume that the exterior of the polygon is inaccessible by light. This is not assumed in [1] as each line segment is treated as an individual obstacle.

2 Preliminaries

Let P denote a polygon possibly with holes. Our quadtree-based decomposition can accommodate degenerate holes. However, if a Steiner triangulation is to be obtained, then we allow a hole degenerate to a single point but not line segments.

A *quadtree* is a ternary tree representing a hierarchical decomposition of the plane, originally proposed for representing point sets. Each node of the quadtree corresponds to a square region, called a *box*. The root usually corresponds to the smallest enclosing square of the given set of objects. In our case, it is the smallest enclosing square of P. A node of the quadtree acquires four children when its associated box is split into its four quadrants. Given a box b, we denote its width by $size(b)$ and its parent by $p(b)$.

In the quadtree, boxes of the same size are of the same height and they form a regular grid partitioning the entire plane. A *neighbor* of a box b is a box of the same size that touches b. An *orthogonal neighbor* of a box b is a box of the same size that shares a side with b.

For any box b and a constant c, we denote by $c \cdot b$ a square with the same center as b and side length $c \cdot size(b)$. Given any square S (not necessarily a box in a quadtree), we call a connected component of $P \cap S$ a *subpolygon* of S. We define the size of a subpolygon α of a box b to be $size(b)$ and denote it by $size(\alpha)$.

3 Quadtree-based decomposition

For each subpolygon α of b, define $zone(\alpha)$ to be the subpolygon of $3b$ that contains α. The decomposition takes a parameter $\delta > 0$ which is fixed beforehand. A subpolygon α of b is *crowded* if $size(\alpha) > \delta$ and $zone(\alpha)$ contains more than one polygon vertex. If α is uncrowded and the subpolygon of $p(b)$ containing α is crowded, then α is a *cell* in the final decomposition. We say the the cell α is *created* at box b and we call b the *home box* of α.

A box b is crowded if one of its subpolygons is crowded. A crowded box is split into its four quadrants, thus generating four children in the quadtree. *Only* the crowded subpolygons of b will be split together with b. The components obtained will be distributed into the four children of b. Although a cell created at b will not be split further, new vertices may be inserted into its boundary when adjacent crowded subpolygons are split. When no box in the tree is to be split further, the collection of cells created form a planar subdivision of the interior of P. We denote this by $\mathcal{D}$. Figure 1 illustrates one step of this hierarchical decomposition.

If we set the parameter δ to be sufficiently small (e.g. less than the smallest Euclidean distance between two polygon vertices), then each cell has at most one polygon vertex. In this case, we call $\mathcal{D}$ *untruncated*. Otherwise, $\mathcal{D}$ is *truncated*. In $\mathcal{D}$, a cell is *normal* if its size is larger than δ, otherwise it is *small*. Normal cells have constant complexity and small cells have the same size. There are two kinds of edges bounding a cell. Those that lie on some polygon edges are called *solid edges* and the others are called *non-solid edges*.

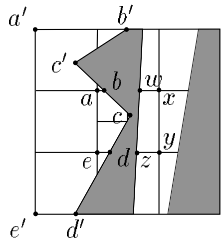

Fig. 1. The shaded regions belong to holes. The subpolygon $wxyz$ is not crowded and so it becomes a cell. The subpolygon $abcde$ is crowded as its zone $a'b'c'cd'e'$ contains two polygon vertices. So $abcde$ is split further when splitting the box. Note that $wxyz$ is not split.

Our hierarchical decomposition proceeds in one pass in a top-down fashion. There are two key steps, namely crowdiness testing and subpolygon splitting. To facilitate the subsequent splitting of subpolygons, we compute a noncrossing spanning tree of the holes and the outer boundary of P. This is done as follows. First, we convert P to a monotone subdivision by adding diagonals in $O(n \log n)$ time [8]. Then we invoke a linear time graph search of the subdivision to identify a spanning tree of the holes and the outer boundary of P. Splitting along the spanning tree edges yields a degenerate simple polygon P^* of size $O(n)$. P^* will be useful later.

We split all crowded subpolygons of boxes of the same size before continuing to the next level. Inductively, a list of subpolygons of a box b will be inherited from the splitting of crowded subpolygons of $p(b)$. Take a subpolygon α in the list. If α contains more than one polygon vertex, then α is crowded. Suppose not. Then we collect subpolygons that touch α. There are two kinds. The first kind has size larger than $size(\alpha)$. These are normal cells of constant complexity. There are at most three such cells touching α, and it can be checked in constant time whether they contribute more than one polygon vertex in $zone(\alpha)$. The second kind are subpolygons of neighbors of b that touch α. These subpolygons are subsets of $zone(\alpha)$. For each such subpolygon, its number of polygon vertices can be precomputed. So it takes constant time to check each such subpolygon. Moreover, since α is assumed to contain at most one polygon vertex (α is already crowded otherwise), there are at most nine subpolygons of neighbors of b that touch α. In all, testing crowdiness of α takes constant time. If α is uncrowded, then it is a cell and we remove it from the list. Otherwise, α will be split.

Splitting α is equivalent to clipping α within each of the four quadrants of b. We discuss how to do this for one quadrant. If α is a simple polygon, then this can be done in linear time using Jordan sorting [6]. However, α may contain holes and in this case, P^* comes to our rescue. We keep an *unnested version* for α defined as follows. The spanning tree edges in P^* separate α into several components. Take any two such components that are separated by a spanning tree edge that cuts completely across b. Remove this separating tree edge and merge the two components into one. Continue until there is no such adjacent pair of components. (See Figure 2.) The resulting set of simple polygons is the unnested version of α and we denote it by α^*. Note that α and α^* covers the same region in the plane.

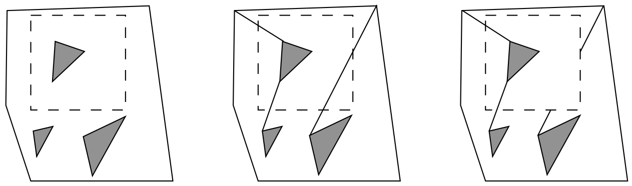

Fig. 2. In the left figure, the shaded regions are holes in the polygon. The dashed square is a box which has only one subpolygon α. After adding the spanning tree edges in P^*, we obtain the middle figure. α is separated into three components and two are separated by a spanning tree edge that cuts completely across the box. In the right figure, after merging, we obtain two simple polygons which form α^*.

We clip each simple polygon in α^* instead of α. This can be done using Jordan sorting [6]. Let m be the size of α. The total complexity of α^* is $O(m)$ as any spanning tree edge bounding α^* is incident to a vertex of α. This implies that the clipping takes $O(m)$ time. What remains is how to pierce together the components resulting from clipping α^* to produce the desired subpolygons and their unnested versions. As before, we repeatedly merge any two components that are separated by a spanning tree edge that cuts completely across the quadrant. This takes time linear in the number of such spanning tree edges which is $O(m)$ as they all bound α^*. Let X be the set of simple polygons produced. Identify all the maximal connected subsets of polygons in X in linear time. The union of each maximal connected subset is a subpolygon of the quadrant whose unnested version is the maximal connected subset. In all, clipping α can be done in linear time.

$\mathcal{D}$ satisfies a structural property: no cell can be bounded by a collinear chain of three non-solid edges. This implies that every normal cell and every small cell containing at most one polygon vertex is adjacent to a constant number of other cells. This is similar in spirit to the "balance" or "smoothness" properties of other quadtree decompositions studied [3, 5, 7].

Lemma 1. *No cell can be bounded by a collinear chain of three non-solid edges.*

Proof: Assume to the contrary there is a collinear chain s of three non-solid edges bounding a cell γ. Let b_1 be the home box of γ. Since s contains quadtree vertices, b_1 does not have the smallest size in the hierarchy and so γ is a normal cell. Let b_2 be the orthogonal neighbor of b_1 that shares s. There must be some proper descendant b_3 of b_2 such that b_3 puts a corner vertex on s, and a proper descendant of b_3 puts another corner vertex v on s. Let α be the subpolygon of b_3 that contains the point v on its boundary. Since α was split, $zone(\alpha)$ contains more than one polygon vertex. Since α is adjacent to γ and $zone(\alpha) \subseteq 3b_3 \subseteq 3b_1$, we conclude that $zone(\gamma)$ also contains more than one polygon vertex, contradiction. ◻

4 Analysis

4.1 Total length of the decomposition

We conceptually refine $\mathcal{D}$ to another decomposition $\mathcal{D}^*$ to ease the analysis. This is done by relaxing the definition of crowdiness. A subpolygon α is crowded if $size(\alpha) > \delta^*$ and $zone(\alpha)$ contains *at least one* polygon vertex (instead of more than one), where δ^* is a constant chosen to be less than δ and close to zero. At the end, we also call a cell γ of $\mathcal{D}^*$ normal if $size(\gamma) > \delta^*$ and small otherwise. Since $\mathcal{D}^*$ is a refinement of $\mathcal{D}$, its total length is larger and so it suffices to bound the total length of $\mathcal{D}^*$. The advantage of $\mathcal{D}^*$ is that any cell that has a polygon vertex must be small.[2] We first state a technical lemma. Its proof is omitted here.

Lemma 2. *Let γ_1 be a normal non-square cell of $\mathcal{D}^*$. Suppose that each solid edge of γ_1 has length less than $size(\gamma_1)/4$. Then γ_1 is adjacent to a cell γ_2 of $\mathcal{D}^*$ of perimeter $\Theta(size(\gamma_1))$ and the total length of solid edges bounding γ_2 is at least $size(\gamma_1)/4$.*

We can now bound the total length of $\mathcal{D}^*$ and hence the total length of $\mathcal{D}$. Our proof follows the approach in [5].

Lemma 3. *For a polygon P of n vertices, the total length of $\mathcal{D}^*$ and hence $\mathcal{D}$ is bounded by $O(w(MWT))$, where $w(MWT)$ is the weight of the minimum weight triangulation of P.*

Proof: Let MWT denote the minimum weight triangulation of P. Recall that no Steiner point is allowed in MWT. Small cells have negligible perimeter as δ^* is close to zero. So we focus on normal cells.

Let γ be a normal square cell. Let Δ be the triangle in MWT that covers the center of γ. If Δ has a vertex u inside 11γ, then we charge the perimeter of γ proportionally to the two edges of Δ incident to u. Since the vertices of Δ are outside γ, at least one edge of Δ has length $size(\gamma)$. Thus, the total length of the two edges incident to u is large enough to absorb the charge. Suppose that Δ does not have a vertex inside 11γ. Let α be the subpolygon of $p(\gamma)$ that contains γ. Since α was split, $zone(\alpha)$ contains a polygon vertex which implies that $zone(\alpha)$ has a polygon vertex v that can see the center, c, of γ. The line segment cv stabs a sequence of triangles in MWT. We visit this sequence in order starting from Δ. Eventually, we must reach a triangle Δ' with a vertex inside 11γ as cv has length less than $5.25 size(\gamma)$. Let xy be the edge of Δ' through which we step into Δ'. So x and y are outside 11γ. Since x and y are at distance

[2] The disadvantage is that the excessive splitting makes the size of $\mathcal{D}^*$ not bounded by the C-complexity of P.

at least $5.5size(\gamma)$ from c, xy has length at least $\sqrt{5.5^2 - 5.25^2}size(\gamma) > size(\gamma)$. Thus, we can charge the perimeter of γ proportionally to the two edges of Δ' incident to the vertex opposite to xy.

In the above charging scheme, if an edge e of MWT acquires a charge O(length of e) from a square cell γ, then γ lies inside a box of width $12size(\gamma)$ centered at an endpoint of e. At one level of the quadtree, there are at most $2 \times 12^2 = 288$ square cells that satisfy this requirement. Since box size halves from level to level, the accumulated charge at e telescopes to O(length of e). This shows that the sum of perimeter of normal square cells is $O(w(MWT))$.

Consider a non-square cell γ_1. If the solid edges bounding γ_1 have a total length at least $size(\gamma_1)/4$, then we charge the perimeter of γ_1, which is $O(size(\gamma_1))$, to these solid edges proportionally. When this is not the case, we apply Lemma 2 and charge the perimeter of γ_1 to the perimeter of an adjacent cell.

In all, the total length of $\mathcal{D}^*$ is asymptotically bounded by the sum of $w(MWT)$ and the boundary length of P. Since $w(MWT)$ includes the boundary length of P, the result follows. ◻

4.2 Approximate MWST

To obtain an approximate MWST, we triangulate the cells in $\mathcal{D}$. By Lemma 1, triangulating a cell with at most one polygon vertex adds only a constant number of diagonals. Thus, the total length will be increased by only a constant factor. Triangulating a cell with $m > 1$ polygon vertices may add $O(m)$ diagonals. However, each such diagonal has length at most δ. Summing over all cells with more than one polygon vertex, the additional length is $O(\delta n)$. Hence, triangulating $\mathcal{D}$ increases the weight by an additive term $O(\delta n)$.

There is a subtle problem when holes are allowed to degenerate to line segments. When this happens, the triangles on opposite sides of a line segment may not be compatible with each other. Thus, when generating Steiner triangulation, we can only accommodate holes that are either simple polygons or isolated points.

Under this assumption, we can now follow the approach in [5] to show that the triangulation of the $\mathcal{D}$ approximates the MWST when δ is sufficiently small (e.g. $\mathcal{D}$ is untruncated).

Theorem 1. *Given a polygon P of n vertices, the weight of the triangulation of $\mathcal{D}$ is bounded by $O(w(T) + \delta n)$, where $w(T)$ is the weight of any Steiner triangulation T of P.*

Proof: We introduce the Steiner points of T as degenerate holes. T can then be viewed as the MWT of the new polygon. If we apply our decomposition algorithm on the new polygon, we will obtain a refinement of $\mathcal{D}$ that has a larger total length than $\mathcal{D}$. Lemma 3 shows the total length of the refined decomposition is bounded by $O(w(T))$. So the triangulation of $\mathcal{D}$ has weight $O(w(T)+\delta n)$

as discussed before. $\square$

If we set $\delta = R/n$, where R is the width of the smallest enclosing square of P, then the triangulation of $\mathcal{D}$ still approximates the MWST as $w(T) \geq R$ for any Steiner triangulation T. The advantage of this setting of δ is that $\mathcal{D}$ can be computed efficiently as proved below.

Theorem 2. *Given a polygon of n vertices, a Steiner triangulation of $O(n \log n)$ size can be computed in $O(n \log n)$ time which approximates the MWST to within a constant factor.*

Proof: We set $\delta = R/n$. Then the quadtree clearly has $O(\log n)$ levels. Since the work done per level of the quadtree is bounded by the complexity of cells created and crowded subpolygons at that level, we bound this quantity below. Consider one level of the quadtree. Let b be a box at this level. Let α be a crowded subpolygon of b or a cell created at b. Our approach is to charge non-polygon vertices of α to some nearby polygon vertices.

Case 1: α contains some polygon vertex. Then we charge its non-polygon vertices to its polygon vertices. Each polygon vertex is charged at most ten times this way.

Case 2: α is a cell without any polygon vertex and α is bordered by a polygon edge h with an endpoint w in $3p(b)$. Then we charge the vertices of α (at most 8) to w. Observe that α is within a distance of $6size(b)$ from w. Since h borders no more than $12^2 = 144$ such cells within a distance of $6size(b)$ from w, w is charged at most $144 \times 8 = 1152$ times this way.

Case 3: α is a cell without any polygon vertex and the endpoints of each polygon edge bordering α are outside $3p(b)$. Let α' be the subpolygon of $p(b)$ that contains α. Since α' was split, $zone(\alpha')$ contains a polygon vertex. Thus, $zone(\alpha')$ has some polygon vertex v that can see α. We charge the vertices of α to v. We can similarly argue as in case 2 that v is charged at most $144 \times 8 = 1152$ times this way.

Combining cases 1, 2 and 3, we conclude that the complexity of cells created and crowded subpolygons is $O(n)$ at each level of the quadtree. Hence, the results follow. $\square$

5 Ray shooting

5.1 Average case

We briefly explain the average-case model given in [1] and point out why it is better to triangulate the interior of P alone when the exterior of polygon is inaccessible to light. Recall that the average is taken over all random choices of light source on the boundary of P and over all random choices of ray direction.

We first relax to talk about any planar subdivision $\mathcal{S}$ (not necessarily a triangulation) of the interior of P. We look at the *line stabbing query*: given a query line ℓ, report the boundary edges of P intersected by ℓ. For line ℓ, let

$o(\ell)$ and $t(\ell)$ be the number of boundary edges of P and edges of $\mathcal{S}$ intersected by ℓ respectively. Let μ be the standard measure on lines invariant under rigid motions. Let $L(U)$ be the set of lines that intersect a set of line segments U. Then for a query line, the average number of subdivision edges intersected per boundary edge intersected is given by $\int t(\ell)d\mu / \int o(\ell)d\mu$. Denote this ratio by $\Gamma(\mathcal{S})$. One can interpret the ratio as: if one starts from a boundary edge, then one expects to encounter an average of $\Gamma(\mathcal{S})$ subdivision edges before exiting the interior of P. This is exactly the average-case ray shooting complexity.

Standard result in integral geometry states that for any line segment e, $\mu(L(e))$ is two times the length of e. Therefore, $\Gamma(\mathcal{S})$ can be rewritten as the ratio of the total length of $\mathcal{S}$ to the perimeter of P. Hence, it is desirable to obtain a planar subdivision $\mathcal{S}$ with minimum total length. For ray shooting purposes, one must be able to determine what is the next face to visit efficiently. Therefore, it is often required that each face in $\mathcal{S}$ has at most a constant number of adjacent faces. A Steiner triangulation of P is a natural choice.

How does our triangulation compare with the triangulation in [1]? The first step in [1] is to compute a quadtree decomposition of the vertices of P. Then this decomposition will basically be triangulated to produce a Steiner triangulation. Figure 3 shows that the quadtree decomposition of the vertices may already have total length $\Omega(\log n)$ away from some Steiner triangulation of P. Thus, it is advantageous to triangulate the interior of P alone.

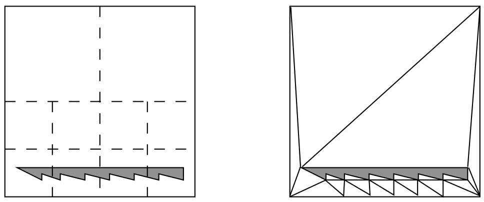

Fig. 3. In the left figure, the dense row of vertices on the hole forces the quadtree decomposition to add horizontal line segments across the polygon as many as $\Omega(\log n)$ times. Thus, the width of the polygon will be added $\Omega(\log n)$ times. In contrast, as shown in the right figure, one can triangulate the polygon interior so that the width of the polygon is also added a constant number of times.

5.2 Query-sensitive ray shooting

Given a query ray r, a planar subdivision and an accompanying data structure is described in [7] that answers ray shooting in $O(\log n + cscc(r))$ time, where $cscc(r)$ is the minimum number of ϵ-strongly simple C-balls that cover r up to the first intersection. The space required is $O(n)$. In the following, we show that the Steiner triangulation resulting from the untruncated $\mathcal{D}$ also supports query-sensitive ray shooting.

Lemma 4. *Let B be an ϵ-strongly simple C-ball of radius s. If B intersects a cell in the untruncated $\mathcal{D}$ of size s', then $s' \geq \epsilon s/4\sqrt{2}$.*

Proof: Let γ be any cell intersected by B in the untruncated $\mathcal{D}$. Assume that to the contrary that $size(\gamma) < \epsilon s/4\sqrt{2}$. Let α be the subpolygon of the parent of the home box of γ that contains γ. Since $size(\alpha) < \epsilon s/2\sqrt{2}$ and α intersects B, $zone(\alpha)$ lies completely within $(1+\epsilon)B$. Since $(1+\epsilon)B$ is a simple ball, $zone(\alpha)$ contains at most one polygon vertex. Thus, α would not be crowded, contradiction. ◻

Lemma 4 implies that the untruncated $\mathcal{D}$, and hence its triangulation, has $O(cscc(P))$ size. Similarly, for any ray r, the number of triangles traversed by r up to the first intersection is $O(cscc(r))$. One can locate the cell containing the light source in $O(\log cscc(P))$ time using point location.

Theorem 3. *Given a polygon of n vertices, there is an approximate MWST of $O(cscc(P))$ size such that given a query ray r, the ray shooting query can be answered using the approximate MWST in $O(\log cscc(P) + cscc(r))$ time.*

As argued in [4, 7], $cscc(P)$ is expected to be small in practice (e.g. $O(n)$) and so the query time in Theorem 3 may often be good enough. When $cscc(P)$ is actually large, we can alternatively locate the light source by descending the quadtree. This gives a running time of $O(\log(R/r)\log n)$, where R is the width of the minimum enclosing square of P, and r is the size of the largest ϵ-strongly C-simple ball containing the light source. When the local geometry around the light source is very simple, one expects $O(\log(R/r))$ to be small.

References

1. B. Aronov and S. Fortune, Average-case ray shooting and minimum weight triangulation, in *Proc. 13th ACM Symposium on Computational Geometry*, 204–211, 1997.
2. S. Arya, S.W. Cheng, and D.M. Mount, Approximation algorithms for multiple-tool milling, *Proc. 14th ACM Symposium on Computational Geometry* 1998, 297–306.
3. M. Bern, D. Eppstein, and J. Gilbert, Provably good mesh generation, *Journal of Computer and System Sciences*, 48 (1994) 384–409.
4. M. de Berg, M. Katz, A.F. van der Stappen, J. Vleugels, Realistic input models for geometric algorithms, in *Proc. 13th ACM Symposium on Computational Geometry*, 294–303, 1997.
5. D. Eppstein, Approximating the minimum weight Steiner triangulation, *Discrete & Computational Geometry*, 11 (1994) 163–191.
6. K.Y. Fung, T.M. Nicholl, R.E. Tarjan, and C.J. Van Wyk, Simplified linear-time Jordan sorting and polygon clipping. *Information Processing Letters*, 35 (1990) 85–92.
7. J.S.B. Mitchell, D.M. Mount, and S. Suri, Query-sensitive ray shooting, *International Journal of Computational Geometry & Applications*, 7 (1997) 317–347.
8. F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.