A NEARLY OUTPUT SENSITIVE PARALLEL HIDDEN SURFACE REMOVAL ALGORITHM IN OBJECT SPACE

Klaus Meyer
Department of Mathematics and Computer Science
University of Paderborn, Germany
e-mail: orpheus@uni-paderborn.de
http://www.uni-paderborn.de/cs/ag-monien.html

ABSTRACT

We present a new method for solving the hidden surface removal (HSR) problem in parallel on a CREW PRAM using a combination of new observations and known methods for solving related geometric problems. The new algorithm obtains the bounds of $O(\log^2 n)$ time and $O(n \log n + I)$ processors, where n is the amount of given endpoints of the input and I is the number of intersections in the viewing plane. In contrast to most known algorithms solving the HSR problem in object space, this method is able to process input scenes where penetrations of line segments and polygonal areas are allowed. The ability to process such an input has an enormous advantage for integrating three dimensional curves $c \subset \mathbb{R}^3$, which have been approximated by segments. Since a penetration of two polygonal areas will be detected during run time, this algorithm could also be used for testing given scenes for intersections of polygonal areas. In this case the algorithm is able to solve the hidden line removal problem in the same bounds. For practical purposes this algorithm allows also non convex (but simple and planar) polygons as input. You do not have to have triangulated scenes. According to the constraints of object space HSR algorithms, we will construct the visibility graph of the given scene as a planar graph in the viewing plane. Although the number of processors depends on the number of intersections in the viewing plane, in most given scenes this method will work like an output sensitive algorithm. Typical examples like "one big rectangle is covering all intersections" will be detected and these intersections are not computed.

Keywords: Parallel algorithm, parallel computational geometry, parallel and distributed graphics, computer graphics, hidden surface removal, hidden line removal.

1 Introduction

The **h**idden **s**urface **r**emoval (HSR) problem and the **h**idden **l**ine **r**emoval (HLR) problem are substantial problems for visualization of given three dimensional data. One of the goals of computer graphics is to compute high quality renderings of a given scene with a fast algorithm. Since the early sixties, there have been numerous developments of serial algorithms for solving the HSR and HLR problem. Nowadays new parallel computers exist and parallel algorithms become very important not only in theory. The known efficient parallel methods are still subject to restrictions on the input like "only functions $z = f(x, y)$ are allowed" (e.g. [RS88]).

In this paper, we will solve the HSR problem for a given set of points, line segments and simple, planar polygons in $\mathbb{R}^3$. All polygons have to be pairwise disjoint (except possibly at boundaries). If any of two polygons penetrate each other, we will detect this situation and can solve the HLR problem instead.

2 Geometric preliminaries

Let $\vec{a}, \vec{b} \in \mathbb{R}^k, k \in \{2, 3\}$ be two points. With $\overline{ab}$ we denote the straight line segment between the endpoints $\vec{a}$ and $\vec{b}$. A *polygon* is a sequence $p = (s_1, \ldots, s_k)$ of segments $s_1 = \overline{a_1 a_2}, s_2 = \overline{a_2 a_3}$, ..., $s_k = \overline{a_k a_1}$. A polygon is said to be *planar*, if there exists a plane $E(p)$ with $s_i \subset E(p)$ for all $s_i \in p$, and *simple*, if any segment s_i does not intersect any other segment of p except its predecessor or successor segment. Let p be a simple and planar polygon. The enclosed area of p will

be denoted by $A(p)$ and the interior $A_i(p)$ of p is defined as $A_i(p) := A(p) - \delta(p)$ with $\delta(p) := \bigcup_{s \in p} s$ (the border of p).
We assume for each given polygon p, that the interior of p lies to the left relative to the viewer when traversing the segments of p.

3 The problem

A *scene* $S = (O, \Phi_{\mathcal{V}}, \prec)$ consists of a set $O = \{O_1, \ldots, O_n\} \subset \mathbb{R}^3$ of valid objects, the viewer's coordinate system $\Phi_{\mathcal{V}}$ and a total order $\prec$, which allows comparing the *visibility* of two points. We call a scene S simple, if all interiors of the polygons are pairwise disjoint (thus a penetration of a single segment and the enclosed area of a polygon is allowed). An object O_i is valid, if O_i is a point, a line segment, or a polyhedron. A point $\vec{q}$ will be associated with the special segment $\overline{qq}$. With the viewer's coordinate system we associate the canonical coordinate system of the $\mathbb{R}^3$ and the viewer is looking from $(0, 0, \infty)^T$. The *viewing plane* is defined as the canonical coordinate system of the $\mathbb{R}^2$, which is embedded in $\Phi_{\mathcal{V}}$. For a definition of *visibility*, we need a projection $pr : \mathbb{R}^3 \to \mathbb{R}^2, \vec{x} \mapsto \vec{x}'$ from three dimensional space on the viewing plane. Since a central projection can be realized as a transformation in $\mathbb{R}^3$, we will use the normal parallel projection with $pr((x, y, z)^T) := (x, y)^T$. In this paper, we will use the term O_i or O_i^3 for a three dimensional object and $O_i^2 := pr(O_i)$ for the projected object.

4 The output and subproblems

As output, we want to construct a planar subdivision of the viewing plane, called the visibility graph G of the scene S. All faces of this graph are bounded by simple polygons or a simple infinite polygonal path. All visible parts of segments and points lie on the border of the polygonal regions and the interior of a face F of the subdivision contains either a visible part of a polygon p of the scene or nothing.
Since we only deal with points, segments and polygons, it is easy to see, that the visibility of a segment s or polygon p can only change, if there exists another segment s' or polygon p', which has an intersection in the viewing plane or in object space or s^2, p^2 is totally included by another polygon. Thus every HSR algorithm working in object space must solve the following problems:
1. Intersections in the viewing plane: Let $LS \subset \mathbb{R}^2$ be a set of n line segments in the plane. Compute all visible intersections between all pairs of segments. These are at most $O(n^2)$ intersections. (Please note that an output sensitive algorithm should only compute $O(k + f(n))$ intersections with k as the number of visible intersections and $f(n) \ll O(n^2)$).
2. Object space intersection: Let S be a given scene with n polygons and m segments. Compute all visible intersections in $\mathbb{R}^3$ between all pairs of objects. These are at most $O(n^3 + m^2 + mn)$ (see [FKN80]).
3. Inclusion: Let s be a visible part of a segment of a given scene. Determine, whether there exists a polygon p of the scene with $s^2 \subset A_i(p^2)$, which is visible at the right resp. left hand side of s (see figure 1 a). In some cases of inclusion, we must insert dummy edges for constructing the visibility graph (see figure 1 b).

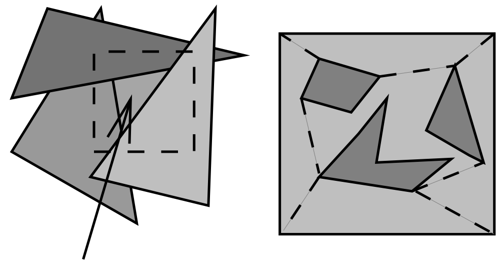

Figure 1: Inclusion: a) A "hole" created by three polygons, which is embedded in an otherwise invisible polygon. b) A partition of one polygon, which includes other polygons.

In this paper, we will solve the HSR problem for simple scenes as mentioned. For a simple scene with n segments, the maximal amount of edges in the visibility graph G is bounded by $O(n^2)$.
Please note for the rest of the paper that $O(f(n), g(n))$ in context with problem A means that there exists a CREW PRAM algorithm solving problem A within time $O(f(n))$ using $O(g(n))$ processors.

5 The basic idea

This section covers a brief idea of the algorithm. Let S be a given simple scene and LS^2 all projected line segments of S. Now draw from each endpoint of every segment a vertical line and compute all intersections of all such lines with all other segments in the viewing plane. Such a situation is illustrated in figure 6 on the left hand side for one line $L(q)$. Look at the unique plane $E(L)$ which is perpendicular to the viewing plane and contains L. We are now going back to 3D space. Since we know all intersections of the line L in the viewing plane with all segments we can easily compute the 3D vertical segments $s^3(p, L)$ (see figure 2) for each polygon which has an intersection with L in the viewing plane.

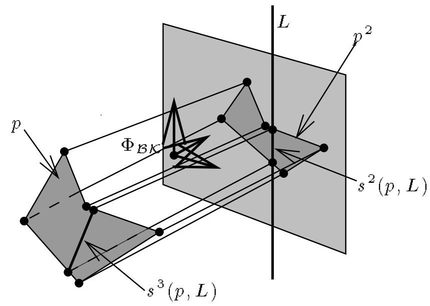

Figure 2: The 2D– and 3D–vertical segments of a polygon p and a vertical line L.

Every 3D vertical segment is lying in the plane $E(L)$. Now look at the plane as it is shown on the right hand side of figure 6 (just simply rotate $E(L)$ two times against 90 degree resp. x- and z-axes). The result is a complete cross section of the plane $E(L)$ and all given objects. We will now use the "Visibility from a Point algorithm" (please refer to [ACG89]) which determine all parts of the segments in $E(L)$ which are visible assuming every segment $s \in E(L)$ is opaque. The result gives us access to determine the *front polygon* exactly obscuring any given point $q \in E(L)$ in logarithmic time. Furthermore it allows us to determine also the polygons which lie directly behind the visible segments.

The vertical lines of the endpoints of every segment subdivide the whole viewing plane in vertical strips (see figure 3). Such a strip contains a set of segments which are subdivided by the strip itself. We will use the information of the VFAP computation of the boundaries (the lines L) of the strips and some intersection information of segments lying in such a strip to get all necessary information for building the visibility graph structure.

This basic approach leads immediatly to a serial complexity of $O(n^2 \log n)$. By using another data structure which utilizes a parallel approach of the plane sweep paradigma and with the help of some observations the HSR problem can be solved in the stated bounds in parallel.

6 Basic methods

The known parallel methods for solving geometric problems we will use, are described now. Let $LS \subset \mathbb{R}^2$ be a set of n segments.

Theorem 1: *Segment arrangements*
A segment arrangement is defined by all pairwise intersections of all segments and their endpoints as well as their vertical shadows. There exists an $O(\log n, n \log n + I)$ CREW PRAM algorithm to construct the segment arrangement as a planar graph with I the number of all pairwise intersections.

Proof: see [Goo91].

Theorem 2: *Segment intersection test*
One can decide in $O(\log n, n)$, whether there exists a pair of intersecting segments.

Proof: see [ACG89].

Theorem 3: *Planar point location*
Let us assume that LS consists of non intersecting segments except possibly at endpoints. Then there exists an $O(\log n, n)$ CREW–PRAM algorithm for computing a data structure, that, once computed, is able to answer in $O(\log n)$ time by using exactly one processor for a query point $q \in \mathbb{R}^2$, which segments $s_1, s_2 \in LS$ lie directly above resp. below q. If a planar subdivision of the plane is given, the face containing q is obtained in the same bounds.

Proof: see [ACG89].

Theorem 4: *Visibility from a point (VFAP)*
Let us assume that all segments in LS are are pairwise disjoint except possibly at endpoints. Let $q \in \mathbb{R}^2$ be a point, then the visibility from a point problem is to determine all parts of the segments of LS, which are visible from q, assuming every segment $s \in LS$ is opaque. There is a $O(\log n, n)$ parallel method for the CREW PRAM model to solve this problem.

Proof: see [ACG89].

Theorem 5: *Trapezodial decomposition of a simple polygon*
Let $p = (v_1, \ldots, v_n)$ be a simple polygon in the plane, where the v_i's denote the vertices of p. For any vertex v_i a *trapezodial edge* of v_i is an edge, which is directly above or below v_i such that the vertical line segment e from v_i to this edge is contained completely in $A(p)$. A vertex v_i can have up to two trapezodial edges. The trapezodial decomposition problem is to find all trapezodial edges for all vertices v_i and can be solved in $O(\log n, n)$ with a CREW PRAM.

Proof: see [ACG89].

7 The plane sweep tree (PST)

For solving the HSR problem we will use the well known parallel data structure of a plane sweep tree. With the help of this geometric structure, many other geometric problems have been efficiently solved in parallel (e.g. [AG86],[ACG89], [Rüb92], [Goo91]). We give a short review of some definitions and terms.
Definition: PST (see figure 3)
Let $LS = (s_1, \ldots, s_n) \subset \mathbb{R}^2$ be a set of line segments and $U = \{q_1, \ldots, q_m\}$ be an ordered set of

points on the x–axis. With the universe U we associate all endpoints of the segments $s \in LS$ projected on the x–axis. A plane sweep tree PST for LS with universe U is a balanced binary tree with $2m+1$ leaves. Each node $v \in PST$ contains an intervall information I_v, which is defined as the union of all intervalls of the subtree $T(v)$ rooted at v. The leaves of PST contain (from left to right) the intervalls $(-\infty, q_1), [q_1, q_1], (q_1, q_2), [q_2, q_2], \ldots, [q_m, q_m], (q_m, +\infty)$.
With $\Pi_v := I_v \times (-\infty, +\infty)$ as the vertical strip defined by the intervall information I_v, we will assign each node the following two segment sets $Cover(v)$ and $End(v)$:

$$\begin{aligned} Cover(v) &:= \{s \in LS \mid s \text{ spans } \Pi_v, \\ &\qquad \text{but not } \Pi_{parent(v)}.\} \\ End(v) &:= \{s \in LS \mid s \text{ has an endpoint in } \Pi_v, \\ &\qquad \text{but does not span } \Pi_v.\} \end{aligned}$$

We know from other research (e.g. [ACG+88], [Cha84]), that for $s \in LS$ there exist at most 2 nodes u, v on every level of the PST such that s is contained in $Cover(u), Cover(v)$ or $End(u)$, $End(v)$. The size of the PST is bounded by $O(n \log m)$.

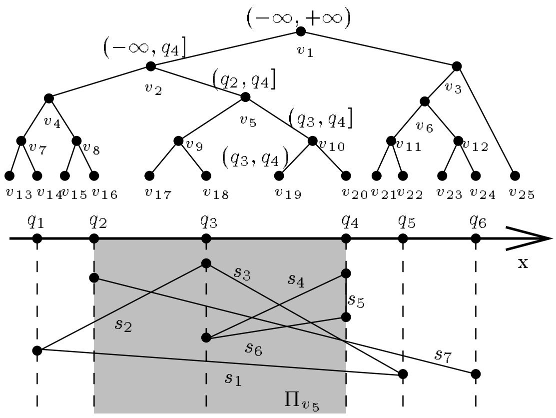

Figure 3: A PST for a given set of line segments consisting of two polygons and one segment.

The set

$$Ham(v) := \{s' \mid s \in Cover(v) \text{ and } s' = s \cap \Pi_v\}$$

represents all line fragments of the set $Cover(v)$. Ham is the abbreviation for *hammock*, since the structure of $Ham(v)$ looks like a hammock (see also [Cha84]).

8 New definitions and terms

For a detailed description of the new algorithm we will also need new terms. Let v be a node of a PST with the segment sets $Cover(v)$ and $End(v)$ and the vertical strip Π_v. In this paper, v_l, v_r denotes the left resp. right son of v. With

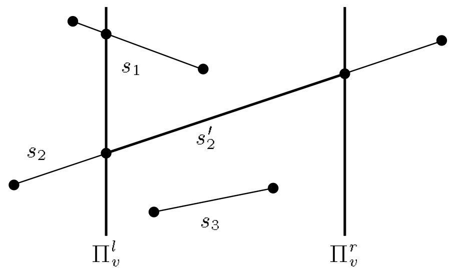

Figure 4: A strip Π_v with $s_1, s_3 \in End(v)$ and $s_2' \in Ham(v)$.

Π_v^l resp. Π_v^r we will denote the left resp. right vertical boundary of the strip Π_v (see figure 4):
1. $Ham^l(v) := \{\vec{p} \mid s \in Cover(v) : \vec{p} = s \cap \Pi_v^l\}$ and
$Ham^r(v) := \{\vec{p} \mid s \in Cover(v) : \vec{p} = s \cap \Pi_v^r\}$.
2. For a segment $s \in S$, the sequence $Cover(s) = (v_1, \ldots, v_k)$ resp. $End(s)$ consist of all nodes $v_i \in PST$ with $s \in Cover(v_i)$ resp. $s \in End(v_i)$. These sequences have at most $O(\log m)$ elements for every s (see e.g. [ACG89]). If one uses the $O(\log m, n \log m)$ method of [ACG+88] to construct the PST, these sequences can be ordered relative to x–coordinates during construction.
3. Let $q \in \mathbb{R}^3$ be a point and P a set consisting of simple and planar polygons in $\mathbb{R}^3$. In addition, we assume that somebody looks in the direction of this set. If there exits a polygon $p \in P$ with $q^2 \in A_i(p^2)$, p is called the *front polygon* of q relative to P, if and only if p is the polygon of P which is the first polygon from the viewers point obscuring q. p is called *back polygon* relative to P if and only if p is the polygon which lies directly behind q looking from the view point of the viewer.
4. For a subtree $T(v)$ with root v, the sets $LF(T(v))$ and $RF(T(v))$ consist of all nodes $u \in T(v)$ lying on the left resp. right fringe of $T(v)$. Then the border $\delta(T(v))$ of $T(v)$ is the union of $LF(T(v))$ and $RF(T(v))$. The root path denoted as $P(T(v), u)$ describes the set of all nodes lying on the shortest path from u to the root v.
Now we are going on to define some terms related to an endpoint $q \in U$ of a given segment:
5. $L(q)$ denotes the vertical line L with $q \in L$.
6. For each $q \in U$ $E(q)$ denotes the unique plane perpendicular to the viewing plane with $L(q) \subset E(q)$.
7. For each $q \in U$ there exists exactly one node $v \in PST$ with the property $\Pi_{v_l}^r = \Pi_{v_r}^l = L(q)$. For such a node v we will define the vertical line $g(v)$ as $L(q)$. For example look at q_4 in figure 3. Only v_1 has the property $\Pi_{v_2}^r = \Pi_{v_3}^l = L(q_4)$ and so $g(v_1) := L(q_4)$. See also figure 5.
8. $q_l(v), q_r(v) \in U$ are defined as the points in

U with $L(q_l(v)) = \Pi_v^l$ and $L(q_r(v)) = \Pi_v^r$. For v_5 e.g. holds (see figure 3) $q_l(v_5) = q_2$ and $q_r(v_5) = q_4$ since $L(q_2) = \Pi_{v_5}^l$ and $L(q_4) = \Pi_{v_5}^r$.

8.1 Cover sets, polygon sets and 3D segment sets

The definitions of the following sets are very important for the whole new algorithm: Let $q \in U$ be an endpoint of a segment and $v \in PST$ be the node with $g(v) = L(q)$. Furthermore, let (s, p, u, L) describe the situation that the segment $s \in LS$ is an element of $Cover(u)$ and intersects the line L. If s belongs to the border of the polygon t, then $p := t$ otherwise $p := NIL$. Now the following sets are defined:

a) $C(q) :=$

$$\bigcup_{u \in RF(T(v_l))} (s, p, u, \Pi_{v_l}^r) \cup \bigcup_{u \in LF(T(v_r))} (s, p, u, \Pi_{v_r}^l)$$

The set $C(q)$ contains all segments of the scene, which have a "cover intersection" with the vertical line $g(v)$ (see figure 5).

b) $P(q) := \{p \mid (s, p, u, L) \in C(q) \text{ and } p \neq NIL\}$
This is the set of all polygons which have a cover intersection with $g(v) = L(q)$.

c) $Q(q) :=$

$$\{s^3(p, L) \mid (s, p, u, L) \in C(q) \text{ and } p \neq NIL\} \cup \{a \in \mathbb{R}^3 \mid (s, p, u, L) \in C(q) : p = NIL \text{ and } a^2 = L \cap s^2\}$$

The set $Q(q)$ consists of 3D–vertical segments of polygons which have a cover intersection in the viewing plane with the line $g(v) = L(q)$ and all points a^3 of single segments s of the input scene S, which intersect $L(q)$ in the viewing plane in the point a^2. Please note that all elements of $Q(q)$ lie in the plane $E(q)$.

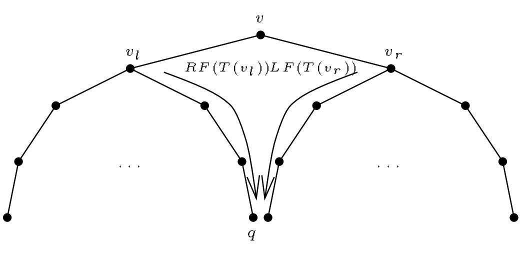

Figure 5: The set $C(q)$ is the union of all cover information in the left resp. right fringe of the right resp. left son of the node v with $g(v) = L(q)$.

Now we are finished with the definitions, so we can continue with the new observations which are essential for the new algorithm.

9 Observations

The new algorithm is based on a few observations, which are related to the structure of a plane sweep tree and the new terms and sets. Let S be a simple scene and PST the plane sweep tree for all n line segments contained in S^2 with universum U of size m.

Observation 1: For any polygon $p = (s_1, \ldots, s_l)$ of the scene S, we can compute all 2D– and 3D–vertical segments $s^2(p, \Pi_v^l)$, $s^2(p, \Pi_v^r)$, $s^3(p, \Pi_v^l)$ and $s^3(p, \Pi_v^r)$ for all nodes $v \in Cover(p)$ with $Cover(p)$ defined as $Cover(p) := \bigcup_{i=1}^{i=l} Cover(s_i)$. In particular, this can be done with a trapezodial decomposition (see [ACG89]) of the polygon p in $O(\log l \log m, l)$. If the node sets $Cover(s)$ are ordered relative to x–coordinates, one can use a similar technique to merge sort for each segment. Since $|Cover(s)| = O(\log m)$ this will reduce the time to $O(\log m)$ for every segment s_i. So we need for the construction of all 3D–vertical segments $s^3(p, \Pi_v^l)$ and $s^3(p, \Pi_v^r)$ in total only $O(\log m)$ time using k processors assuming that there are k segments of polygons in the scene S.

Observation 2: There exists a $O(\log n, n \log m)$ CREW PRAM algorithm constructing $C(q)$, $P(q)$ and $Q(q)$ for all q in parallel.

Proof: First note that $|C(q)| = O(n)$ since at most all segments of the scene may have an intersection with the vertical line $L(q)$ in the viewing plane. Furthermore the number of cover intersections is bounded by $O(n \log m)$ at all therefore $\sum_{q \in U} |C(q)| = O(n \log m)$.
The construction of $C(q)$ can be done in $O(\log m, n \log m)$ by collecting all cover sets $Cover(v)$, $v \in PST$ in one array A, sorting A relative to their left resp. right boundary $q_l(v), q_r(v)$. Then use a prefix computation and put all segments with the same left resp. right boundary into arrays A_q associated with each q. This can be done with the parallel merge sort of Cole (see [Col88]) in $O(\log(n \log m), n \log m) = O(\log n,\ n \log m)$. The construction of $P(q)$ can be done similarly by sorting the tupels $(s, p, u, L) \in C(q)$ relative to p and deleting all tupels with $p = NIL$ via a prefix computation for all $q \in U$ in parallel. Using observation 1 the construction of the sets $Q(q)$ for all $q \in U$ is possible in $O(\log n, n \log m)$ using the same strategy as for building $C(q)$ to store all computed 3D–vertical segments $s^3(p, L(q))$, $p \in P(q)$ in $Q(q)$. The points of $Q(q)$ are computed directly for each s with $(s, p, u, L) \in C(q)$ and $p = NIL$.

Observation 3: Let $c^3 \in Q(q)$, so c^3 is a single point or is contained by a 3D–vertical segment. With the help of the VFAP method running in the plane $E(q)$ with the input of all 3D–vertical segments $s^3(p, L(q)) \in Q(q)$ we are able to decide, whether there exists a front polygon $p \in P(q)$ for the point c^3. This is possible due to the contraint that all polygonal areas have to be pairwise disjoint. So all 3D–vertical segments in $Q(q)$ may not intersect except possibly at endpoints.

Observation 4: (s. figure 6) There exists nodes $v \in PST$, for which the sets $Q(q_l(v))$ and $Q(q_r(v))$ are complete cross-sections of all surfaces $A(S)$ of the scene S with the two planes $E(q_l(v))$ and $E(q_r(v))$. These are exactly the nodes $v \in PST$ where $Cover(v) \neq \emptyset$ holds the first time on every possible path from the root of the PST to any leaf of the PST.

Proof: obvious.

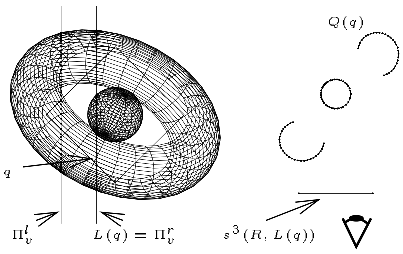

Figure 6: A scene S with an open torus, a sphere and a rectangle R obscuring the sphere and parts of the torus. On the right hand side you see the set $Q(q_r(v))$ for a node v as stated in observation 4.

Observation 5: This observation will give a hint for computing front and back polygons relative to the whole given scene for points $c^3 \in Q(q_l(v))$ or $c^3 \in Q(q_r(v))$.
Let $q \in U$ and $s^3 = s^3(p, L(q)) = \overline{ab}$ be a 3D–vertical segment of the line $L(q)$ and a polygon p of the input scene. Furthermore let s_1, s_2 denote the two segments on the border of p which contain the endpoints a, b of $s^3(p, L(q))$ and $v \in PST$ the node with the property $g(v) = L(q)$. In addition let $c^3 \in Q(q)$ with $c^2 \cap s^2 \neq \emptyset$, so the point c^3 is lying behind, on or in front of s^3. Then either holds $s \in Q(q)$ or for each segment s_1, s_2 exists a node $u_1, u_2 \in P(PST, v)$ with $s_1 \in Cover(u_1)$ and $s_2 \in Cover(u_2)$.
If the area of a polygon p obscures a point $c^3 \in Q(q)$ then either holds $s^3(p, L(q)) \subset Q(q)$ or the cover intersections of this area are stored in cover sets of nodes lying on the path from v to the root of the PST (v included).

Proof: omitted.

Observation 6: We will now show, how the problem of inclusion can be solved for a segment fragment $s \in Ham(v)$.
Let p be a polygon of the scene S and $s \in Ham(v)$ a segment fragment with $s \subset A_i(p^2)$, thus $s \cap \delta(p^2) = \emptyset$ (s is totally included by p). Then either holds $s^3(p, \Pi_v^l) \subset Q(q_l(v))$ **and** $s^3(p, \Pi_v^r) \subset Q(q_r(v))$ or for all segments t of p with $t^2 \cap \Pi_v \neq \emptyset$ exists a node $u \in P(PST, v) - v$ with $t \in Cover(u)$.
If we are searching such a polygon, we must look in the set of polygons, which have cover nodes in $RF(T(v))$ and $LF(T(v))$, thus $p \in P(q_l(v)) \cap P(q_r(v))$, or try to find it above v (v excluded).

Proof: omitted.

Observation 7: Let $Ham(v)$ be a hammock of the node $v \in PST$. If all intersections of the segment fragments in $Ham(v)$ with all segments of the input scene are known and for each point $c \in Ham^l(v), c \in Ham^r(v)$ the possible front– and back polygon relative to the input scene has been computed then the visibility relation for each segment fragment is computable in a total independent way, that means in parallel.

Proof: see section 11

10 An overview of the algorithm

The algorithm is running in two phases. Let S be a scene consisting of points, segments and polygons. Let LS^3 be all line segments contained in S and $n := |LS^3|$.

First phase: In the first step, we build the PST of LS^2 with universum U, $m := |U|$ and compute all sets $Cover(v)$, $End(v)$ for each node $v \in PST$ using the $O(\log m, n \log m)$ method of [ACG+88]. We assume for all segments $s \in LS^2$ that the node sets $Cover(s)$ and $End(s)$ are given ordered relative to x–coordinates. This can be easily achieved by using the same method as for building the PST. For each polygon p we will need a trapezodial decompostion $TD(p)$. This can be done in $O(\log n, n)$. Next we determine in $O(\log n, n \log m)$ for each $q \in U$ the sets $C(q)$, $P(q)$, and $Q(q)$. We now test in parallel for all sets $Q(q)$, if any pair of the $s^3(p, L(q)) \in Q(q)$ segments intersects (we make a difference between valid intersections at the endpoints of two segments and other intersections) in the plane $E(q)$ in $O(\log n, n \log m)$ using theorem 2. If there exists such an intersection, we have found a penetration of two polygons in object space. Let us assume, there is no penetration. Then we solve the VFAP problem with input $Q(q)$ in the plane $E(q)$ by using the given method of theorem 4 for each set $Q(q), q \in U$ in parallel. That means the y– resp. z–coordinate of each endpoint from a segment $s^3(p, L(q)) \in Q(q)$ and each other point $a \in Q(q)$ will be used as the x– resp. y–coordinate in $\mathbb{R}^2$, where the observer is at negative infinity below all segments. This can still be done in $O(\log n, n \log m)$. We transform the result back to $Q(q)$ and now know for each (end)point $a \in Q(q)$ whether there exists a front polygon $p \in P(q)$ relative to $P(q)$ which is obscuring a. Furthermore, we now know whether

there exists another object obscuring a like a single segment or a single point of the input, which has an intersection with $L(q)$ and belongs to the set $Q(q)$.
Now we are finished with the first phase, which can be done in $O(\log n, n \log m)$.

The main phase: We are now ready to process the PST level by level from the root to the leafs in $O(\log m)$ steps. In each level L_i we will execute the following steps in parallel for each node $v \in L_i$:

1. First we determine all intersections between the segment fragments in $Ham(v)$ and all segments of LS^2 using the method of *Goodrich* (s. [Goo91]). Note that all segments $s, s \in End(v)$ know exactly, whether they intersect a segment fragment in $Ham(v)$.

2. In each hammock $ham(v), v \in L_i$ now all visible parts of the segment fragments and their possibly existing back polygones relative to the input scene will be computed. In the next section we will show, how to compute all visible parts of the segment fragments in $Ham(v)$. In addition we will solve the problem of inclusion determining for each visible segment part s' the possibly existing polygons of the scene with a visible portion to the left and right side of s', if they exist.

3. Next we construct for all visible parts of segments computed so far a planar subdivision G like the solution of the segment arrangement problem including vertical shadows. Note that all determined visible parts of segments do not intersect.

4. According to observation 5, we will update the front– or back polygons for all points in $C(q)$ in parallel for each $q \in \{ r \in U \mid \exists\, v \in L_i : g(v) = L(r) \}$ by searching the given planar subdivision. Thus the information of $Ham^r(u)$ and $Ham^l(u')$ for each $u \in RF(T(v_l))$ and each $u' \in LF(T(v_r))$ will be correct relative to the whole given scene, not only to $Q(q_r(u))$ resp. $Q(q_l(u'))$.

If the algorithm stops, the constructed planar subdivision is the visibility graph. The inserted dummy edges of inclusions are given by the vertical shadows which have been inserted during step 3.

11 Computing all visible segment parts and their back polygons in $Ham(v)$

Let us assume that the algorithm has finished phase 1 and is now running in Level $L_i, 0 \leq i \leq O(\log m)$. Let $v \in L_i$ be a node with $Cover(v) \neq \emptyset$ and $r^2 = \overline{ab} \subset s^2, s \in S$ be a segment fragment of $Ham(v)$. Then we got from the previously executed steps the following information:

We already know all intersections with all projected segments of the scene because of the use of the intersection algorithm of *Goodrich*. We also know the front polygons relative to the whole input set for both points a^3 and b^3, if they exist. The front polygons relative to $P(q)$ with $L(q) = g(v)$ will be computed in the first phase. Observation 5 tells us that other candidates for front polygons have been processed already in prior levels and will be detected during the update step 4 in the main phase of the algorithm for the next level. This is easy to prove via induction for each level by using observation 4, observation 5, and the update step 4 in the main phase of the algorithm. Please note that during the update process also candidates for possible back polygons of the points a^3, b^3 will be found, whose segments are stored in cover sets above v.

Part A: Computation of all visible parts of a segment fragment

Let $r^2 = \overline{ab}$ be a segment fragment of $Ham(v)$. If there is no intersection of r^2 with any other segment s of the scene, then the visibility of r depends only on the front polygons of the both endpoints a, b. If there exists any front polygon for a or b, the complete segment fragment is invisible. So let $((t_1, s_1), \ldots, (t_i, s_i)), i \geq 1$ be the sequence of the intersection parameters of r with segments s_j^2 at the points t_j. The use of *Goodrich's* algorithm for finding intersections assures that this sequence is orderd relative to the parameter values. If there is any polygon p^2 adjacent to a segment s_j, then r^2 will either enter or leave the area of p^2 or will have an intersection with a corner of $\delta(p^2)$. We can decide this by using a simple cross product test, since all segments of p^2 are ordered. The interior lies always to the left relative to the viewer traversing the border segments of p^2. Furthermore we can compute a penetration point of r^3 and p^3. A penetration point c^3 is valid, iff $c^3 \in r^3$. If r^3 is part of a segment of an input polygon p', a forbidden penetration of two input polygons is detected. A penetration of a single segment with a polygonal area is allowed.
Let $P(r)$ be the set of all intersected polygons by r, united with the two possibly existing front polygons of the endpoints a, b of r. After the computation of all possible penetration points of r^3 with $A(p), p \in P(r)$ we simply add this parameter values to the given sequence of the intersection parameters of r and sort this sequence again according to the parameter values. This can be done in $O(\log i, i)$, since there exist at most $O(i)$ penetration points.
Since we know exactly which part of r^2 lies in an area of a polygon p^2 we now go back to 3D space using the same VFAP method as before.

Let $L(r^2)$ be the line which contains r^2 and $E(r^2)$ the perpendicular plane containing $L(r^2)$. With the help of the intersection information compute the 3D segment $s^3(p, L(r^2))$ relative to line $L(r^2)$ for each polygon $p \in P(r)$ restriced to the lenght of r (see figure 7).
The result of this method is a set M. In addition, we have to add to the set M all intersections of single segments $s_j \cap E(r) \subset \mathbb{R}^3$, which are members of the intersection sequence of r and the segment fragment r^3 itself.
All elements of M are lying in the plane $E(r)$ (see figure 7). For this input one can use the VFAP method to compute all visible parts of r^2 (with some caution in respect to the allowed input set of the VFAP algorithm) in $O(\log i, i)$, since $|M| = O(i)$.

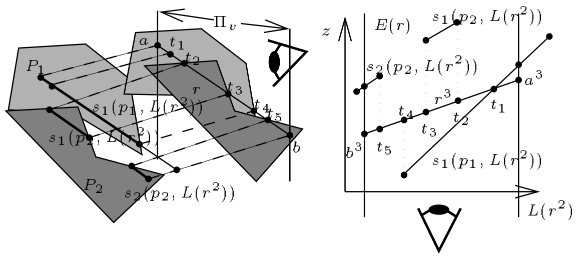

Figure 7: Left: the 3D segments $s(P_1, L(r))$ and $s(P_2, L(r))$. Right: The set M in the plane $E(r)$ which contains $L(r^2)$ and is perpendicular to the viewing plane. t_1 is a penetration point.

Thus the visible parts of a segment fragment $r \in Ham(v)$ are computable in $O(\log i, i)$ where i is the amount of intersections of the segment fragment r with all other segments in the given scene S.

Part B: Computation of back polygons for visible parts of a segment fragment

With the computation of the back polygons for visible parts of a segment fragment the problems of inclusion and holes can be solved (see figure 1). Let s be a visible part of the segment fragment $r \in Ham(v)$. Let us assume that there exists a back polygon $p \in S$ for s. Then at most the following cases are given:

1. $A(p^2)$ has intersections with r^2 in Π_v.
2. r^2 has the property $r^2 \subset A_i(p^2)$, thus $r^2 \cap \delta(p^2) = \emptyset$.

Case 1 can be solved similarly to Part A. Just take a look at the set $P(r)$ and the set M. It is rather simple to delete all elements of M, which are in front of the segment fragment r, with the help of simple cross product tests and then remove the segment fragment r itself from M to obtain the set M'.
In figure 7 the set M' would consist of the elements $s_1(p_2, L(r^2))$, $s_2(p_2, L(r^2))$ and s_3, where s_3 is the part of the segment $s(p_1, L(r^2))$ which lies behind r (above in $E(r)$). Now use the VFAP method for the input M' and make a binary search for each intersection parameter of r^3 (in $E(r)$!!) by using the answer of the VFAP method. Then you get all back polygon information for each intersection of r^2 with other segments and thus for the visible part s. This can be done in $O(\log i, i)$ as well.

Case 2, however, is more intricate (see figure 8).

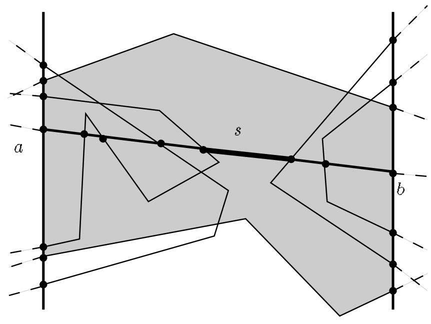

Figure 8: The shaded area is part of a possible back polygon for s, which is not determinable by intersections.

A parallel solution to this problem is given by building a datastructure D for the whole hammock with the help of observation 6. Then this datastructure can be asked from each intersection of all segment fragments in $Ham(v)$ independently for a back polygon of the kind case 2 in logarithmic time. Observation 6 tells us that such a back polygon lies either completely above v or is contained in the set $P(q_l(v)) \cap P(q_r(v))$. The possible back polygons p for a visible part s of a segment fragment, which are stored completely above v, will be found during the update step but only for the endpoints a^3, b^3 of r. If r^2 leaves or enters such a polygon p^2 it has an intersection with the border of p^2. This case is already solved in case 1. If not, such a p is a candidate for the correct back polygon. For the determination of all other candidates with $r^2 \subset A_i(p^2)$ we firstly have to build $P(v) := P(q_l(v)) \cap P(q_r(v))$. This can be done by sorting in $O(\log n, n)$, since $|P(q_l(v))| = O(n) = |P(q_r(v))|$ and thus $|P(v)| = O(n)$. For a polygon $p \in P(v)$ always holds $s^3(p, \Pi_v^l) \subset Q(q_l(v))$ and $s^3(p, \Pi_v^r) \subset Q(q_r(v))$. Take a look at the set $Q_l(v) := \{s^3(p, \Pi_v^l) \mid p \in P(v)\}$. This set consists of all 3D–vertical segments of the polygons $p \in P(v)$ with the line Π_v^l. Since all polygons have to be pairwise disjoint, this set contains only segments which possibly intersect at most at endpoints. Furthermore, all of these segments are lying in the same plane $E(q_l(v))$. For this set we now build a datastructure $D(v)$ in the plane $E(q_l(v))$ which is able to answer the following question in logarithmic time: Let $c \in E(q_l(v))$ be a point of the plane $E(q_l(v))$. Which segments of $Q_l(v)$ are lying directly above and below the point c ? This datastructure $D(v)$ can be built in $O(\log n, n)$ by using theorem 3.

With the help of the datastructure $D(v)$ all other candidates for back polygons can be determined entirely independent using the following algorithm: Let (t_j, s_j) be an intersection point in the viewing plane of $r^2 = \overline{ab}$ with a segment s_j^2 which is adjacent to a polygon p. Now project the endpoint a^3 of r onto the plane $E(p)$, which contains the polygon p. So we got a point $(a,p) \in \mathbb{R}^3$ with the property $(a,p) \in E(q_l(v))$. Now construct the set $A_r := a^3 \cup$

$\{(a,p) \mid \delta(p^2) \cap r^2 \neq \emptyset$ or r^3 penetrates $A(p)$ $\}$

The set A_r has the property $|A_r| = O(i)$. For every point (a,p) we are now able to compute the segment $below((a,p), D(v))$ with the help of $D(v)$. This segment does not exist or is the 3D–vertical segment of a polygon $p' \in P(v)$. Thus we have constructed a polygon set $P'(v,r) \subset P(v)$ for each segment fragment $r \in Ham(v)$ with the following property: $r^2 \cap A(p^2) \neq \emptyset$ for all $p \in P'(v,r)$, because $a^2 \cap s^2(p, \Pi_v^l) \neq \emptyset$. Now mark all polygons $p \in P'(v,r)$ which have an intersection on the viewing plane with the fragment r (we know all these intersections !) and delete them with the help of sorting from $P'(v,r)$ to build the set $P''(v,r)$. The set $P''(v,r)$ has now the property $r^2 - a, b \subset (A_i(p^2))$ for all $p \in P''(v,r)$. The endpoints of r may have an intersection with the border of such a p^2, but this is not a contradiction for a candidate of a back polygon. The only thing we have to do is to sort the constructed set $P''(v,r)$ relative to depth. The closest polygon to the viewer is the candidate for a correct back polygon for all visible parts of r.

Finding such a candidate for a back polygon for all visible parts of a segment fragment r can be done for each segment fragment $r \in Ham(v)$ in $O(\log i_r, i_r)$ with a preprocessing of $O(\log n, n)$ for the hammock $Ham(v)$ and i_r as the number of intersections in the viewing plane of r with all other segments of the scene and all penetration points.

All candidates of back polygones (back polygones from the update, from the intersections, and the polygon from P"(v,r)) are now tested against depth for each intersection point and if necessary, assigned to the visible parts of r as their back polygon relative to the whole input scene.

Summary: In this part of the paper we have shown how to compute in parallel all visible parts of a segment fragment $r \in Ham(v)$ and their possibly existing back polygons. Finishing this section we want to give the time and processor complexity for processing a whole level of the PST in parallel.
This leads to the following theorem:

Theorem 6 Let L_i be the ith Level of the PST being processed. Then all visible parts of every segment fragment $r \in Ham(v)$ for all $v \in L_i$ with $Cover(v) \neq \emptyset$ and their back polygons can be determined in $O(\log n, n \log m + I)$ with I as the number of all intersections and penetrations of the fragments with other segments and polygons of the input scene.

Proof: Except from the construction of $P(v)$ and $D(v)$ all operations will be performed on sets proportional to the amount of intersections a segment fragment can have, which is at most $O(n)$. Thus the time for all operations is clearly bounded by $O(\log n)$ for building $P(v)$ and $D(v)$ and all the operations for computing the visible parts of each segment fragment. The number of processors used is bounded by the number of intersections I and the maximum number of elements for the construction of $P(v)$ and $D(v)$ for each $v \in L_i$. Since the number of elements of $D(v)$ is bounded by the amount of elements in $P(v) = P(q_l(v)) \cap P(q_r(v))$ at most

$$\sum_{v \in L_i} |P(q_l(v)| + |P(q_r(v)| \leq 2 * \sum_{q \in U} |P(q)| = O(n \log m)$$

processors will be needed for the construction of $P(v)$ and $D(v)$ for the whole level L_i.

12 Complexity of the algorithm

Theorem 7 *Complexity of the HSR problem in object space*
Let S be a simple scene consisting of n segments (a single point c will be counted as a special segment $\overline{cc}$) in total. Let I be the amount of all intersections between segments in the viewing plane and all penetrations between single segments and polygonal areas. Then there exists a $O(\log^2 n, n \log n + I)$ CREW PRAM algorithm which solves the HSR problem in object space by constructing the visibility graph in the viewing plane.

Proof:
The overall complexity of phase 1 has been proven already during the description as $O(\log n, n \log m)$. Please note that the the universum U contains at most $O(n)$ points and I is bounded by $O(n^2)$. Now let us take a closer look at the steps 1 to 4 of the main phase. Since we are using the method of *Goodrich*, see [Goo91], the number of computations of intersections for the whole PST

are bounded by $O(\log n, n \log n + I)$, thus also for each level L_i in step 1. Theorem 6 reports the same amount for computing all visible parts of segment fragments and their back polygons for a whole level (step 2). The construction of the planar subdivision G in step 3 is bounded by the number k of all visible parts, which have been already processed. Thus this can be done using theorem 3 in $O(\log k, k)$. Always holds $k \leq (I + n)$, because there will not ever be more visible parts of segment fragments as the sum of intersections and segments. So this step is bounded by $O(\log(n + I), n + I) = O(\log n, n + I)$ for each level. For the last step at most all $O(n \log m)$ elements of the sets $C(q)$ have to be updated. All queries can be done in parallel in time $O(\log(n + I)) = O(\log n)$ with $O(n \log m)$ processors.

Thus the complexity $O(\log n, n \log n + I)$ of the steps 1 and 2 is dominating all other complexities. Since the main phase will run at most $O(\log m)$ times, the overall complexity of the algorithm is given by:

$$O(\log n \log m, n \log m + I) = O(\log^2 n, n \log n + I)$$

13 Avoiding intersections

Since the algorithm still computes all intersections, we will show now, how to avoid unnecessary computations of intersections between segments. Let v be a node in level L_i and $s = \overline{ab} \in Ham(v)$ be a segment fragment. Then we mark s as unvisible, if a and b have the same front polygon p with $p \notin P(v)$ and there is no intersection with any segment of $\delta(p)$. The intersections of (visible) polygons covered by nodes u above v and s are known, because $s \in End(t)$ holds for all $t \in P(PST, v) - v$). All CC– and EC–intersections (s. [Goo91]) of such a segment s do not have to be computed. This helps to give the algorithm an output sensitive touch.

14 Concluding remarks

We presented a new CREW PRAM method solving the HSR problem for complex scenes. The best known algorithm (see [MG90]) so far dealing only with pairwise disjoint polygons uses a running time of $O(\log n)$ and $O((n + I) \log n)$ processors and this algorithm is not able to handle penetrations of single line segments with polygonal areas. Here the penetration of single segments with polygonal areas is allowed. So it is possible to integrate any curve $c \subset \mathbb{R}^3$ approximated by segments into the scene without any problems. This is the most efficient parallel method the author knows about, being able to manage such an input. If inclusions are detected, the intersections contained in the interior of the covering polygon are not computed, thus many unnecessary computations are avoided. This method will be realized on an available parallel computer.

References

[ACG+88] A. Aggarwal, B. Chazelle, L. Guibas, C. Ó'Dúnlaing, and C. Yap. Parallel computational geometry. *Algorithmica*, 3:293–327, 1988.

[ACG89] M. Atallah, R. Cole, and M. Goodrich. Cascading divide and conquer: A technique for designing parallel algorithms. *SIAM J. Comput*, 18(3):499–532, 1989.

[AG86] M.J. Atallah and M.T. Goodrich. Efficient plane sweeping in parallel. Technical Report CSD-TR-563, Purdue University, 1986. s.a. ACM Symp. on Computational Geometry 1986, pp. 216-225.

[Cha84] B. Chazelle. Intersecting is easier than sorting. In *Proc 16th Symp. Theory of Comp.*, pages 125–134, 1984.

[Col88] R. Cole. Parallel merge sort. *SIAM J. Comput.*, 17:770–785, 1988.

[FKN80] H. Fuchs, Z.M. Kedem, and B.F. Naylor. On visible surface generation by a priori tree structures. *Computer Graphics*, 14:124–133, 1980.

[Goo91] M.T. Goodrich. Intersecting line segments in parallel with an output sensitive number of processors. *Siam J. Comput.*, 20(4):737–755, August 1991.

[MG90] J. Bright M.T. Goodrich, M.R. Ghouse. Generalized sweep methods for parallel computational geometry. In *2nd ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 280–298, 1990.

[Rüb92] C. Rüb. Line-segment intersection reporting in parallel. *Algorithmica*, 8:119–144, 1992.

[RS88] J.H. Reif and S. Sen. An efficient output-sensitive hidden-surface removal algorithm and its parallelization. In *Proc. of the fourth annual Symp. on Computational Geometry*, pages 193–200, 1988.