On-line randomized construction of the upper envelope of triangles and surface patches in R3

Jean-Daniel Boissonnat, Katrin Dobrindt

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

On-line randomized construction of the upper envelope of triangles and surface patches in $\mathbb{R}^3$

Jean-Daniel BOISSONNAT
Katrin DOBRINDT

N° 1878
Avril 1993

PROGRAMME 4

Robotique, Image
et
Vision

On-line Randomized Construction of the Upper Envelope of Triangles and Surface Patches in $\mathbb{R}^3$

Construction randomisée en-ligne de l'enveloppe supérieure de triangles et carreaux de surface dans $\mathbb{R}^3$ *

Jean- Daniel Boissonnat[†] **Katrin Dobrindt**[†]

Programme 4 : Robotique, Image et Vision.

Keywords: Computational Geometry, On-line Algorithms, Randomized Incremental Algorithms, Upper Envelope, Hidden Surface Removal.

[†]Institut National de Recherche en Informatique et Automatique
— B.P.93 — 06902 Sophia-Antipolis cedex (France)
Telephone : +33 93 65 77 62 — Fax : 93 65 76 43
e-mail : dobrindt@sophia.inria.fr

* *This work was partially supported by the ESPRIT Basic Research Actions Nr. 7141 (ALCOM II) and Nr. 6546 (PROMotion).*

Abstract

In this paper we describe an on-line randomized algorithm for computing the upper envelope (i.e. pointwise maximum) of a set of n triangles in three dimensions. The main new feature of this algorithm is the combination of two layers of influence graphs, which were introduced in [1]. We can insert the n-th triangle in $O(\log n \sum_{r=1}^{n} \frac{f^0(\lfloor r/2 \rfloor)}{r^2})$ expected time, where $f^0(r)$ is the expected size of an intermediate result for r triangles. Since $f^0(r) = O(r^2 \alpha(r))$, the expected time for the insertion of the last triangle is bounded in the worst case by $O(n\alpha(n)\log n)$. This algorithm is easy to implement and works also nicely for surfaces and surface patches of fixed maximum degree.

Résumé

Dans cet article nous décrivons un algorithme randomisé en-ligne pour calculer l'enveloppe supérieure (i.e. le maximum, point par point) d'un ensemble de n triangles dans l'espace. La caractéristique principale de cet algorithme est la combinaison de deux couches de graphes d'influence introduits dans [1]. On peut insérer le n-ième triangle en temps moyen $O(\log n \sum_{r=1}^{n} \frac{f^0(\lfloor r/2 \rfloor)}{r^2})$ où $f^0(r)$ est la taille moyenne d'un résultat intermédiaire pour r triangles qui est plus petite que $O(r^2\alpha(r))$. Le cout moyen de la dernière étape est donc borné dans le pire des cas par $O(n\alpha(n)\log n)$. Cet algorithme est simple à implémenter et s'étend aux surfaces et carreaux de surface de degré borné.

1 Introduction

In this paper we consider the following problem. Let $\mathcal{S}$ be a set of n triangles in three dimensions. We wish to compute the upper envelope (i.e. pointwise maximum) of the triangles in $\mathcal{S}$. It has been shown in [11] that the combinatorial complexity of such an envelope is $\Theta(n^2\alpha(n))$, where α is the functional inverse of the Ackermann function. This problem has several interesting applications including the hidden surface removal problem where we want to determine the portions of objects in space visible from some viewpoint (see [15] for an early history). This problem is a special case of the envelope problem, since the objects do not intersect.

The problem of constructing the upper envelope of a set of n triangles in three dimensions has been studied in [6], where a worst case optimal deterministic algorithm was presented that takes time $O(n^2\alpha(n))$. However, since the algorithm includes the computation of an arrangement of $3n$ lines in the plane, its lower bound is $\Omega(n^2)$. Furthermore, it does not extend to surfaces or surface patches. An incremental randomized but static algorithm for hidden surface removal of non-intersecting faces in three dimensions whose expected running time is $O(n\log^2 n + \theta(1)\log n)$ was given in [9]. $\theta(1)$ is equal to $\sum_l \frac{\text{number of junctions at level } l}{l}$, where the *level* of a junction is the number of faces which make it invisible, and is $O(n^2)$ in the worst case. In [10] a running time that depends linearly on $\theta(1)$ is achieved. In [12] upper bounds on the size of the upper envelope for special cases of surfaces were obtained and deterministic algorithms for their calculation by a factor $\log n$ worse than the bounds were presented.

An attractive approach in computational geometry to solve difficult problems is to use simple randomized incremental algorithms. Their complexities are not worst case optimal, but only expected when averaging over all permutations of the input. In [1] a general data structure, the *influence graph*, for the design of on-line geometric algorithms was developed. Since the algorithms are on-line, there is no prior information needed about the as yet uninserted input. In the following we present an on-line randomized algorithm for constructing the upper envelope of a set of n triangles in three dimensions. The main new feature of our algorithm is the combination of two layers of influence graphs. Its expected update time for the n-th triangle is $O(\log n \sum_{r=1}^{n} \frac{f^0(\lfloor r/2 \rfloor)}{r^2})$, where $f^0(r)$ is the expected size of an intermediate result for r triangles. Thus, with the complexity result mentionned above it is bounded by $O(n\alpha(n)\log n)$ in the worst case. However, its behavior will be better as soon as $f^0(r) = o(r^2\alpha(r))$, which is the case in most practical situations. This algorithm is simple and it can be used for surfaces and surface patches of fixed maximum degree with the same time complexity depending on the complexity $f^0(r)$ of an upper envelope of r surface or surface patches. More precisely, if $f^0(r)$ is non-linear, our general algorithm has the same time complexity as the specialized algorithms of [12]. Furthermore, it can be used to compute the upper envelope of those surface patches considered in [7] in expected time $O(n\sqrt{\lambda_q(n)}\log n)$, where q is a constant depending on the degree of the surfaces and $\lambda_q(n)$ the maximum length of a (n, q) Davenport-Schinzel sequence.

The paper is organized as follows. In Section 2 we recall some definitions and results

needed in the following. The incremental algorithm for adding a triangle is explained in Section 3. Section 4 gives the analysis of insertions. The extension of the insertion algorithm to surface patches is presented in Section 5. We conclude this paper with a discussion of our results.

2 Preliminaries

This section provides the definition and well known results for the upper envelope. It also recalls the algorithm of [3] and [8] for constructing the arrangement of line segments in the plane which is a basic tool for our algorithm.

Definition 2.1 Let $S_1, S_2, \ldots, S_n$ be n d-simplices in $\mathbb{R}^{d+1}$. Each d-simplex S_i can be regarded as the graph of a partially defined linear function $x_{d+1} = f_i(x_1, x_2, \ldots, x_d)$. The *upper envelope* M of the d-simplices is the pointwise maximum of these functions, i. e. $M(x_1, x_2, \ldots, x_d) = max_{1 \leq i \leq n} f_i(x_1, x_2, \ldots, x_d)$.

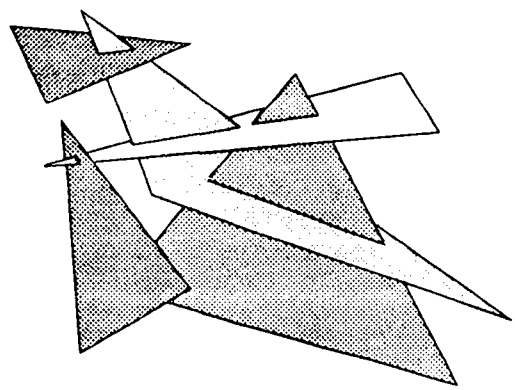

Figure 1: The upper envelope of five triangles

An example of an upper envelope of a set of 2-simplices, which are triangles in $\mathbb{R}^3$, is shown in Figure 1. We suppose that the triangles are non-vertical and in general position, i. e. no two of them overlap one another, no vertex of them lies on another triangle, no two edges of distinct triangles intersect, no edge of one triangle meets the intersection of any other two triangles, and the edges of the projections do not overlap one another. For d-simplices with $d > 2$ we define general position analogously. With these assumptions, the total number of all i-dimensional faces with $0 \leq i \leq d$ of such envelopes was bounded in [11] and [5].

Fact 2.1 *The combinatorial complexity of the upper envelope of n d-simplices in $\mathbb{R}^{d+1}$ is $\Theta(n^d \alpha(n))$, where α is the functional inverse of the Ackermann function.*

In [6] a worst case optimal deterministic algorithm for constructing the upper envelope of a set of n triangles in three dimensions was given which needs $O(n^2\alpha(n))$ time and storage. However, it does not extend to surfaces or surface patches and its generalization to higher dimensions does not lead to a worst case optimal algorithm. If the d-simplices are pairwise disjoint, then the combinatorial complexity of their upper envelope is $\Theta(n^d)$ and it can be constructed using the above algorithm in the same time bound.

Our algorithm is similar in some features to the algorithm of [3] and [8] for constructing the arrangement of line segments in the plane. Therefore, we recall this result in the following. The simple randomized algorithms of [3] and [8] construct the *trapezoidal decomposition* induced by n line segments in the plane which are supposed to be non-vertical and in general position. This decomposition is a subdivision of the plane into *trapezoids* defined as follows. From every endpoint of a segment or an intersection point of two segments, we extend vertical segments to the first segment above and the first segment below this point. An example for a trapezoidal decomposition induced by four segments is shown in Figure 2 left. A trapezoid is *defined* by at most four

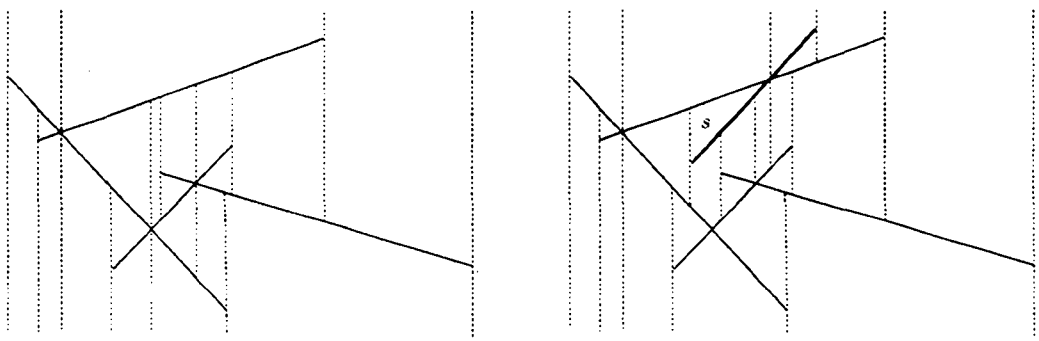

Figure 2: Trapezoidal decomposition of four segments and insertion of segment s

segments and may degenerate to a triangle. The problem of computing the trapezoidal decomposition of a set of line segments is solved incrementally, i. e. the segments are added one by one and we maintain the trapezoidal decomposition of the set of the already inserted segments. In order to do that, we need to define the notion of a *conflict*. A line segment and a trapezoid are in conflict, if and only if the segment intersects the interior of the trapezoid. The goal is to compute the trapezoids which do not conflict with any triangle. When a new segment s is inserted, we determine the trapezoids in conflict with s. Each such trapezoid is split into at most four new trapezoids. Some (at most two) of these new trapezoids not properly defined trapezoids yet. More precisely, each vertical segment which does not contain a vertex of the current arrangement must be removed and the two trapezoids incident to this segment are merged (see Figure 2 right). The following fact which generalizes to arrangement of planar curves of bounded degree was shown in [3] and [8].

Fact 2.2 *The arrangement of n line segments in the plane can be constructed on-line with $O(n + A)$ expected space and $O(\log n + \frac{A}{n})$ expected update time, where A is the number of intersections of the line segments.*

3 The Insertion Algorithm

The algorithm for constructing the upper envelope of a set of triangles in three dimensions described in the following is an incremental algorithm which uses the general technique of the *influence graph* of [1]. Its analysis is randomized.

The triangles are added one by one and we maintain a decomposition decomp$M(\mathcal{S})$ of the space above the current upper envelope $M(\mathcal{S})$ of the set $\mathcal{S}$ of already inserted triangles. This decomposition is a *prism decomposition* of a part of the space defined as follows: from every point on an edge of $M(\mathcal{S})$, we extend a vertical ray in positive x_3-direction (see Figure 3 left). By doing so we obtain a decomposition of the space above $M(\mathcal{S})$ in unbounded cells with a unique bottom face. However, the number of vertical walls of an obtained cell needs not to be constant and the cell may not be simply connected. Therefore, we decompose the vertical projection of its bottom face into trapezoids by drawing segments parallel to the x_2-axis. These segments are drawn dotted in Figure 3 right. The corresponding operation in three dimensions is to raise a wall vertically above each inserted segment. Each cell of the prism decomposition is now an unbounded prism with a trapezoidal base which may degenerate to a triangle. During the algorithm, we maintain the adjacency graph of the cells of the prism decomposition.

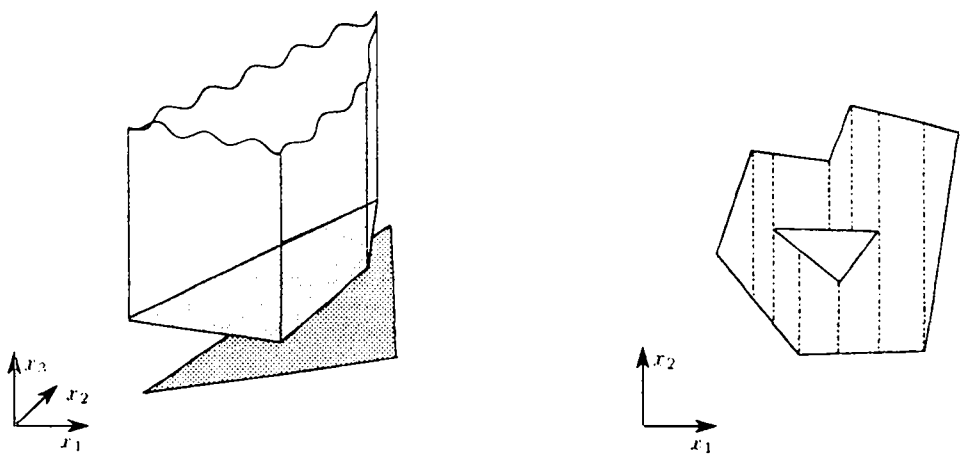

Figure 3: Vertical walls are raised to decompose cells into prisms

Each prism F is defined by a set of triangles one of which is the unique triangle t_F in which its *floor* is contained. The floor of F is a trapezoid which is defined by at most four segments $s_1, \ldots, s_b$ ($b \leq 4$) (as mentionned in Section 2) where the two

segments defining left and right vertical segment are not necessarily unique. These non-unique segments are edges of the same triangle. Each segment s_i is either a portion of intersection between the two triangles t_i and t_F or the projection of a portion of an edge of the triangle t_i onto t_F. Thus a prism is *defined* by at most five triangles: the triangle t_F and the at most four triangles $t_1, \ldots, t_b$ corresponding to $s_1, \ldots, s_b$.

We now define conflicts between triangles and prisms. A triangle and a prism are in *conflict*, if their intersection is not empty. The goal of our algorithm is to construct for a given set $\mathcal{S}$ of triangles the prisms which are defined by triangles of $\mathcal{S}$ and which do not conflict with any triangle of $\mathcal{S}$. Such prisms are called *empty*. These are the cells of the prism decomposition of the upper envelope of $\mathcal{S}$.

Let t be the new triangle to be inserted, $M(\mathcal{S})$ the current upper envelope and decomp$M(\mathcal{S})$ its prism decomposition. First, we determine the prisms of decomp$M(\mathcal{S})$ which intersect t. These prisms disappear from the current prism decomposition. The triangle t is their *killer*. Then, we update the upper envelope and its prism decomposition. The new prisms are *created* by t. The upper envelope after the update is called $M(\mathcal{S} \cup \{t\})$. To find the prisms which conflict with the new inserted triangle efficiently, we maintain the *influence graph* or I-DAG as location structure. This graph contains the history of the construction. Its nodes are associated with those prisms that belonged to the prism decomposition at a previous stage of the incremental construction. The I-DAG structure is characterized by the two following fundamental properties. Firstly, at each stage in the incremental construction, the empty prisms are leaves of the I-DAG. Secondly, the prism associated with a node is included in the union of the prisms associated with its parents. The I-DAG is initialized by a prism big enough to enclose all triangles in $\mathcal{S}$. Thus, this graph is rooted, directed and acyclic.

In the following we describe more precisely how to proceed, when a new triangle is added. We execute first a location step and then an update step.

Location step: When a new triangle t is added, we first locate all the prisms in conflict with t. This is done by a graph traversal that starts at the root of the I-DAG and visits recursively, for each node in the I-DAG, all its children, which have not already been visited and which conflict with t, and finally reports the leaves visited. If no leaf of the I-DAG is in conflict with t, t intervenes neither in the current upper envelope nor in the upper envelopes to be computed in the following.

Note that as opposed to other randomized constructions it is not sufficient to locate only the first conflicting prism. Indeed since the conflicting prisms do not necessarily belong to one connected component, we cannot find all conflicting prisms by using adjacency relations.

Update step: We partition the cells of decomp$M(\mathcal{S})$ which intersect t in those of *type 2* which are cut in two sub-cells by t and the others of *type 1*.

Firstly, we construct only the cells of the decomposition of the new upper envelope $M(\mathcal{S} \cup \{t\})$ whose floor is not contained in t. Therefore, let F be a cell of decomp$M(\mathcal{S})$ of type 1. The portion of F belonging to $M(\mathcal{S} \cup \{t\})$ is decomposed in at most six new

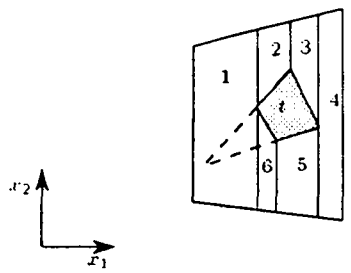

Figure 4: Six new cells whose floor is not t in a vertical projection.

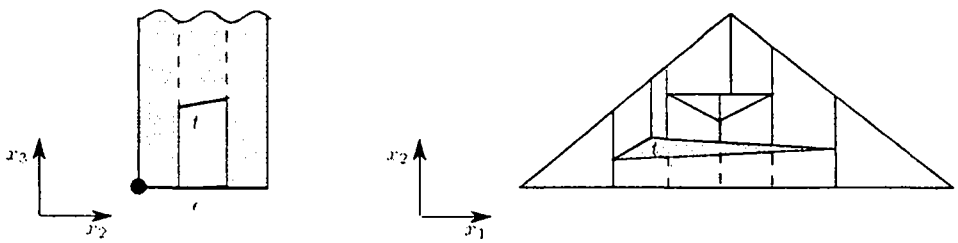

Figure 5: The merge seen in projection.

---

prisms (see Figure 4). A new node of the I-DAG is created for each new prism and linked to the node corresponding to F.

Some of the prisms are not proper prisms yet and have to be merged with adjacent prisms to obtain the prism decomposition of $M(S \cup \{t\})$. This merge is analogous to the merge used in the incremental construction of the trapezoidal decomposition of line segments in the plane and proceeds as follows. Each wall of F is cut by t in at most three wall portions (see Figure 5 right). The lower edge of at most two of such portions is not contained in t. Those wall portions which do not contain a vertex of the upper envelope are not walls of the new prism decomposition and have to be removed. Their two incident prisms have to be merged: the dotted walls in Figure 5 left are removed. The adjacency graph is updated accordingly.

At the end we have constructed all those cells of the decomposition of the new upper envelope whose floor is not contained in t. We still have to construct the remaining cells whose floor is contained in t. We will denote by U the union of the cells of type 2 and of those parts of cells of type 1 (whose base is not necessarily trapezoidal) not treated in the first step. As it can be seen in Figure 6. U is not necessarily connected.

We compute U using the adjacency graph of the cells. A connected component of U corresponds to a face f of the upper envelope $M(S \cup \{t\})$ above which the maximum height is assumed by t. The nodes in the I-DAG corresponding to the cells of the

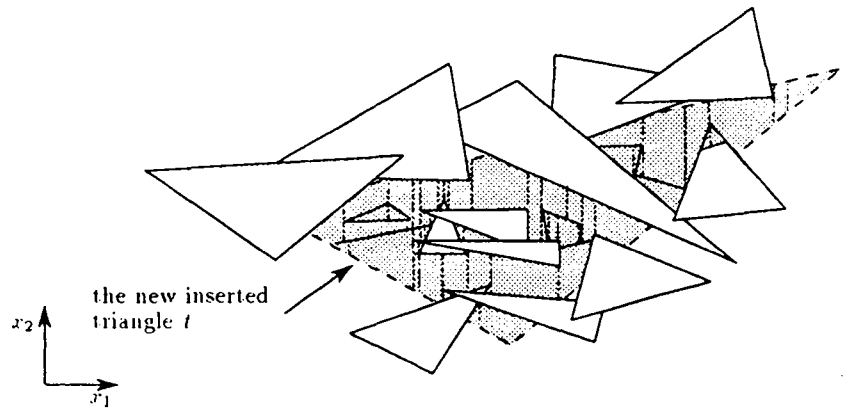

Figure 6: The new inserted triangle t and union U in a vertical projection.

connected component of U are linked to a unique node associated to f. The face f may contain a great number of edges and be not simply connected. We decompose it into trapezoids using the incremental randomized algorithm in [1] and raise a wall vertically above each edge of the trapezoidal decomposition. By that we obtain a secondary influence graph which represents the prism decomposition of the part of the space above f. The root of this graph is the node associated to f. We repeat this process for all the connected components of U.

We conclude the description of the update step and by that the description of the algorithm with a remark. Note that it is not possible to construct the new cells whose floor is contained in t directly, since in this case the number of sons of a node cannot be bounded, a condition needed in the analysis of [1]. An example is illustrated in figure 7. In this figure, each of the $O(r)$ nodes corresponding to the prisms F_i may have $O(r)$ sons corresponding to the new prisms, since their prisms intersect partially. In this example also the update conditions for the conflict graph in [3] is not fulfilled.

4 Analysis of Insertions

We will now analyze the running time using the main results from [1] and the backwards analysis from [4] and [13]. In the influence graph we distinguish between the *principal* nodes corresponding to empty prisms at a previous stage of the incremental construction and *secondary* nodes which are inner nodes of the secondary influence graphs.

For the analysis we need some notations. According to the preceding section each prism is defined by at most five triangles. Let $f^0(r)$ be the expected number of empty prisms defined by subsets of at most five triangles of an r-random sample of $\mathcal{S}$. The expected number of prisms defined by subsets of size at most five of an r-random sample of $\mathcal{S}$ with exactly one conflict $f^1(r)$ is bounded in [3] by $O(f^0(\lfloor \frac{r}{2} \rfloor))$.

First we give a bound for the expected size of the influence graph constructed during

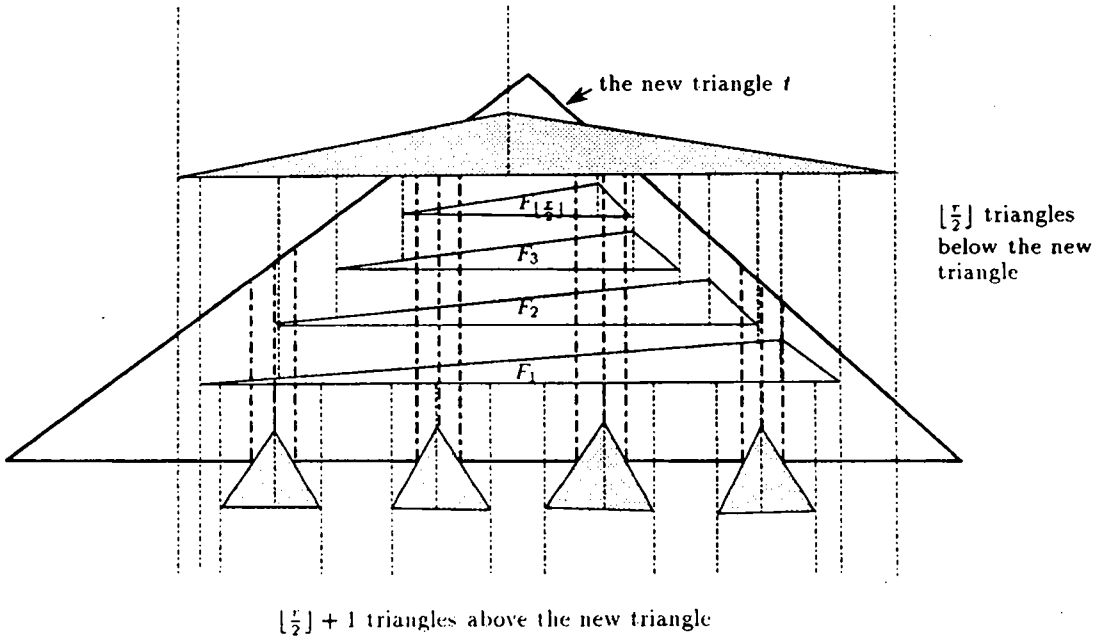

Figure 7: The number of sons of a node in the I-DAG cannot be bounded

---

our algorithm.

Lemma 4.1 *The expected total number of nodes and edges in the influence graph of* $\mathcal{S}$ *is* $O(\sum_{r=1}^{n} \frac{f^0(r)}{r})$.

Proof: The expected number of principal nodes created by the insertion of the r-th triangle is less than $\frac{5f^0(r)}{r}$. Indeed, a principal node created by the insertion of the r-th triangle t_r is necessarily an empty prism defined by the r triangles. By averaging over the $n!$ possible permutations of $\mathcal{S}$, the first r triangles may be any sample of $\mathcal{S}$ of size r with the same probability and t_r may be any element of the sample with probability $\frac{1}{r}$. Hence, $\frac{5}{r}$ is an upper bound for the probability for t_r to be one of the at most 5 triangles defining a prism.

To bound the number of secondary nodes in the I-DAG created at stage r of the algorithm, we observe that the size of the union U_r is bounded by the number of empty prisms in conflict with the r-th triangle t_r. These are the prisms killed by t_r. Since such a prism has exactly one conflict after t_r's addition, namely t_r, the number of prisms killed by t_r is expected to be $\frac{1}{r}f^1(r)$. The complexity of an arrangement of $\frac{1}{r}f^1(r)$ non-intersecting line segments is linear in the number of line-segments. Therefore using Fact 2.2 it can be incrementally constructed in expected space $O(\frac{f^1(r)}{r})$. This is exactly the expected number of secondary nodes created during the insertion of the r-th triangle and is less than $O(\frac{f^0(r)}{r})$.

The number of nodes in the influence graph of $\mathcal{S}$ is simply the sum over all stages of the expected number of nodes created at one stage. By the preceding observations the expected number of principal and secondary nodes created at stage r is less than $O(\frac{f^0(r)}{r})$. It follows that the expected total number of nodes in the influence graph is $O(\sum_{r=1}^{n} \frac{f^0(r)}{r})$. Since the outdegree of each node in the I-DAG is bounded, this is also an upper bound for the number of edges in the I-DAG. □

According to Fact 2.1, $f^0(r)$ is less than $O(r^2\alpha(r))$. Thus, the total number of nodes and edges created during the insertion of n triangles is bounded by $O(n^2\alpha(n))$.

Now, we analyze the location and the update step for the insertion of the last triangle t. A principal node and the secondary influence graph of which it is a parent can be treated in $O(\log n)$ expected time. Indeed, according to Fact 2.2, the location in a secondary influence graph can be done in $O(\log n)$ expected time. Hence the expected costs of the location step are at most to $\log n$ times the expected number of principal nodes visited.

The construction of the secondary influence graph needs expected time $O(|U| \log n)$, where $|U|$ is the size of the union U and is proportional to the number of empty prisms in conflict with the n-th inserted triangle. The other parts of the update step requires time proportional to the number of edges created by the insertion of the n-th inserted triangle. Since the nodes in the I-DAG are of bounded outdegree, this is bounded by the number of empty prisms in conflict with n-th inserted triangle, which is less than the expected number of principal nodes visited. Thus also the update step needs time proportional to $\log n$ times the expected number of principal nodes visited.

The expected number of principal nodes visited during the last insertion is given in the following lemma.

Lemma 4.2 *The expected total number of principal nodes visited during the location step for the n-th inserted triangle is $O(\sum_{r=1}^{n} \frac{f^0(\lfloor r/2 \rfloor)}{r^2})$.*

Proof: During the insertion of the n-th triangle, the conflicts are located by a traversal of the influence graph. A principal node F is visited and yields a positive answer, if F is in conflict with the n-th triangle.

Firstly, we show that the expected number of conflicts of the n-th triangle t with prisms, which were empty at stage $r < n$, is $\frac{f^1(r+1)}{r+1}$. By averaging over all possible permutations of S, the first r triangles plus t may be any sample of size $r+1$ with the same probability. Then the prisms, which were empty at stage r and are in conflict with t, are defined by subsets of these $r+1$ elements and have exactly one conflict after t's insertion. Since t may be any element of the sample of size $r+1$, with probability $\frac{1}{r+1}$, we obtain the above expression.

Secondly, we show that the expected number of conflicts of the n-th triangle with the prisms, created by the insertion of the r-th triangle with $r < n$, is less than $\frac{5}{r}\frac{f^1(r+1)}{r+1}$. This expected value is the sum over all prisms F of the probability that F occurs as a principal node at stage r and is in conflict with t. We decompose this probability conditionnally with the event that F is in conflict with t and had no conflict at stage r. If F is in conflict with t and has no conflict at stage r, the probability that F occurs as a principal node at stage r is $\frac{5}{r}$. Otherwise, if F has a conflict at stage r, F could not have been created by the r-th triangle and thus the probability is zero. Hence, we obtain with the preceding observation the expected value $\frac{5}{r}\frac{f^1(r+1)}{r+1}$.

Now, summation over the stage of creation r of F yields the result of the lemma $\sum_{r=1}^{n-1} \frac{5}{r}\frac{f^1(r+1)}{r+1} = O\left(\sum_{r=1}^{n} \frac{f^0(\lfloor r/2 \rfloor)}{r^2}\right)$. □

This expected number is bounded by $O(n\alpha(n))$. Thus, the location and the update step for inserting the n-th triangle may cost in total $O(\log n \sum_{r=1}^{n} \frac{f^0(\lfloor r/2 \rfloor)}{r^2})$ expected time, which is less than $O(n\alpha(n)\log n)$.

This completes the analysis of the algorithm and yields the following theorems.

Theorem 4.3 *The n-th triangle can be inserted in $O(\log n \sum_{r=1}^{n} \frac{f^0(\lfloor r/2 \rfloor)}{r^2})$ expected time where $f^0(r)$ is the expected size of the upper envelope of r triangles.*

Corollary 4.4 *The n-th triangle can be inserted in $O(n\alpha(n)\log n)$ expected time.*

Theorem 4.5 *The upper envelope of a set of n triangles in $\mathbb{R}^3$ can be constructed in $O(n \log n \sum_{r=1}^{n} \frac{f^0(\lfloor r/2 \rfloor}{r^2}))$ expected time with $O(\sum_{r=1}^{n} \frac{f^0(r)}{r})$ expected space.*

Corollary 4.6 *The upper envelope of a set of n triangles in $\mathbb{R}^3$ can be constructed in $O(n^2\alpha(n)\log n)$ expected time with $O(n^2\alpha(n))$ expected space .*

The results stated in the Corollaries 4.3 and 4.5 are obtained by using worst case bounds for the size of an upper envelope. In most pratical situations the performance of the algorithm will be much better. If the triangles do not intersect, $f^0(r) = O(r^2)$

which results in an $O(n \log n)$ expected insertion time for the n-th triangle. If the triangles are half-planes limited by straight lines, then $f^0(r) = O(r^2)$ and we obtain the same insertion time as for the case of non-intersecting triangles.

Note, that, in general, an incremental algorithm cannot be output sensitive, since an intermediate result of the incremental construction may be greater than the final one. This can be easily illustrated by the following example. Let $\mathcal{S}$ be a set of n non-intersecting triangles of Figure 8. The upper envelope for $\mathcal{S}$ is quadratic, but if a new triangle t is inserted, which is above all triangles, the upper envelope of $\mathcal{S} \cup \{t\}$ becomes constant. The insertion of t takes at least time proportional to the size of the envelope before its insertion. Thus, the expected costs of inserting t are at least $\Omega(\frac{1}{n} \sum_{i=1}^{n} i^2) = \Omega(n^2)$.

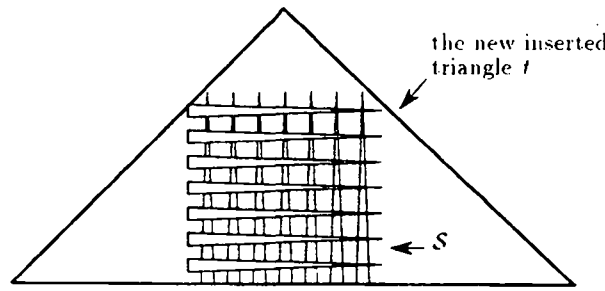

Figure 8: Example for which the algorithm cannot be output sensitive in a vertical projection

5 Extension of the Insertion Algorithm

The above algorithm can be generalized with slight modifications to calculate the upper envelope of surfaces of fixed maximum degree. Therefore we first define the problem precisely and recall known results.

Let $\mathcal{S}$ be a set of n surfaces (resp. surface patches) in $\mathbb{R}^3$. Each surface is a continuous bivariate function satisfying the two following conditions. Firstly, each triple of functions intersects in at most s points. Secondly, each pair of functions intersects in a curve having at most p singular points, at which the curve ceases to be defined, has a discontinuity or a vertical tangency. Algebraic surfaces of fixed maximum degree meet the above conditions. Each surface patch is the graph of a partially defined bivariate function and these functions as well as the curves delimiting these patches are of fixed maximum degree and the number of delimiting curves is constant. The *upper envelope* M of $\mathcal{S}$ is the pointwise maximum of these functions. In [12] it has been conjectured that the combinatorial complexity of such envelopes is only about quadratic in n. However, this conjecture has only been proved for a few types of surfaces and surface patches such as piecewise linear functions in [11] and some special cases of bivariate functions

in [12]. If in addition to the conditions above, the interior of three surface patches intersect in at most two points, it has been recently shown in [7] that the complexity of the envelope is $O(n^2\sqrt{\lambda_q(n)})$, where q is a constant depending on the degree of the surfaces and where $\lambda_q(n)$ is the maximum length of a (n,q) Davenport-Schinzel sequence.

Now we describe the algorithm. As before we first look at the trapezoidal decomposition of planar curves of bounded degree. The algorithm of Section 2 can be extended as in [1], since the segments can be broken in a constant number of x-monotonic portions and every two segments intersect only constant times. The top and bottom edges of a trapezoid in this case are portions of x-monotonic planar curves. Its prism decomposition is then defined as in section 3 with the slight difference that the floor of a prism is contained in a surface and that its projection is a curved trapezoid. Now we have everything together such that the algorithm for the construction of the upper envelope of the set $\mathcal{S}$ as defined above works nicely. For its analysis we assume that each operation involving two or three functions can be done in constant time. Hence we obtain the same bounds as in the Theorems 4.3 and 4.5 depending on the expected number $f^0(r)$ of empty prisms defined by an r-sample of $\mathcal{S}$.

Corollary 5.1 *The n-th surface (resp. surface patch) can be inserted in expected time $O(\log n \sum_{r=1}^{n} \frac{f^0(\lfloor r/2 \rfloor)}{r^2})$ where $f^0(r)$ is the expected size of the upper envelope of r surfaces (resp. surface patches).*

In the non-linear cases, our algorithm for computing the upper envelope has the same time complexity as the algorithms presented in [12] but is on-line. For example, it can be used to compute the upper envelope for the surface patches considered in [7].

Corollary 5.2 *The n-th surface patch can be inserted in $O(n\sqrt{\lambda_q(n)}\log n)$ expected time, where q is a constant depending on the degree of the surfaces.*

6 Conclusion

In this paper we have given a simple on-line algorithm for constructing the upper envelope of triangles in three dimensions which allows the successive insertion of triangles without knowing them in advance. Its analysis is randomized and extends the general technique from [1] to two layers of influence graphs. The running-time of our algorithm is optimal up to a $\log n$ factor in the worst case. Its behaviour is good in special cases and in practice. It also applies to surfaces and surface patches of fixed maximum degre. It remains open if it is possible to remove the $\log n$ factor in our analysis. Furthermore, it would be nice to make the algorithm fully dynamic so as to allow also deletions of triangles. Another open problem is its generalization to higher dimensions.

References

[1] J.-D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Applications of random sampling to on-line algorithms in computational geometry. *Discrete Comput. Geom.*, 8:51–71, 1992.

[2] J. D. Boissonnat and K. Dobrindt. Randomized construction of the upper envelope of triangles in R^3. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 311–315, 1992.

[3] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[4] O. Devillers. Randomization yields simple $O(n \log^* n)$ algorithms for difficult $\Omega(n)$ problems. *Internat. J. Comput. Geom. Appl.*, 2(1):97–111, 1992.

[5] H. Edelsbrunner. The upper envelope of piecewise linear functions: tight complexity bounds in higher dimensions. *Discrete Comput. Geom.*, 4:337–343, 1989.

[6] H. Edelsbrunner, L. Guibas, and M. Sharir. The upper envelope of piecewise linear functions: algorithms and applications. *Discrete Comput. Geom.*, 4:311–336, 1989.

[7] D. Halperin and M. Sharir. New bounds for lower envelopes in three dimensions, with applications to visibility in terrains. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, 1993.

[8] K. Mulmuley. A fast planar partition algorithm, I. In *Proc. 29th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 580–589, 1988.

[9] K. Mulmuley. An efficient algorithm for hidden surface removal. *Comput. Graph.*, 23(3):379–388, 1989.

[10] K. Mulmuley. On obstructions in relation to a fixed viewpoint. In *Proc. 30th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 592–597, 1989.

[11] J. Pach and M. Sharir. The upper envelope of piecewise linear functions and the boundary of a region enclosed by convex plates: combinatorial analysis. *Discrete Comput. Geom.*, 4:291–309, 1989.

[12] J. T. Schwartz and M. Sharir. On the two-dimensional Davenport-Schinzel problem. *J. Symbolic Comput.*, 10:371–393, 1990.

[13] R. Seidel. Backwards analysis of randomized geometric algorithms. Manuscript, ALCOM Summerschool on efficient algorithms design, Arhus, Denmark, 1991.

[14] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl.*, 1:51–64, 1991.

[15] I. E. Sutherland, R. F. Sproull, and R. A. Shumacker. A characterization of ten hidden surface algorithms. *ACM Comput. Surv.*, 6:1–55, 1974.

Unité de Recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 SOPHIA ANTIPOLIS Cedex (France)

Unité de Recherche INRIA Lorraine Technopôle de Nancy-Brabois - Campus Scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 VILLERS LES NANCY Cedex (France)
Unité de Recherche INRIA Rennes IRISA, Campus Universitaire de Beaulieu 35042 RENNES Cedex (France)
Unité de Recherche INRIA Rhône-Alpes 46, avenue Félix Viallet - 38031 GRENOBLE Cedex (France)
Unité de Recherche INRIA Rocquencourt Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

EDITEUR
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 LE CHESNAY Cedex (France)

ISSN 0249 - 6399