Viewing a Set of Spheres while Moving on a Linear Flightpath

Frank Follert [*]

April 22, 1996

Abstract

In this paper we investigate a hidden surface removal problem with respect to a moving point of view. We describe an algorithm which maintains the visibility graph of a set of n nonintersecting spheres in 3-space, while the viewpoint moves along a given linear flightpath. To this end, the algorithm computes the initial visibility graph of the spheres and determines all points in time at which the topology of the visibility graph changes as well as the corresponding topology changes. The algorithm runs in time $O(n^{\frac{8}{5}+\epsilon} + (co_{per} + tc_{tr}) \log n)$, where co_{per} denotes the number of pairs of spheres, which have intersecting projections during the entire motion, and tc_{tr} denotes the number of transparent topology changes, i.e., the number of topology changes under the assumption that all spheres are transparent. (Throughout the paper, ϵ is an arbitrarily small positive constant.)

1 Introduction

In this paper we investigate a three-dimensional hidden surface removal problem with respect to a moving point of view. Consider a situation where you want to compute the perspective view of a set of fixed nonintersecting spheres in 3-space from a sequence of viewpoints lying on a linear flightpath. Such a situation arises for example in molecular graphics. Here, a molecule is often modeled as the union of atoms, each represented by a sphere of its van der Waals radius. The possibility of viewing such molecules from a prespecified linear flightpath in 3-space can provide valuable insights into their spatial structure. Instead of computing the images from scratch for each viewpoint along the trajectory, we can exploit the fact that the visibility map of the spheres, i.e., the partition of the viewing plane into maximally connected regions in which only one or no sphere is visible, undergoes only minor changes when we move from one viewpoint to the next. We thus can compute the visibility map at the beginning of the motion and update it as we move along the trajectory. For this purpose, it is necessary to precompute a list of critical points on the flightpath. This list contains (a superset of) all the points where the topology of the visibility map changes, along with the topological changes occuring at these points. We describe an algorithm for this problem with running time $O(n^{\frac{8}{5}+\epsilon} + (co_{per} + tc_{tr}) \log n)$, where n denotes the number of spheres, tc_{tr} is the number of *transparent topology changes* (i.e., the number of topology changes along the path assuming transparent spheres), and co_{per} denotes the number of pairs of spheres which have intersecting projections during the entire motion. Note that co_{per} is trivially bounded by $O(n^2)$ and tc_{tr} is bounded by $O(n^3)$. A major open problem is to develop algorithms with a running time which is senstive to the number tc_{op} of *opaque topology changes*, i.e., topology changes which occur in the visibility map assuming opaque spheres. The paper is organized as follows: In section 2 we discuss some related work, section 3 contains a formal description of the problem and some basic properties, in section 4 we present the algorithm in detail and section 5 concludes with some remarks and open problems.

2 Related Work

The study of hidden surface removal problems with a moving point of view has been initiated in the pa-

[*]Universität des Saarlandes, Fachbereich 14, Informatik, Lehrstuhl Prof. G. Hotz, Postfach 151150, 66041 Saarbrücken, Germany. Email: `follert@cs.uni-sb.de`, Fax: +49 681 302 4421. The author is supported by a Graduiertenkolleg fellowship from DFG, Germany

per [2]. The authors investigate the situation of a viewpoint moving on a linear flightpath through a polygonal scene. They present algorithms with running times $O((n^2+tc_{tr})\log n)$, $O(n^2+tc_{tr}\log n)$ and $O(n^2\log n + tc_{tr})$ respectively, where tc_{tr} denotes the number of transparent topology changes. For linear flightpaths through terrains, they obtain an $O((n + tc_{op})\lambda_3(n)\log n))$ algorithm, tc_{op} being the number of opaque topology changes. In the case of a vertical flightpath in a terrain, they give an $O(n\lambda_4(n)\log n)$ algorithm, which matches an earlier result of Cole and Sharir [3]. ($\lambda_4(n)$ denotes the slightly superlinear length of a Davenport-Schinzel sequence of order 4). Mulmuley [7] introduces the concept of semi-opaque topology changes and describes an $O(tc_{sop}\log n + n^2\alpha(n)\log n)$ algorithm for a linear flightpath in a polygonal scene, where tc_{sop} denotes the number of semi-opaque topology changes. Lenhof and Smid [6] consider scenes of nonintersecting spheres for the first time. Their algorithm, which computes parallel views from a viewpoint moving on a circle at infinity takes $O((n+tc_{tr}+co_{per})\log n)$ time. Based on this approach, we show how to solve the related (but apparently more complicated) problem of a viewpoint moving on a linear flightpath amidst nonintersecting spheres efficiently.

3 The Problem

We will now give a more formal description of the problem under consideration. Let $\mathcal{B}$ denote a set of n nonintersecting spheres $b_1,\ldots,b_n$ in 3-space, the *scene*. Each sphere $b_i(x_i,r_i)$ is specified by its center x_i and radius r_i. We assume that the viewpoint moves monotonically on an oriented line segment f in 3-space, which does not intersect any of the spheres in $\mathcal{B}$. We transform the scene in time $O(n)$ such that the flightpath f coincides with the intervall $[0,1]$ on the x-axis of the coordinate frame. The points on f are canonically represented by a monotonically increasing time parameter $t \in [0,1]$, such that $f(0)$ corresponds to the origin and $f(1)$ to the other endpoint of f. The point on f with parameter value t is denoted by $f(t)$. At time t, we imagine projecting the scene $\mathcal{B}$ from $f(t)$ onto a sphere $\mathcal{S}(t)$ which is centered at $f(t)$ and encloses the whole scene $\mathcal{B}$ at any time. At time t, the sphere b_i projects to a disc $d_i(t)$ with boundary circle $c_i(t)$ on the sphere $\mathcal{S}(t)$. The collection of circles $\mathcal{C}_t = \cup_i c_i(t)$ induces a graph G_t,the *transparent visibility graph*, on $\mathcal{S}(t)$: its vertices correspond to the points of intersection of the circles in $\mathcal{C}_t$, and its edges correspond to the arcs connecting these intersections. We call a point *visible* from $f(t)$, if the line segment connecting the point with $f(t)$ does not intersect any sphere in $\mathcal{B}$. If we delete all those vertices and edges from G_t which are induced by scene points that are not visible from $f(t)$, we obtain a subgraph of G_t. We label the faces of this subgraph with the spheres visible in it and name the resulting labeled graph the *visibility map* V_t of $\mathcal{B}$ at time t. A *transparent (opaque) topology change* occurs at time t iff the graphs $G_{t-\delta}$ ($V_{t-\delta}$) and $G_{t+\delta}$ ($V_{t+\delta}$) are nonisomorphic for each small $\delta > 0$. Our goal is to compute the initial transparent visibility graph G_0 and an ordered list of all critical points in time t at which transparent topology changes occur, together with a (constant length) description of these changes. Since the transparent topology changes are a superset of the opaque changes, this enables us to compute the maps $V_t, 0 \le t \le 1$, as we move along f. Following a lemma of [6], there are three types of transparent topology changes to be detected: At time t, either two circles $c_i(t)$ and $c_j(t)$ touch in one point (type I) or become identical (type II), or three circles $c_i(t), c_j(t), c_k(t)$ have a common point of intersection (type III).

4 The Algorithm

4.1 Overview

The structure of the algorithm crucially depends on the notion of conflicting spheres. Therefore, we first give a formal definition of this relation. We say that two spheres $b_i, b_j \in \mathcal{B}$ are in *conflict at time* $t \in [0,1]$, iff their projected images, the *discs* $d_i(t), d_j(t) \subseteq \mathcal{S}(t)$ have a nonempty intersection. (Note that two spheres are in conflict if one disc contains the other). Two spheres have a *conflict*, iff there is a time $t \in [0,1]$ at which they are in conflict; they have a *permanent conflict*, iff they are in conflict for all $t \in [0,1]$.
The algorithm which computes all transparent topology changes which arise during the motion along f proceeds in three phases. In the first phase, it computes the initial visibility graph G_0 along with the information which pairs of spheres are in con-

flict at $t = 0$. This certainly includes those pairs of spheres, which have a permanent conflict during the entire motion; let co_{per} denote the number of such pairs. This phase can be implemented by extending the plane-sweep hidden line removal algorithm of [8]. The running time of this phase is bounded by $O((n + co_{per} + tc_{tr}) \log n)$. Next, we determine all pairs of spheres having a conflict, which have not been detected in the first phase. This is done by employing recent results on range searching with semialgebraic sets. Phase two takes $O(n^{\frac{9}{5}+\epsilon} + tc_{tr})$ time. With this information, phase three of the algorithm can compute the transparent topology changes of types I and II in time $O(tc_{tr} + co_{per})$. Subsequently, the algorithm detects the transparent topology changes of type III by a series of sweep procedures. This can be done in time $O((tc_{tr} + co_{per}) \log n)$. Finally, sorting the transparent topology changes by increasing t takes time $O(tc_{tr} \log n)$. Summing up, this yields the main theorem.

Theorem 1 *Let $\mathcal{B}$ be a set of n nonintersecting spheres in 3-space, let f be an oriented line segment in 3-space. The sorted sequence of transparent topology changes occuring in the graph G_t as the viewpoint moves along f can be computed in time $O(n^{\frac{9}{5}+\epsilon} + (tc_{tr} + co_{per}) \log n)$, where tc_{tr} is the number of transparent topology changes and co_{per} denotes the number of permanent conflicts.*

We now describe the phases of the algorithm in greater detail.

4.2 Phase 1

In the first phase, the algorithm computes the initial transparent visibility graph G_0 of the scene. This is done by separately computing the visibility graphs seen through each of the six faces of an axis-parallel cube centered at the origin, glueing these parts together and projecting the resulting graph on the sphere $\mathcal{S}(0)$. It is well known [4], that computing the view of the scene $\mathcal{B}$ from the origin through one of these six windows can be reduced to computing the view from above of a set of horizontal discs by a perspective transformation. In order to obtain this parallel view, we use Nurmi's sweepline hidden line elimination procedure for polygonal scenes [8], which can easily be modified to handle circular arcs. Furthermore, the sweep procedure can be extended to report all pairs of spheres conflicting at time $t = 0$: If the projected circles of two conflicting spheres intersect, this intersection is found during the sweep. The case of one circle containing the other can be detected by making use of a special data structure maintained by the algorithm, the node-lists. This data structure can report quickly which circles cover the intervall between two adjacent intersections on the sweepline. Thus, whenever the sweepline touches a new circle, the circles containing it can be easily identified. The time consumed by the sweep procedure is $O((n + co_{per} + k) \log n)$, where k is the number of circle-circle intersections found by the algorithm. Since $k = O(co_{per} + tc_{tr})$, the running time is dominated by $O((n + co_{per} + tc_{tr}) \log n)$.

4.3 Phase 2

The goal of phase two is to quickly enumerate those pairs of spheres which have a conflict. A naive approach to this problem is to simply check all pairs in time $O(n^2)$. Since the number of conflicts can be as large as $O(n^2)$, we cannot hope to find an algorithm with a better worst-case performance. Instead, we would like to obtain a time bound which is subquadratic in n and which depends on the number of conflicting pairs in an output-sensitive manner. This was stated as an open problem in [5]. We now show how this goal can be achieved by reducing the problem to range searching with semialgebraic sets.
The basic idea is to preprocess the set of spheres into a data structure which when queried with a sphere b_i reports the spheres in (nonpermanent) conflict with b_i quickly. To accomplish this, we employ a theorem from [1] concerning range searching with semialgebraic sets.

Theorem 2 (Agarwal, Matousek)
Let $f(x_1, \ldots, x_d, a_1, \ldots, a_p)$ be a $(d+p)$-variate polynomial. Assume that $p, d, q, deg(f)$ are bounded by a constant. Let

$$\Gamma = \{\{x \in \mathbf{R}^d \mid f(x, a^1) \geq 0, \ldots, f(x, a^q) \geq 0\} \mid a^1, \ldots, a^q \in \mathbf{R}^p\}.$$

Then the Γ-range searching problem can be solved with $O(n)$ space, $O(n \log n)$ preprocessing time, and $O(n^{1-1/b+\epsilon})$ query time, where the parameter b satisfies $b = d$ for $d \leq 3$ and $b \leq 2d - 3$ for $d > 3$.

A close inspection of the data structure shows that reporting the points in the query range takes additional time linear in the size of the answer set. The theorem can also be extended to handle range searching with a constant number of conjunctions and disjunctions of (strict or nonstrict) inequalities involving a constant number of fixed bounded degree polynomials, i.e., general semialgebraic sets of constant description complexity. Note that the query time still depends only on the dimension d of the point set in this case.

In order to apply this result, we represent the set $\mathcal{B}$ of spheres as a point set in $\mathbf{R}^4$ with coordinates $(x_i, r_i)\ \forall i$. Now, we have to express the query with a sphere b_i as a query with a semialgebraic set (depending only on the parameters (x_i, r_i) of b_i) which contains exactly those points representing spheres in conflict with b_i.

We start with some basic definitions. Given a point $p \in \mathbf{R}^3$ and a direction vector v, we denote the ray emanating from p with direction v by $r_{p,v}$. We define the set $\mathcal{R}(b_i, b_j)$ to be the set of points in 3-space, for which b_i, b_j are in conflict, when viewed from that point.

$$\mathcal{R}(b_i,b_j) = \{p \in \mathbf{R}^3 \mid \exists v \in \mathbf{R}^3 : r_{p,v} \cap b_i \neq \emptyset \\ \wedge r_{p,v} \cap b_j \neq \emptyset\}$$

The following lemma relates $\mathcal{R}(b_i, b_j)$ to some simply shaped point sets defined by two nonintersecting spheres $b_i(x_i, r_i)$ and $b_j(x_j, r_j)$: Let l_{ij} be the line containing x_i, x_j and pl be an arbitrary plane containing x_i, x_j. pl intersects the spheres b_i, b_j in two circles k_i, k_j. Let t_1, t_2 be the inner tangents to k_i, k_j in pl, i.e., those tangents which cross on the line l_{ij}. Let $cone_{ij}$ be the double cone in $\mathbf{R}^3$ obtained by rotating the double wedge enclosed by t_1, t_2 which contains k_i, k_j around l_{ij}. The surface $conesurf_{ij}$ of this cone touches the spheres b_i, b_j in two circles cir_i, cir_j which lie in two parallel planes pl_i, pl_j. The layer between these two planes is called $layer_{ij}$.

Lemma 1 *Let $b_i(x_i, r_i), b_j(x_j, r_j)$ be two spheres with $b_i \cap b_j = \emptyset$. Then*

$$\mathcal{R}(b_i, b_j) = cone_{ij} - layer_{ij} \cup b_i \cup b_j$$

Proof: A ray emanating from a point p outside $cone_{ij}$ can only intersect one of the two halfcones of $cone_{ij}$ or its apex and thus can't intersect both spheres b_i and b_j. Thus $\mathcal{R}(b_i, b_j) \subseteq cone_{ij}$. Now consider a point p inside $cone_{ij}$ but not contained in $layer_{ij}$. A ray starting at p and passing through the apex of $cone_{ij}$ clearly intersects both spheres. Through each point p in $cone_{ij}$, which is contained

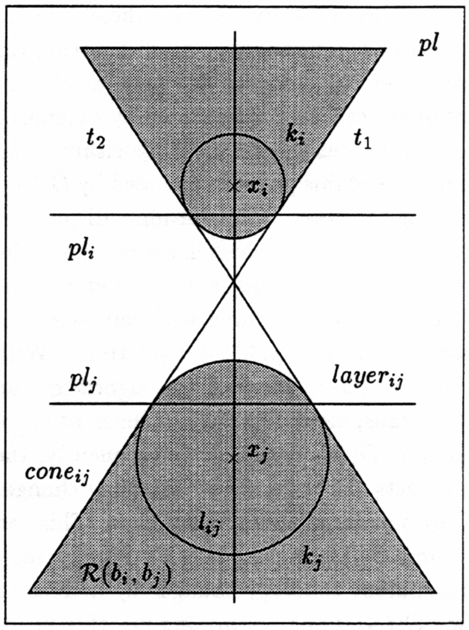

Figure 1: The intersection of $\mathcal{R}(b_i, b_j)$ with plane pl.

in $layer_{ij}$ but in neither of the two spheres, there is a plane separating b_i and b_j. Hence, those points can't be starting points of rays piercing both spheres. Finally, both b_i and b_j obviously belong to $\mathcal{R}(b_i, b_j)$. □

The following observation is crucial: Two spheres are in conflict, iff $\mathcal{R}(b_i, b_j)$ and the flightpath f have a nonempty intersection. Since permanent conflicts $(f \subseteq \mathcal{R})$ are already detected in phase one, we can restrict our attention to nonpermanent conflicts. A nonpermanent conflict occurs iff $\partial\mathcal{R}(b_i, b_j) \cap f \neq \emptyset$. Since f and b_i, b_j are disjoint, we can focus on intersections of $conesurf_{ij}$ with f which are not contained in $layer_{ij}$. Remembering the initial transformation of the scene (f being the intervall $[0, 1]$ on the x-axis), we have to test, whether an intersection of $conesurf_{ij}$ with the x-axis falls into the intervall $[0, 1]$ and is not contained in $layer_{ij}$. Since $conesurf_{ij}$ is a quadratic surface in $\mathbf{R}^3$, its intersections with the x-axis can be described as the at most two roots of a quadratic polynomial $g(t)$. The coefficients of g are low degree polynomials in the parameters x_i, r_i, x_j, r_j of b_i, b_j. Together with the additional constraints we obtain a system of (in)equalities

$$g(x_i, r_i, x_j, r_j, t) = 0 \wedge 0 \leq t \leq 1 \\ \wedge (t, 0, 0) \notin layer_{ij}$$

A straightforward calculation shows that the last constraint can be rewritten as

$$h_1(x_i,r_i,x_j,r_j) \leq t \cdot h_2(x_i,r_i,x_j,r_j)$$
$$\vee\ h_3(x_i,r_i,x_j,r_j) \leq t \cdot h_4(x_i,r_i,x_j,r_j)$$

where h_1, h_2, h_3, h_4 are low degree polynomials in x_i, r_i, x_j, r_j.

In order to obtain a set of polynomial inequalities depending only on the parameters x_i, r_i, x_j, r_j we have to eliminate the time parameter t. For this purpose, we solve the quadratic equation $g(t) = 0$ for t and substitute the resulting expressions $t_1(x_i, r_i, x_j, r_j)$ and $t_2(x_1, r_i, x_j, r_j)$ into the remaining inequalities.

$$dis(x_i,r_i,x_j,r_j) \geq 0 \wedge$$
$$(\ \ (0 \leq t_1 \leq 1 \wedge (h_1 \cdot t_1 \leq h_2 \vee h_3 \cdot t_1 \leq h_4))$$
$$\vee (0 \leq t_2 \leq 1 \wedge (h_1 \cdot t_2 \leq h_2 \vee h_3 \cdot t_2 \leq h_4))\ )$$

Here, $dis(x_i, r_i, x_j, r_j)$ denotes the discriminant polynomial of $g(t)$. These inequalities can be transformed into a set of polynomial inequalities by repeated squaring in order to eliminate square roots. Of course, one has to distingish carefully between different sign cases to maintain the equivalence of the sets of inequalities. In [9], a somewhat similar procedure is described in detail.

We finally end up with a set of a constant number of polynomial inequalities expressing necessary and sufficient conditions for two spheres to conflict in the time intervall [0,1]. This is what we needed to apply the above range searching result: All spheres conflicting with a sphere b_i lie in the semialgebraic set which is defined by the above inequalities after substituting variables for the parameters x_j, r_j. Hence, after preprocessing the set of spheres in time $O(n \log n)$, these spheres can be reported in time $O(n^{\frac{4}{5}+\epsilon} + co_{np,i})$, where $co_{np,i}$ denotes the number of spheres in nonpermanent conflict with b_i. This is because the parameter b in theorem 2 is bounded by $b \leq 2 \cdot d - 3 = 5$ in our case. Iterating this query for all spheres in $\mathcal{B}$ increases the overall time by a linear factor. This leads to the following

Lemma 2 *After preprocessing the spheres in time $O(n \log n)$, all pairs of nonpermanently conflicting spheres can be determined in time $O(n^{\frac{9}{5}+\epsilon} + co_{np})$, where co_{np} denotes the number of nonpermanent conflicts.*

Since each nonpermanent conflict causes a transparent topology change, co_{np} is bounded by tc_{tr}. Hence, the running time of phase two can be rewritten as $O(n^{\frac{9}{5}+\epsilon} + tc_{tr})$.

4.4 Phase 3

In this phase we determine the set of all transparent topology changes, based on the information gained in the previous subsections. Topology changes of the types I and II only occur for pairs of spheres which are in conflict. Hence, we can simply run through the list of pairs of conflicting spheres and check in constant time per pair whether it causes a topology change of type I or II.

We still need to determine the topology changes of type III. For each sphere b_i which has at least one conflict we perform a "sweep" from $t = 0$ to $t = 1$. The sweepline corresponds to the circle $c_i(t)$ on $\mathcal{S}(t)$. It is implemented as a balanced binary tree and stores the intersections of $c_i(t)$ with the $c_j(t)$, where b_j conflicts with b_i, in clockwise order. The initial status of the sweepline can be determined by inspection of the initial transparent visibility graph G_0. The event schedule contains those points in time t at which either

- a point of intersection appears on or disappears from $c_i(t)$

- two points of intersection become coincident

The event queue is initialized with the events of the first type which can be determined by checking all conflicting spheres in constant time per pair. The sweep procedure works in a way reminiscent of the Bentley-Ottmann algorithm. Whenever an event point is reached, the sweep line is updated and the upcoming points of coincidence of all now neighboring pairs of intersections are inserted into the queue. Such coincidences correspond to topology changes of type III and are output during the sweep whenever they are reached. It is important to note that there can be only a constant number of such common intersections for each triple of spheres. To see this, consider the set of planes $p_i(t), 1 \leq i \leq n$ uniquely determined by $p_i(t) \cap \mathcal{S}(t) = c_i(t)$. For a given triple b_i, b_j, b_k of spheres, a common intersection of the projected circles $c_i(t), c_j(t), c_k(t)$ at time t occurs iff the planes $p_i(t), p_j(t), p_k(t)$ have a common point of intersection x which lies on $\mathcal{S}(t)$. The point of intersection x can be obtained by solving the system of linear equations

$$A(t) \cdot x = b(t)$$

induced by the plane equations of $p_i(t), p_j(t), p_k(t)$. By Cramer's rule, the coordinates of x are given by

$$x_i = \frac{\det A_i(t)}{\det A(t)} \quad . \; i = 1, 2, 3\,,$$

where $A_i(t)$ denotes the matrix obtained by replacing the i-th column of $A(t)$ by $b(t)$. The condition, that x lies on the surface of the projection sphere $\mathcal{S}(t)$ centered at $f(t)$ with radius r_S, can be expressed as

$$||x - f(t)||^2 = r_S^2$$

This is equivalent to

$$(\det A_1(t) - t \cdot \det A(t))^2 + \det^2 A_2(t) + \det^2 A_3(t) = r_S^2 \det^2 A(t)$$

This condition can be converted into a polynomial of degree 32 in t, which necessarily vanishes whenever the condition is fulfilled. Thus, for each tripel of spheres, we get a constant number of points in time, at which a transparent topology change of type III can occur. For a given sphere b_i, the corresponding sweep involving all co_i spheres conflicting with it, takes $O((co_i + tc_{tr,i}) \log(co_i + tc_{tr,i}))$ time, where $tc_{tr,i}$ is the number of transparent topology changes found during the sweep. Summing up, we get an overall running time of $O((co_{per} + tc_{tr}) \log n)$ for phase three. Finally, we sort all transparent topology changes, which carry the information by which spheres they are induced, by increasing t. This obviously takes time $O(tc_{tr} \log tc_{tr})$ which is $O(tc_{tr} \log n)$.

5 Conclusion

We gave an algorithm which maintains the visibility graph of a set of n nonintersecting spheres in 3-space while the viewpoint moves on a linear flightpath. The algorithm runs in time $O(n^{\frac{9}{5}+\epsilon} + (co_{per} + tc_{tr}) \log n)$, where co_{per} denotes the number of permanently conflicting pairs of spheres and tc_{tr} is the number of transparent topology changes. It uses recent results on range searching with semialgebraic sets to determine the number of conflicting pairs in subquadratic time with regard to n. It seems possible to further improve the $O(n^{\frac{9}{5}+\epsilon})$ term in the time bound by reducing the query time at the cost of additional space requirement, i.e., to obtain a time-space trade-off. This is currently investigated. A major open problem is to obtain an algorithm with a running time depending on the number of opaque and not the transparent topology changes.

Acknowledgement

The author thanks Elmar Schömer for helpful discussons on this subject.

References

[1] P. K. Agarwal, J. Matousek. On range searching with semialgebraic sets. *Discrete Comput. Geom.* 11 (1994), pp. 393-418.

[2] M. Bern, D. Dobkin, D. Eppstein, and R. Grossmann. Visibility with a moving point of view. *Algorithmica*, 11 (1994), pp. 360-378.

[3] R. Cole, M. Sharir. Visibility problems for polyhedral terrains. *J. Symbolic Comput.* 7 (1989), pp. 11-30.

[4] M. Katz, M. Overmars, M. Sharir. Efficient hidden surface removal for objects with small union size. *Comput. Geometry: Theory and Appl.* 2 (1992) pp. 223-234.

[5] H.P. Lenhof. *Distanz- und Suchprobleme in der algorithmischen Geometrie und Anwendungen in der Bioinformatik.* Ph.D. Thesis, University of Saarland (1993).

[6] H.P. Lenhof, M. Smid. Maintaining the visibility map of spheres while moving the viewpoint on a circle at infinity. *Algortihmica*, 13 (1995), pp. 301-312.

[7] K. Mulmuley. Hidden surface removal with respect to a moving point of view. *Proc. 23rd ACM Symp. on Theory of Computing* (1991), pp. 512-522.

[8] O. Nurmi. A fast line-sweep algorithm for hidden line elimination. *BIT*, 25 (1985), pp. 466-472.

[9] E. Schömer, Ch. Thiel. Efficint collision detection for moving polyhedra. *Proc. 11th ACM Symp. Comp. Geom.* (1995), pp. 51-60.