Randomized Construction of the Upper Envelope of Triangles in $\mathbb{R}^3$*

J. D. Boissonnat† K. Dobrindt†

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. Adapting the analysis technique from [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. Thus, in the worst case the expected time for the insertion of the last triangle is bounded by $O(n\alpha(n)\log n)$. This algorithm is easy to implement and works also nicely for surfaces of fixed maximum degree. Furthermore, it is an algorithmic tool for solving translational motion planning problems on a terrain.

1 Introduction

In this paper we consider the following problem. 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* of the d-simplices is the pointwise maximum of these functions. For an example of an upper envelope of a set of triangles see Figure 1. We wish to compute the upper envelope of a set of triangles in three dimensions. In [8] and [4] the total number of all i-dimensional faces of such envelopes was analyzed. It was shown that 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.

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 the possible executions of the algorithm. In [1] a data structure, the *influence graph*, for the design of on-line geometric algorithms was developed which provides bounds for their expected space and time complexities when averaging over all permutations of the input data. In the following we will present an on-line randomized algorithm for constructing the upper envelope of a set of n triangles in three dimensions which adapts this general technique to 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, in the worst case it is bounded by $O(n\alpha(n)\log n)$. However, in most pratical situations its behavior will be much better. This algorithm is simple and it can be used for surfaces of fixed maximum degree with the same time complexity depending on $f^0(r)$.

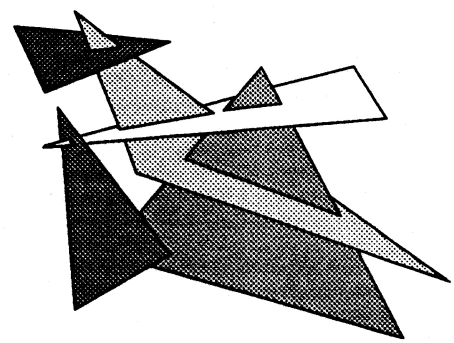

Figure 1: The upper envelope of five triangles

In [5] a worst case optimal deterministic algorithm for constructing the upper envelope of a set of n triangles in three dimensions was presented, which takes $O(n^2\alpha(n))$ time and storage. However, since the algorithm includes the computation of an arrangement of $3n$ lines in the plane, its lower bound is $\Omega(n^2)$. In the case of the pairwise disjoint d-simplices 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. The hidden surface removal problem where the viewpoint is assumed to be at infinity in the direction of the third coordinate axis is a special case of the envelope problem. An incremental randomized but static algorithm for hidden surface removal of non-intersecting faces in three dimensions whose expected running time is $\mathcal{O}(n\log^2 n + \theta(1)\log n)$ was given in [6]. The value $\theta(1)$ of the θ series is defined to be $\sum_l \frac{\text{number of junctions at level } l}{l}$, where the *level* of a junction is the number of faces which make it invisible. Hence, $\theta(1)$ is $O(n^2)$ in the worst case. In [7] a running time that depends linearly on $\theta(1)$ is achieved. In [9] bounds on the size of the upper envelope for special cases of bivariate functions were obtained and deterministic algorithms for their calculation by a factor $\log n$ worse than the bounds were presented. In the non-linear cases our general algorithm has the same time complexity as these specialized algorithms.

†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 77 65
e-mail : boissonn@sophia.inria.fr dobrindt@sophia.inria.fr

*This work was partially supported by the ESPRIT Basic Research Actions Nr. 3075 (ALCOM) and Nr. 7141 (ALCOM II).

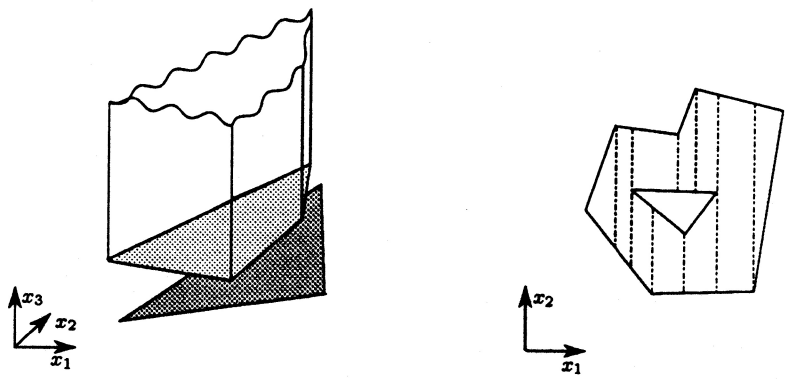

Figure 2: Vertical walls are raised to decompose cells into prisms

2 The Algorithm

The algorithm described in the following is an incremental algorithm and 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 $\mathcal{C}$ of the space above the current upper envelope $\mathcal{U}$ 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 $\mathcal{U}$ extend a vertical ray in positive z-direction (see Figure 2 left). By doing so we obtain a decomposition of the space above $\mathcal{U}$ 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 y-axis. These segments are drawn dotted in Figure 2 right. The corresponding operation in three dimension is to raise a wall vertically above each inserted segment. Each cell of the prism decomposition is now a 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.

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 $s_1, \ldots, s_b$ ($b \leq 4$) where two of them are not necessarily unique. 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, $\mathcal{U}$ the current upper envelope and $\mathcal{C}$ its prism decomposition. First, we determine the prisms of $\mathcal{C}$ which intersect t. Then, we update the upper envelope and its prism decomposition. The upper envelope after the update is called $\mathcal{U}'$. To find the prisms which conflict with the new inserted triangle efficiently, we maintain the *influence graph* or I-DAG as location structure, which contains the history. The nodes of this graph are associated with those prisms that belonged to the prism decomposition at some stage in 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. We initialize the I-DAG with a prism big enough to enclose all triangles in $\mathcal{S}$. Therefore, 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.

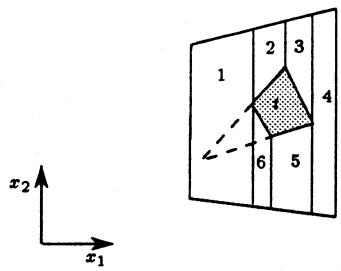

Figure 3: Six new cells whose floor is not t in a vertical projection.

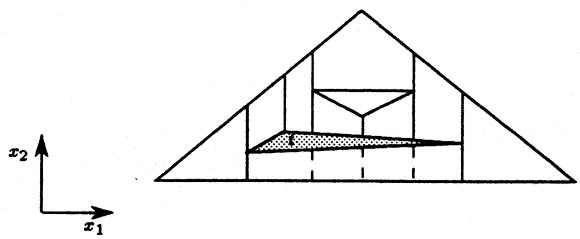

Figure 4: The merge seen in projection onto a horizontal plan: the dotted walls are removed.

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

Firstly, we construct only the cells of the new upper envelope $\mathcal{U}'$ whose floor is not contained in t. Therefore, let F be a cell of $\mathcal{C}$ of type 2. The portion of F belonging to $\mathcal{U}'$ is decomposed in at most six new prisms (see Figure 3). 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 $\mathcal{U}'$. This merge is analogous to the merge used in the incremental construction of the trapezoidal decomposition of line segments in the plane (see for example [2]) and proceeds as follows. Each wall of F is cut by t in at most three wall portions. The lower edge of at most two of such portions is not contained in t. Those wall parts 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 (see Figure 4). The adjacency graph is updated accordingly.

At the end we have constructed all those cells of the new upper envelope whose floor is not contained in t. We still have to construct the remaining cells. We will denote the union of the cells of type 1 and of those parts of cells of type 2 (prisms whose base is not necessarily trapezoidal) whose floor is contained in t by $\mathcal{P}$. As it can be seen in Figure 5, $\mathcal{P}$ is not necessarily connected.

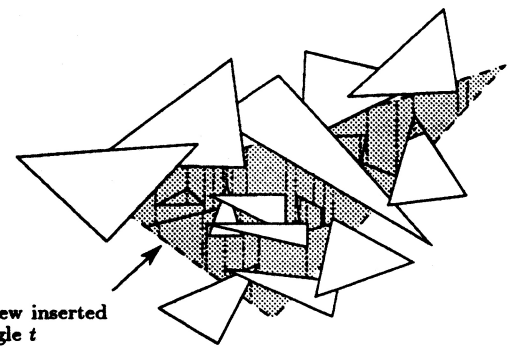

Figure 5: The new inserted triangle t and union $\mathcal{P}$ in a vertical projection.

We compute $\mathcal{P}$ using the adjacency graph of the cells. A connected component of $\mathcal{P}$ corresponds to a face f of the upper envelope $\mathcal{U}'$ above which the maximum height is assumed by t. We construct the trapezoidal decomposition of f using the incremental randomized algorithm in [1] and raise a wall vertically above each edge of the trapezoidal decomposition. Thus, we obtain a secondary influence graph which represents the prism decomposition of the part of the space above f. Then we link the nodes in the I-DAG corresponding to the cells of the connected component of $\mathcal{P}$ to the root of this secondary influence graph. We repeat this process for all the connected components of $\mathcal{P}$.

This concludes the description of the update step and by that the description of the algorithm.

3 Analysis of the Algorithm

We will now analyze the running time using the main results from [1]. In our graph we distinguish between the *principal* nodes corresponding to empty prisms at some stage in 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 an r-random sample of $\mathcal{S}$. The expected number $f^1(r)$ of prisms defined by an r-random sample of $\mathcal{S}$ with exactly one conflict is bounded in [2] by $O(f^0(\lfloor r/2 \rfloor))$. Although, we are not exactly in the framework of [1] because of the presence of the secondary nodes, we can apply their proofs to our case without modification to bound the number of principal nodes created and visited during a location step.

It follows that the expected total number of principal nodes in the influence graph is $O(\sum_{r=1}^{n} \frac{f^0(r)}{r})$. To bound the number of secondary nodes in the I-DAG of $\mathcal{S}$, we observe that at stage r of the algorithm the size of the union $\mathcal{P}_r$ is bounded by the number of princi-

pal nodes in conflict with the r-th triangle t_r. Since the width of a prism refined due to the addition of t_r is one after t_r's addition, this number is expected to be $\frac{1}{r}f^1(r)$. According to the result from [2] and [1] for the incremental randomized construction of the trapezoidal decomposition of non-intersecting line segments, the expected number of secondary nodes created during the insertion of the r-th triangle is $O(\frac{f^1(r)}{r})$, which is less than $O(\frac{f^0(r)}{r})$. Thus the expected total number of secondary nodes is bounded by $O(\sum_{r=1}^{n}\frac{f^0(r)}{r})$. 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. In the worst case, $f^0(r)$ is $O(r^2\alpha(r))$. Thus the total number of nodes and edges created during the insertion of n triangles is bounded by $O(n^2\alpha(n))$.

According to the results from [2] and [1] cited above, the location in a secondary influence graph can be done in $O(\log n)$ expected time, and the update of $\mathcal{P}$ needs expected time $O(k\log n)$, where k is the size of $\mathcal{P}$ which is proportionnal to the number of conflicting principal nodes. The other parts of the update step require time proportionnal to the number of principal nodes in conflict with t. It follows that the expected costs of the location and the update step are proportionnal to $\log n$ times the expected number of principal nodes visited. The expected total number of principal nodes visited is $O(\sum_{r=1}^{n}\frac{f^0(\lfloor r/2\rfloor)}{r^2})$, which 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 1 *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 1 *The n-th triangle can be inserted in $O(n\alpha(n)\log n)$ expected time.*

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

Corollary 2 *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 and in $O(n^2\alpha(n))$ expected space .*

The results stated in the Corollaries 1 and 2 are obtained by using worst case bounds for the size of a upper envelope. In most pratical situatons, however, 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, then $f^0(r) = O(r^2)$ and we obtain the same insertion time as for the case of non-intersecting triangles.

Note that since we can identify each connected component of $\mathcal{P}$, we can use the algorithms of [3] or [10] and compute the trapezoid decomposition of $\mathcal{P}$ in $O(k\log^* n)$ expected time. However, this does not change the overall complexity of our algorithm, because the location in a secondary influence graph still requires $O(\log n)$ expected time.

4 Extension

The above algorithm can be generalized with slight modifications to calculate the upper envelope of surfaces of fixed maximum degree.

Therefore, let $\mathcal{S}$ be a set of n continuous bivariate functions 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. The *upper envelope* of $\mathcal{S}$ is the pointwise maximum of these functions. Its prism decomposition is defined as in section 2. In this case, the floor of a prism is a trapezoid whose top and bottom edges are portions of x-monotonic planar curves. The algorithm for the construction of the upper envelope of the set $\mathcal{S}$ as defined above works nicely using the result from [1] on the computation of the trapezoidal decomposition of planar curves of bounded degree. For its analysis we assume that each operation involving two or three functions can be done in constant time. By that we obtain the same bounds as in the Theorems 1 and 2 depending on the expected number f^0 of empty prisms defined by an r-sample of $\mathcal{S}$.

Corollary 3 *The n-th surface 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 surfaces.*

However, non-trivial bounds for f^0 have only been proved for piecewise linear functions in [8] and for some special cases of bivariate functions in [9]. In the non-linear cases, our algorithm for computing their upper envelope has the same time complexity as the algorithms presented there but is on-line. For example, if we assume in addition to the above conditions that each intersection curve intersects the plane $x = const$ in at most two points we get the following corollary.

Corollary 4 *The n-th surface can be inserted in $O(\lambda_{s+2}(n)\log n)$ expected time, where $\lambda_{s+2}(n)$ is the maximum length of a $(n, s+2)$ Davenport-Schinzel sequence.*

5 Applications

In addition to the applications studied in [5] we present two further applications in the field of motion planning. Moving a robot on a terrain requires the computation of all free and stable placements of the robot. To do this, the terrain is modelled by a polyhedral terrain $\mathcal{T}$, i.e. the graph of a real bivariate continuous function, with n triangular faces. The model of the robot is an arbitrary polyhedron P with m vertices. At l of its vertices, vertical springs with a certain maximal extension h are attached. We now want to find all translations of P avoiding $\mathcal{T}$ at which the vertical distance from each of the l chosen vertices of the polyhedron to $\mathcal{T}$ is less than the maximal extension h.

To obtain the set of stable placements we first form the Minkowski differences of $\mathcal{T}$ and v_i where v_i is a chosen vertex of P with $1 \leq i \leq l$ and translate them then by h in a vertical upwards direction. The set of all stable placements is the lower envelope of these l translations of $\mathcal{T}$. The size of this lower envelope is $O(l^3n^2)$ for $l \leq \alpha(ln)$ (otherwise $O((ln)^2\alpha(ln))$), since we can replace the upper bound for the size of the lower envelope restricted over some line used in the proof of main theorem in [8] by $O(l^2n)$, which is an upper bound for the size of the lower envelope of l monotone chains of length n.

From [5] it follows that the set of free placements is the upper envelope of the Minkowski differences of the convex parts, which partition $\mathcal{T}$, and P.

Now, instead of first constructing the set of all stable placements and the set of free placements separately and then computing their intersection to obtain the solution space, we slightly modify our algorithm for the computation of the envelopes to compute the solution space directly. The complexity of the solution space is therefore $O((mn)^2\alpha(mn))$.

To model a more realistic robot with wheels, we attach spheres with radius r at the bottom endpoints of the springs. In this case we have to construct upper and lower envelopes of triangles and spheres and cylinders with radius r by applying the extended algorithm of the previous section.

6 Conclusion

In this paper we have given a simple on-line algorithm for constructing the upper envelope of triangles in three dimensions. Its analysis is randomized and adapts the general technique from [1]. However, it remains open if it is possible to remove the $\log n$ factor in our analysis. Furthermore, a main remaining open problem is the generalization of the algorithm 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 and Computational Geometry.* To appear. Available as Technical Report INRIA 1285. Abstract published in IMACS 91 in Dublin.

[2] K.L. Clarkson and P.W. Shor. Applications of random sampling in computational geometry, II. *Discrete and Computational Geometry*, 4(5), 1989.

[3] O. Devillers. Randomization yields simple $O(n \log^* n)$ algorithms for difficult $\Omega(n)$ problems. *International Journal of Computational Geometry and Applications*, 2(1), March 1992.

[4] H. Edelsbrunner. The upper envelope of piecewise linear functions: tight bounds on the number of faces. *Discrete and Computational Geometry*, 4:337–343, 1989.

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

[6] K. Mulmuley. An efficient algorithm for hidden surface removal. In *ACM SIGGRAPH, Symposium on Computer Graphics, Vol. 23*, pages 379–388, 1989.

[7] K. Mulmuley. On obstruction in relation to a fixed viewpoint. In *IEEE Symposium on Foundations of Computer Science*, pages 592–597, 1989.

[8] 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 and Computational Geometry*, 4:291–309, 1989.

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

[10] R. Seidel. A simple and fast randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry Theory and Applications*, 1(1):51–64, 1991.