The Visibility Graph Contains a Bounded-Degree Spanner

Gautam Das*

Abstract

Given a collection of polygonal obstacles with n vertices on the place, and any $t > 1$, we present an $O(n \log n)$ time algorithm that constructs a bounded-degree t-spanner of the visibility graph, without first having to construct the visibility graph.

1 Introduction

An *Euclidean graph* is defined as a graph whose vertices are points in k-dimensional space, edges are line segments joining pairs of points, and edge weights are from the underlying distance metric, typically the L_2 metric. If all edges are present, the graph is a *complete Euclidean graph*, otherwise it is a *non-complete Euclidean graph.* A well-known example of a non-complete Euclidean graph is the *visibility graph*, defined as follows. Consider a scenario where we are given a collection of pairwise disjoint polygons on the plane. This frequently arises in motion planning problems, where the polygons represent obstacles in a cluttered workspace within which a point robot has to navigate. Consider a graph over the polygon vertices, where an edge (u, v) belongs to the graph if the corresponding line segment does not intersect the interior of any obstacle. Such a graph is known as the *visibility graph.* This graph is useful because it contains the shortest obstacle-avoiding path between any pair of vertices. Visibility graphs have been the subject of a great deal of recent research, from both computational and combinatorial aspects (for example, see [14, 16, 17, 22]). It is known that while visibility graphs are not necessarily complete, they can be quite dense, with as many as $\Omega(n^2)$ edges and $\Omega(n)$ degree. It is of interest to investigate whether a visibility graph contains a sparse subgraph which "approximates" shortest paths between all pairs of vertices. Such a subgraph would be a more compact structure in motion planning applications, or in applications where a communication network is being designed (for example, a road network linking all the vertices).

We make this notion of "approximate" shortest paths more precise. Let $G = (V, E)$ be a n-vertex connected graph with positive edge weights. A subgraph G' is a *t-spanner* if for all $u, v \in V$, the distance between u and v in the subgraph is at most t times the corresponding distance in G. The value t is known as the *stretch factor* of the spanner. Spanners are important structures since they represent the original graph more compactly, albeit approximately. In constructing spanners, it is frequently necessary to endow them with additional properties, such as few edges, small total weight, small degree, small diameter, etc. Spanners of arbitrary weighted graphs as well as special classes of graphs such as complete Euclidean graphs have been the subject of much recent research. Spanners find applications in a variety of areas: communication network design, distributed algorithms, network routing, computational geometry and robotics. They are also fascinating from a theoretical point of view, and possess many interesting combinatorial and geometric properties. A good bibliography of past spanner research may be found in [5]. Additionally, in this paper we list several recent references.

While spanners of complete Euclidean graphs have been well studied, relatively little work has been accomplished on non-complete Euclidean graphs, such as for example, visibility graphs. An early result is by Clarkson ([6]) who showed how to construct, for any $t > 1$, a linear-sized t-spanner of the visibility graph in $O(n \log n)$ time without having to first construct the visibility graph. Clarkson (and later Chen, [7]) applied this spanner to solving approximate shortest path problems. In [8], Chew shows that the *constrained Delaunay triangulation* is a planar $O(1)$-spanner of the visibility graph, and can be constructed in $O(n \log n)$ time (however, in this result the stretch factor cannot be arbitrarily close to 1). Recently an $O(n \log n)$ time algorithm has been designed by Arikati et. al. ([2]) to construct, for any $t > 1$, a *Steiner* t-spanner (here the spanner is not strictly a subgraph of the visibility graph because it may contain additional Steiner vertices and edges, however distances be-

*Dept. of Mathematical Sci., The Univ. of Memphis, Memphis, TN 38152, USA, dasg@next1.msci.memphis.edu

tween obstacle vertices still stretch by at most t). These Steiner spanners find applications in answering all-pairs shortest path queries amidst obstacles.

But suppose we are interested in constructing a t-spanner such that, (a) it is a subgraph of the visibility graph, and (b) it has *bounded degree*? The problem is interesting from a theoretical standpoint, because we are trying to discover new combinatorial and geometric properties of visibility graphs and their subgraphs. From a practical standpoint, such a spanner may be used in the design of a road network linking all obstacle vertices, where the objective is to decrease congestion by only allowing a few links to be incident to any vertex. We mention that the corresponding bounded-degree spanner problem for complete Euclidean graphs in k-dimensional space has attracted considerable attention recently (for example, see [3, 5, 9, 20]). However bounded-degree spanners of visibility graphs seem harder to construct, mainly because previously developed techniques for complete Euclidean graphs cannot be immediately used (such techniques rely on the fact that "any vertex can be joined with any other vertex", which is not true when there are obstacles).

In this paper we have developed an algorithm for constructing bounded-degree spanners of visibility graphs. Our algorithm combines a few old techniques with several new techniques. For example, the algorithm is loosely based on the "covering by cones" paradigm, which in the past has been very useful in spanner construction (see [1, 6, 18]). What is interesting is that we extend the idea much beyond its earlier scope, for example when we have to deal with the special geometric constraints that the polygonal obstacles pose. The following theorem summarizes our result.

Theorem 1.1 *Given a set of polygonal obstacles with n vertices in the plane, and any $t > 1$, a bounded-degree t-spanner of the visibility graph exists, and can be constructed in $O(n \log n)$ time. The constants implicit in the big-O depend on t.*

The rest of the paper is organized as follows. In Section 2 we review Clarkson's spanner (see [6]), because it provides the foundation for our spanner algorithm. In Section 3 we show how to construct a bounded-degree spanner, thereby proving Theorem 1.1. We present some open problems in Section 4.

2 Clarkson's Spanner

In this section we start by reviewing Clarkson's spanner (see [6]), which has linear size, but may not have bounded degree. (In our presentation, we retain the main ideas, but present the algorithm somewhat differently. For example, we use a plane sweep, and also use the concept of *projected distances*).

Consider an infinite horizontal line passing through an arbitrary point z. Let θ be a small constant angle (its actual value depends on the given t) and let $L_1, L_2, \ldots, L_{\pi/2\theta-1}$ be semi-infinite rays radiating downward from z such that the angle between adjacent rays is θ. This partitions the lower half-plane into a constant number of unbounded triangles called *cones*, $C_1, C_2, \ldots, C_{\pi/2\theta}$. For each cone C_i, define the *axis* R_i as the semi-infinite ray from z which angularly bisects the cone. Let y be any other point in the interior of C_i. The *projected distance* between z and y, $proj(z, y)$, is defined to be the distance between z and the projection of y on R_i. For a small θ, clearly the projected distance is almost equal to the *actual distance*, $d(z, y)$. (Projected distances were first used in [18] for constructing spanners).

The algorithm sweeps the plane in a particular direction (say from bottom to top, to be consistent with the diagrams to be introduced later), and on encountering a vertex v, decides to select only some of the visibility graph edges that connect it to the vertices below. The selection of visibility graph edges is simple. Translate all the cones $C_1, C_2, \ldots, C_{\pi/2\theta}$ such that their apexes become v. For every cone C_i, of all the visibility graph edges incident to v and contained within C_i, the algorithm selects the edge with the shortest projected length. Once the sweep is over, the selected edges represent the spanner, G.

It is easy to see that the output has a linear number of edges, since at every vertex at most a constant number of edges are selected (at most one per cone). However, the degree may not be a constant. To see this, imagine that the algorithm is actually creating a *directed* graph; at vertex v the edges that are selected are given downward directions (from v to the other endpoints below). While the output graph has a bounded out-degree, it may have an unbounded in-degree. The output is also a t-spanner of the visibility graph; for a proof we refer the reader to [6]. The algorithm can be implemented to run in $O(n \log n)$ time without having to first create the visibility graph, using techniques such as planar point location and *conical Voronoi* *diagrams*; the details are in [6].

3 Bounding the Degree

In this section we describe our more complex algorithm, which produces a bounded-degree spanner. The algorithm consists of four steps.

Step 1: *Construction of a linear-sized spanner*:

Select numbers t_1 and t_2 such that $t_1, t_2 > 1$, t_1 is very close to 1, t_2 is somewhat bigger but still small enough so that $t_1 \cdot t_2$ is closer to 1 than to t.[1] Create a t_1-spanner of the visibility graph, using Clarkson's algorithm. Let this spanner be G.

Step 2: *Partition into forests*:

Recall that we had used $O(1)$ cones $C_1, C_2, \ldots, C_{\pi/2\theta}$ in the previous algorithm. We are going to refine this even further. Select an angle α to be much smaller than θ but which divides θ evenly (some intuition about its value is given in Step 3). We partition each cone C_i into $O(1)$ *subcones* $C_{i,1}, C_{i,2}, \ldots, C_{i,\theta/\alpha}$, such that the angle of each subcone is α. The following terms will be useful in our explanations: the subcones in the central region of C_i are known as *central subcones*, while the subcones to the far left or far right of C_i are known as *peripheral subcones*. Notice that the total number of subcones over all the cones is $\pi/2\alpha$, which is a constant.

We partition G into a constant number of subgraphs (actually forests) as follows. Consider G as a directed graph, and let (v, u) be any directed edge of G (i.e. directed downwards from v to u). If it lies inside the subcone $C_{i,j}$ (with apex at v), then (v, u) belongs to the graph $G_{i,j}$. It is easy to see that each $G_{i,j}$ has an outdegree of one, but may have an unbounded indegree. Clearly the undirected version of $G_{i,j}$ is a forest.

Step 3: *Removal of more edges*:

In this step, the algorithm performs a downward sweep of G (actually over all the $G_{i,j}$'s simultaneously), and at each vertex possibly removes some of the incoming edges. The output graph, G', will be a spanner of the visibility graph. It will have a smaller, though still not constant, in-degree. The stretch factor will be larger than that of G. The logic for selection or removal of edges of $G_{i,j}$ is as follows. Let c be a constant (approximately) equal to $\frac{t_1 \cdot t_2 - t_2}{t_1 \cdot t_2 - 1}$ (the exact value of c is somewhat different; it also depends on α and θ, but we omit details from this version). Notice that $0 < c < 1$. Since t_1 is selected very close to 1, c is very close to 0. Suppose the downward sweep visits a vertex v. Sort the incoming edges of $G_{i,j}$ at v by increasing length. Select the shortest, whose length is say l_1. Remove all edges of length at most l_1/c. Select the smallest of the remaining, say l_2. Remove all edges of length at most l_2/c. Repeat this process until all incoming edges are either selected or removed. Do this for all the forests, then sweep downwards to the next vertex.

Once the sweep is over, each forest $G_{i,j}$ has been reduced to a sparser forest, say $G'_{i,j}$. The union of these forests, G', is the output of Step 3.

Before we describe Step 4, let us analyze G', the output of Step 3. Clearly the in-degree of any $G'_{i,j}$ may not be bounded. However if two incoming edges of $G'_{i,j}$ meet at a common vertex, then one edge is much longer than the other. In addition, the following lemma shows that G' is a spanner.

Lemma 3.1 *G' is a $(t_1 \cdot t_2)$-spanner of the visibility graph.*

Proof : In this version we use very rough calculations and simply sketch the proof. Essentially we have to show that G' is a t_2-spanner of G. Suppose an edge of G, say (u, v), is removed while the algorithm is visiting v. There should be an alternate path in G' of length at most $t_2 \cdot d(u, v)$. Suppose (u, v) originally belonged to some $G_{i,j}$. Then some other edge, (w, v) belongs to $G'_{i,j}$ such that $d(u, v) < d(w, v)/c$. Let us use Figure 1 as an illustration. For both u and w, the position of their C_i cone is shown, where the right ray of u's cone intersects the left ray of v's cone at x. It is not hard to see that the line segments (u, x) and (x, w) *do not intersect other obstacles.* We also observe that the two edges (u, v) and (w, v) are almost parallel, since α is very small. Finally the angle uxw is θ, and since α is much smaller than θ, the length of (x, w) is negligible compared to the length of (v, w).

Define $cv(w, x, u)$ to be the convex chain obtained by placing a stretched rubber band from w to x to

[1]Due to lack of space, we will avoid deriving the exact values of the various constants used in this version of the paper

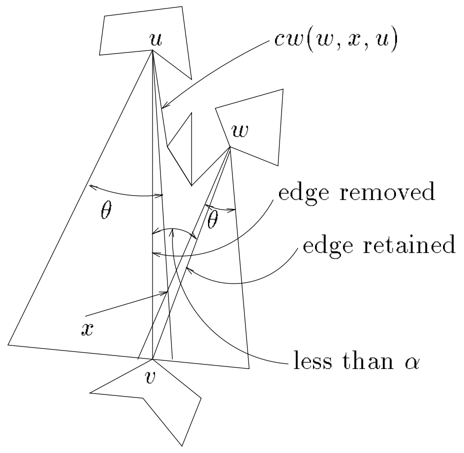

Figure 1: Analysis of Step 3

u, anchoring it at w and u and releasing it at x. It will assume a convex shape because the shrinkage will be halted by various obstacles. The chain is confined within the triangle wxu, where (w,u) is defined as the *base boundary* and (w,x) and (x,u) are defined as the *side boundaries* of the chain. Note that this chain is a path from w to u in the original visibility graph. By an induction argument which we omit, it can be assumed that there is a path P from w to u in G' of length at most $t_1 \cdot t_2$ times the length of $cv(w,x,u)$. We thus have an alternate path Q from v to u in G' as follows: go from v to w, then go along P from w to v. The length of Q is at most $d(v,w) + t_1 \cdot t_2 \cdot (d(w,x) + d(x,u))$. But since (v,u) and (v,w) are almost parallel, and $d(x,w)$ is negligible compared to (v,w), the length of Q is maximized when $d(u,w)$ is the smallest possible, i.e. when $d(u,w) = c \cdot d(v,u)$. In this situation $d(u,x) + d(x,v)$ is approximately equal to $(1-c) \cdot d(u,v)$. Substituting these quantities in, we get the length of Q to be approximately at most $c \cdot d(v,u) + t_1 \cdot t_2 \cdot (1-c) \cdot d(u,v)$. Substituting for c, this simplifies to at most $t_2 \cdot d(u,v)$. Thus we can conclude that G' is a t_2-spanner of G, hence a $(t_1 \cdot t_2)$-spanner of the visibility graph. ■

At this stage we can make some more observations. Consider a $G_{i,j}$ whose corresponding subcone is $C_{i,j}$. Let v be a vertex with a large in-degree. If $C_{i,j}$ is a central subcone, then all the incoming edges at v will be approximately of the same length (since α is much smaller than θ). In this case, Step 3 will remove all but the shortest edge. On the other hand, if $C_{i,j}$ is a peripheral subcone, then the incoming edges at v could be of all possible lengths.

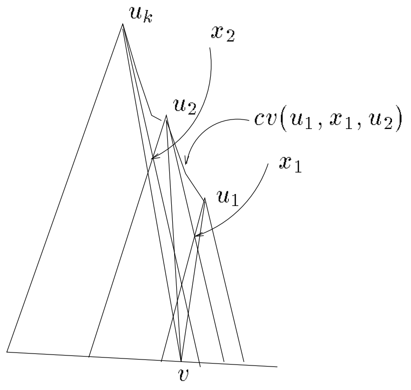

Figure 2: Constructing the bounded-degree spanner

Though Step 3 prunes some of them, an unbounded number could be left over. We can only guarantee that for any pair of left over edges, the shorter edge is much shorter (at most c times) than the longer edge. Figure 2 (or a symmetric equivalent) correctly describes the geometry of all incoming edges of $G'_{i,j}$ at v. Notice that the order of these edges by length is the same as their order by slope.

Step 4: *Achieving bounded degree*:

This final step is different from the others because here we remove more edges, and also *add back* some edges. The idea is to consider each $G'_{i,j}$, and create another forest $G''_{i,j}$ which has bounded degree. However, $G''_{i,j}$ is not necessarily a subgraph of $G'_{i,j}$. The union of all $G''_{i,j}$, defined as G'', is the final output of our algorithm.

The details of the construction of G'' are simple. For all $G'_{i,j}$, for all vertex v of $G'_{i,j}$, perform the following operation. Let $(u_1,v),(u_2,v),\ldots,(u_k,v)$ be the incoming edges at v in increasing length (see Figure 2). Retain (u_1,v). Remove $(u_2,v),(u_3,v),\ldots,(u_r,v)$. *Add* the chains $cv(u_1,x_1,u_2), cv(u_2,x_2,u_3),\ldots,cv(u_{k-1},x_{k-1},u_k)$.

This completes the description of the algorithm. We now analyze the algorithm and its output G''.

Lemma 3.2 *G'' is a t-spanner of the visibility graph.*

Proof : We sketch the proof. Consider any edge (u_m,v) of G' that got removed. Consider the alternate path in G'': go from v to u_1, then along $cv(u_1,x_1,u_2)$, then along $cv(u_2,x_2,u_3)$, and so on until you reach u_m. Since α is small, and the constant c (used in Step 3) is small, from the geometry of the situation it can be shown that this path is not much longer than $d(u_m, v)$. In fact, by selecting smaller α and smaller c (which can be achieved by selecting t_1 closer to 1), we can make this path length as close to $d(u_m, v)$ as we like. Since G' is a $(t_1 \cdot t_2)$-spanner of the visibility graph, it follows that G'' is also a spanner of the visibility graph, but with a slightly larger stretch factor. Since $t_1 \cdot t_2$ has been selected to be closer to 1 than to t, we can make sure that the stretch factor of G'' is no more than t. ■

The next two lemmas eventually show that the degree of G'' is bounded. In the construction of $G''_{i,j}$, recall that at a vertex v, all incoming edges except the shortest are removed, and several convex chains are added instead. For each such convex chain, we define its *base vertex* to be v.

Lemma 3.3 *If we start with the empty plane, and draw on it all the convex chains of $G''_{i,j}$ and their respective side boundaries, all the line segments in the arrangement will be pairwise disjoint, except possibly for sharing common endpoints.*

Proof : Let $cv_1 = cv(u_m, x_m, u_{m+1})$ and $cv_2 = cv(u_p, x_p, u_{p+1})$ be any two convex chains of $G''_{i,j}$. If they have the same base vertex, v, then the lemma is clearly true (as Figure 2 shows). On the other hand, suppose the two chains have different base vertices, say v_1 and v_2 respectively. This involves a detailed case analysis, which we omit. Instead, consider Figure 3 which shows the most "crowded" situation where we try and make the two chains intersect each other. However, since v_1 is an obstacle vertex, the convex chain cv_2 either includes v_1 or is below it. Thus it neither intersects cv_1 nor the side boundaries of cv_1. ■

Lemma 3.4 *G'' has bounded degree.*

Proof : Since there a constant number of $G''_{i,j}$ it will suffice to prove that each of the latter has bounded degree. Consider any $G''_{i,j}$. It has two kinds of edges, the convex chains, and the shortest edges retained from $G'_{i,j}$ at every vertex. Consider the subgraph consisting of the convex chains. Due to Lemma 3.3, its degree is at most 2. Next consider the subgraph consisting of the shortest edges. The degree of this subgraph is at most 2, because at any vertex there is at most one incoming edge and at most one outgoing edge. Thus the degree of $G''_{i,j}$ is bounded, which implies that the degree of G'' is also bounded. ■

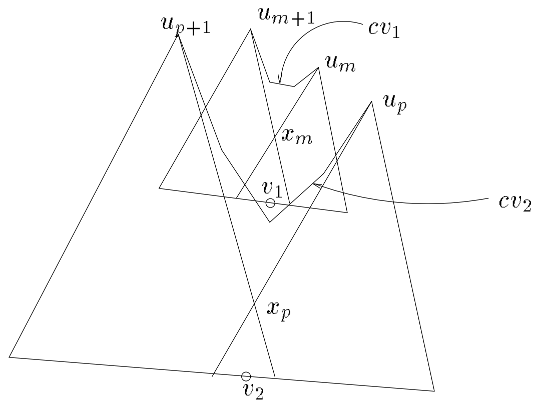

Figure 3: Convex chains and side boundaries are disjoint

We now present an efficient implementation of the algorithm. Step 1 takes $O(n \log n)$ time. Step 2 takes $O(n)$ time since there are $O(n)$ edges in G and assigning an edge to its respective forest takes $O(1)$ time. Step 3 involves sorting and thus takes $O(n \log n)$ time per $G_{i,j}$, thus $O(n \log n)$ time overall.

Implementing Step 4 is nontrivial. For each forest $G'_{i,j}$ the main task is to compute several convex chains. Let the complete set of convex chains to be constructed for $G'_{i,j}$ be $CV_{i,j}$. First start with the empty plane, and lay out the convex hull of the original set of obstacles. Then for each chain in $CV_{i,j}$, lay out its two side boundaries on the plane (the two side boundaries are known, even if the chain is not yet computed). This will result in a collection of line segments in the interior of the hull, pairwise disjoint except at endpoints (this is due to Lemma 3.3). Treat these line segments as edge obstacles, and compute a trapezoidal decomposition of the interior of the hull, where the parallel boundaries of the trapezoids are parallel to the axis of the cone C_i. Using planar point location, assign each original obstacle vertex to the trapezoid that contains it. For each chain cv_r, let the set of obstacle vertices inside the trapezoids immediately above the chain's side boundaries be V_r. Using an optimal convex hull algorithm, compute cv_r *using only* V_r as input. It should be clear that for two different chains cv_r and cv_s, the corresponding sets V_r and V_s are disjoint. Thus computing $G''_{i,j}$ takes $O(n \log n)$ time, and therefore Step 4 takes $O(n \log n)$ time. Thus the entire algorithm runs in $O(n \log n)$ time.

4 Open Problems

We conclude this paper with some open problems.

The foremost open problem is, are there are spanners of the visibility graph with a maximum degree of 3? Notice that the spanner in Theorem 1.1 has $O(1)$ degree, but that sheds no light on whether a degree-4 (or even a degree-3) spanner of the visibility graph exists. We mention that degree-3 spanners exist for the case of complete Euclidean graphs without obstacles ([9]). That paper also shows that 3 is a lower bound on the degree. It would be interesting to extend this to the case with obstacles, however we feel this may require considerably more complicated techniques than used in [9].

For complete Euclidean graphs, spanners are known with *several* sparseness properties, in addition to small degree. Some examples are low weight, and small diameter ([4]). Extending these results to the case of spanners of visibility graphs is a challenging problem.

In general, the problem of constructing spanners of non-complete Euclidean graphs (not just visibility graphs) in both two as well as higher dimensions needs to be studied.

References

[1] I. Althöfer and G. Das and D. P. Dobkin and D. A. Joseph and J. Soares: On sparse spanners of weighted graphs: Discrete Comput. Geom., Vol 9, 1993, pp 81–100.

[2] S. Arikati, D. Chen, L. P. Chew, G. Das, M. Smid, C. D. Zaroliagis: Planar spanners and approximate shortest path queries among obstacles in the plane: Proc. European Symp. on Algorithms (ESA), 1996.

[3] S. Arya and M. Smid: Efficient construction of a bounded degree spanner with low weight: Proc. 2nd Annu. European Sympos. Algorithms (ESA), Lecture Notes in Computer Science, Vol 855, 1994, pp 48–59.

[4] S. Arya and G. Das and D. M. Mount and J. S. Salowe and M. Smid: Euclidean spanners: short, thin and lanky: Proc. ACM Sympos. Theory of Comput. (STOC), 1995.

[5] B. Chandra and G. Das and G. Narasimhan and J. Soares: New sparseness results on graph spanners: Proc. 8th Annu. ACM Sympos. Comput. Geom., 1992, pp 192–201.

[6] K. L. Clarkson: Approximation algorithms for shortest path motion planning: Proc. ACM Sympos. Theory of Comput. (STOC), 1987, pp 56–56.

[7] D. Z. Chen: On the all pairs Euclidean short path problem: Proc. SIAM-ACM Sympos. on Discrete Algorithms (SODA), 1995.

[8] L. P. Chew: There are planar graphs almost as good as the complete graph: J. of Computer and System Sciences, 39, 1989, 205–219.

[9] G. Das and P. Heffernan: Constructing degree-3 spanners with other sparseness properties: Intl. Sympos. Algorithms and Comput. (ISAAC), 1993.

[10] G. Das and P. Heffernan and G. Narasimhan: Optimally sparse spanners in 3-dimensional Euclidean space: Proc. 9th Annu. ACM Sympos. Comput. Geom., 1993, pp 53–62.

[11] G. Das and G. Narasimhan and J. S. Salowe: A new way to weigh malnourished Euclidean graphs: Proc. 6th SIAM-ACM Sympos. Discrete Algorithms (SODA), 1995.

[12] G. Das and G. Narasimhan: A fast algorithm for constructing sparse Euclidean spanners: Proc. 10th Annu. ACM Sympos. Comput. Geom., 1994.

[13] D. Eppstein: Spanning Trees and Spanners: Tech. Report 96-16, Dept. of ICS, UC Irvine, 1996.

[14] S. K. Ghosh and D. M. Mount: An output-sensitive algorithm for Computing visibility graphs: SIAM J. Comput. 20, 1991, pp 888–910.

[15] K. Mehlhorn: Data structures and algorithms 1: sorting and searching: Springer-Verlag, 1984, pp 290–296.

[16] M. H. Overmars and E. Wetzel: New methods for computing visibility graphs: Proc. 4th Annu. ACM Sympos. Comput. Geom., 1988, pp 164–171.

[17] M. Pocchiola and G. Vegter: The visibility complex: Proc. 9th Annu. ACM Sympos. Comput. Geom., 1993, pp 328–337.

[18] J. Ruppert and R. Seidel: Approximating the d-dimensional complete Euclidean graph: Proc. 3rd Canad. Conf. Comput. Geom., 1991, pp 207–210.

[19] J. S. Salowe: Construction of multidimensional spanner graphs with applications to minimum spanning trees: Proc. 8th Annu. ACM Sympos. Comput. Geom., 1991, pp 256–261.

[20] J. S. Salowe: On Euclidean spanner graphs with small degree: Proc. 8th Annu. ACM Sympos. Comput. Geom., 1992, pp 186–191.

[21] P. M. Vaidya: A sparse graph almost as good as the complete graph on points in K dimensions: Discrete and Comput. Geom., 6, 1991, pp 369–381.

[22] E. Welzl: Constructing the visibility graph for n line segments in $O(n^2)$ time: Inform. Process. Letters, 20, 1985, pp 167–171.