On the Size of the 3D Visibility Skeleton: Experimental Results

Linqiao Zhang, Hazel Everett, Sylvain Lazard, Christophe Weibel, Sue Whitesides

On the Size of the 3D Visibility Skeleton: Experimental Results

Linqiao Zhang[1], Hazel Everett[2], Sylvain Lazard[2],
Christophe Weibel[3], and Sue Whitesides[1]

[1] McGill University, School of Computer Science, Montreal, Quebec H3A 2A7, Canada.
{lzhang15, sue}@cs.mcgill.ca
[2] INRIA Nancy Grand Est , Université Nancy 2, LORIA, Nancy, France.
FirstName.LastName@loria.fr
[3] McGill University, Math Department, Montreal, Quebec H3A 2A7, Canada.
weibel@math.mcgill.ca

Abstract. The 3D visibility skeleton is a data structure used to encode global visibility information about a set of objects. Previous theoretical results have shown that for k convex polytopes with n edges in total, the worst case size complexity of this data structure is $\Theta(n^2k^2)$ [Brönnimann et al. 07]; whereas for k uniformly distributed unit spheres, the expected size is $\Theta(k)$ [Devillers et al. 03].
In this paper, we study the size of the visibility skeleton experimentally. Our results indicate that the size of the 3D visibility skeleton, in our setting, is $Ck\sqrt{nk}$, where C varies with the scene density but remains small. This is the first experimentally determined asymptotic estimate of the size of the 3D visibility skeleton for reasonably large n and expressed in terms of both n and k. We suggest theoretical explanations for the experimental results we obtained. Our experiments also indicate that the running time of our implementation is $O(n^{3/2}k\log k)$, while its worst-case running time complexity is $O(n^2k^2\log k)$.

1 Introduction

Computing visibility information in a 3D environment is crucial to many applications such as computer graphics, vision and robotics. Typical visibility problems include computing the view from a given point, determining whether two objects partially see each other, and computing the umbra and penumbra cast by a light source.

In a given scene, two points are visible if the segment joining them does not properly intersect any obstacle in the scene. The study of visibility is thus intimately related to the study of the set of free line segments in a scene. The visibility complex, which is, roughly speaking, a partition of the space of maximal free line segments into connected components of segments that touch the same objects, was proposed by Pocchiola and Vegter as a data structure encoding visibility information of a scene in 2D [21]. Durand et al. [8, 10] initiated the study of the visibility complex in 3D, and furthermore, introduced the visibility skeleton, which is essentially the zero and one-dimensional cells of the visibility complex, arguing that this smaller and simpler structure suffices for solving interesting visibility queries related to shadow computation [8, 9]. Although smaller than the visibility complex, the visibility skeleton is, nevertheless, a potentially

enormous structure; it has worst-case size complexity $O(n^4)$. Note that the visibility skeleton is related to the aspect graph [19] as the embedding of the visibility skeleton in $\mathbb{R}^3$ is almost the partition of $\mathbb{R}^3$ whose dual is the aspect graph.

Durand et al. implemented the visibility skeleton for polygonal scenes and demonstrated that use of the skeleton indeed results in higher quality light simulation in rendered images with improved computation time compared to previous algorithms [9]. Despite these positive results their pioneering approach suffers a major drawback. Their algorithm is not efficient because it is based on a systematic enumeration of all possibilities and thus has worst-case time complexity $\Theta(n^5)$ although the use of heuristics gives an observed complexity $\Theta(n^{2.4})$ [8]. Two other implementations, one by Glaves [14] for random triangles and another by Schröder [22] for polytopes, yielded similar results. Duguet et al. [7] report an implementation that computes elements of the visibility skeleton needed for light simulations in the restricted setting of point-light sources (possibly at infinity); their results can thus not be compared to those mentioned previously.

Worst-case examples are somewhat artificial and indeed Durand et al. [8] provide empirical evidence indicating that the $O(n^4)$ worst-case upper bound on the size of the visibility skeleton is largely pessimistic in practical situations; they observe a quadratic growth rate, albeit for the rather small scenes (at most 1,500 triangles) that they were able to test. Again, these results were experimentally confirmed by Glaves [14] and Schröder [22]. Some recent theoretical results also support the observation that the $O(n^4)$ is pessimistic. When the inputs are k polytopes with n edges in total, [17, 2] show that the number of vertices of the visibility skeleton is $\Theta(n^2k^2)$ in the worst case. If the polytopes are disjoint and their silhouettes have worst-case complexity $O(\sqrt{n/k})$, then the size of the visibility skeleton is $O(nk^2\sqrt{nk})$ [15]. Moreover, when the inputs are randomly distributed unit balls, the expected size is linear [6]. Nevertheless, the problem of estimating the size of the visibility skeleton in practice for reasonably large input scenes has remained open for years because of the absence of a sufficiently efficient implementation.

We address in this paper the problem of computing and estimating the size of the visibility skeleton of k disjoint polytopes of total complexity n in generic position. In fact, we measure the size of the skeleton as the number of its vertices (since vertices have constant degree under general position assumptions). A definition of the visibility skeleton is given in Section 2. We present a robust implementation based on a sweep-plane algorithm that was first presented in [17] for the case of pairwise disjoint polytopes and generalized to the case of possibly intersecting polytopes in [2]. We then present experiments on k disjoint polytopes of size n/k, with vertices on congruent spheres randomly distributed with fixed densities in a given (spherical) universe. These experiments show that the number of vertices of the visibility skeleton is roughly $Ck\sqrt{nk}$ where the observed constant C varies with scene density but remains small (less than 5 in our setting). Our experiments also indicate that the average running time of our implementation is $O(n^{3/2}k\log k)$. By contrast, the theoretical worst-case running time of the algorithm in our setting is $O(n^2k^2\log k)$.

These results are significant for three reasons. First, this is the first experimentally determined asymptotic estimate of the number of vertices of the 3D visibility skeleton

that takes into account not only the total number n of edges, but also the number k of polytopes in the scene. The results show that the size of the visibility skeleton may be sub-quadratic; in particular, they show a sub-linear growth in n and a sub-quadratic growth in k. Second, assuming that the size of the silhouette of a polytope on n/k vertices is $O(\sqrt{n/k})$, our results show that we may express the size of the visibility skeleton as a function that is linear in the size of the silhouette and quadratic in the number of polytopes; that is, the number of vertices in the scene impacts the size of the visibility skeleton only insofar as it increases the size of the silhouettes. Finally, our results indicate that there is no large constant hidden in the big-Oh notation.

Notice that if each polytope has constant complexity (*i.e.*, n/k in $\Theta(1)$), our experiments show a quadratic growth (in n or k). This latter observation is consistent with previous experiments [8, 14] in which the scenes consist of polygons of constant complexity. We recall, however, that for constant-size polytopes of bounded aspect ratio contained in randomly distributed disjoint congruent spheres, the expected number of visibility events (sufficiently inside the universe) is linear [6]. It is thus possible that our experiments did not treat scenes large enough to demonstrate the asymptotic behavior of the complexity; however, in such a case, our bound would be an overestimate.

In the next section, we give a definition of the visibility skeleton. In Section 3, we review the algorithm and discuss technical details of our implementation. We present our experimental results in Section 4 and conclude in Section 5.

2 The Visibility Skeleton of a Set of Polytopes

We start with some preliminary definitions. A *polytope* is the convex hull of a point set. Here, polytopes are assumed to have nonempty interior. A plane is *tangent* to a polytope if it intersects the polytope but not its interior. A line or segment is *tangent* to a polytope if it intersects the polytope and is contained in a tangent plane. A line or segment is *free* if it does not intersect the interior of any polytope. A free line segment is *maximal* if it is not properly contained in another free line segment.

A *support vertex* of a line is a polytope vertex that lies on the line. A *support edge* of a line is a polytope edge that intersects the line but has no endpoint on it (a support edge intersects the line at only one point of its relative interior). A *support* of a line is one of its support vertices or support edges. The supports of a segment are defined to be the supports of its relative interior; thus if a maximal free segment ends at a vertex of a polytope, this vertex is not a support.

The visibility complex of smooth disjoint objects is, roughly speaking, the partition of the space of maximal free line segments into connected components of segments that are tangent to the same objects [21]. For polytopes, each cell of the complex is further subdivided so that the corresponding maximal free line segments have the same set of supports. The visibility skeleton [8] is then naturally defined as the one-skeleton (*i.e.* the set of vertices and arcs) of the visibility complex.

In this paper, we study not the full one-skeleton but rather the skeleton containing only those arcs that correspond to local changes in the view, *i.e.*, arcs such that, when a viewpoint crosses the surface generated by the set of segments corresponding to the arc, a new polytope comes into view or a previously seen polytope disappears; in particular, we do not consider the appearance or disappearance of a polytope feature as a change

in the view. For pairwise disjoint convex smooth algebraic surfaces, it well known that these arcs consist of one-dimensional sets of maximal free line segments that are tangent to three objects (triple tangent events) or that are tangent to two objects in planes tangent to the two objects (tangent crossing events) [20]. This also holds for pairwise disjoint polytopes [5].[1]

With this in mind, we classify the arcs and vertices of the skeleton, in the spirit of Durand et al. [8], as follows. Unless stated otherwise, no two supports come from the same polytope. An arc is of type *EEE* if its set of supports consists of three edges; it is of type *EV* if its set of supports consists of an edge and a vertex that define a plane tangent to their respective polytopes. A vertex is of type *EEEE* if its set of supports consists of four edges; it is of type *VEE* if its set of supports consists of a vertex and two edges; it is of type *FEE* if its set of supports consists of two edges on one face, and two additional edges; it is of type *VV* if its set of supports consists of two vertices that lie on a plane that is tangent to their two respective polytopes.[2]

In this paper, we study the number of vertices of the visibility skeleton thus defined and refer to it, with abuse of notation, as the size of the visibility skeleton. Since, under our general position assumptions, the degree of each skeleton vertex is bounded by a constant, the actual size of the skeleton, including the arcs, will be a constant factor away from what we measure.

3 Algorithm and Implementation

In this section, we first give a brief overview of the sweep-plane algorithm for computing the vertices of the visibility skeleton and then give details about its implementation.

Algorithm. We give here a brief overview of the $O(n^2k^2 \log k)$ algorithm for computing the vertices of the visibility skeleton of k convex disjoint polytopes with n edges in total and in general position. (Discussion about the general position assumption can be found below.) This algorithm was presented in [17] for the case of pairwise disjoint polytopes[3] and then generalized to possibly intersecting polytopes in [2].

Given k convex disjoint polytopes in general position, that have n edges in total, the algorithm sweeps a plane about each edge e of each polytope in turn. The sweep plane is initially coplanar with one face incident to edge e and rotates about edge e until it becomes coplanar with the other face incident to e.

Initially, the sweep plane intersects the input polytopes in a set of polygons and the 2D visibility skeleton of these polygons is computed. This involves computing all

[1] Note that the visibility skeleton for smooth objects does not contain the arcs that correspond to tangent crossing events. On the other hand, the one-skeleton of the visibility complex of polytopes contains all arcs corresponding to visual events (as they appear as sets of segments tangent to two objects); however, it also contains many arcs that do not correspond to a local changes in the view (but only to the the appearance or disappearance of a polytope feature).

[2] This catalog is a subset of the one in [8] because Durand et al. essentially consider the one-skeleton of the visibility complex. They consider, in particular, line segments supported by two vertices which do not lie on a plane tangent to the two polytopes. This has an impact on the asymptotic size of the structure since the number of such vertices is presumably linear in the total complexity of the polytopes (in our experimental setting).

[3] Efrat et al. [11] presented a similar algorithm for computing not necessarily free isolated transversals in the same setting.

the *bitangents*, the maximal free line segments tangent to two polygons. Generically, a bitangent is tangent to two polygons in the sweep plane at two vertices. Each of these vertices lies on an edge of the input polytopes; we call these edges the *support edges* of the bitangent. We represent the 2D skeleton by storing, for each polygon, the list of bitangents supported by that polygon, in sorted order about the polygon.

During the sweep, we maintain the 2D visibility skeleton of the polygons intersected by the sweep plane. An event occurs whenever a bitangent appears or disappears, the support edges of a bitangent change, or when there is a change in the order of the bitangents around a polygon vertex. The *EEEE*, *VEE*, and *FEE* skeleton vertices arise from some of these events. (See [23] for a video on the algorithm.)

The worst-case running time of this algorithm is $O(nk^2 \log k)$ per sweep, that is $O(n^2k^2 \log k)$ in total.[4] Notice that the $\Theta(nk^2 \log k)$ worst-case bound for one sweep can be quite pessimistic: the time complexity of each sweep is, modulo the logarithmic factor, proportional to the complexity of the 2D visibility skeleton over the whole sweep, that is, to the number of combinatorially distinct bitangents occurring during the sweep. In particular, the time complexity of one sweep can be sub-linear in n. Note that this differs from the algorithms presented in both [11] and [2] which maintain all line segments tangent to two polygons in the sweep plane, even if they are not free.

Implementation. While our ultimate objective is a robust implementation that correctly computes the visibility skeleton vertices on a set of polyhedra in arbitrary position in a reasonable amount of time, our current implementation takes as input any set of convex polyhedra and either outputs the skeleton vertices or reports that the polytopes are not in general position (see below). We implemented the algorithm in C++ using the *CGAL* library [3] with the 2D visibility skeleton package due to Angelier and Pocchiola [1].

Predicates. Several predicates are required by the algorithm including, for example, determining whether four segments admit a line transversal. As in all sweep algorithms, an essential predicate is one that compares two positions of the sweep plane. The algebraic degree of some of these predicates is quite high; in particular, comparing two positions of the sweep plane is implemented with a procedure of degree 168 in the Cartesian coordinates of the input vertices. See [12] for details.

Number type. Our implementation follows the paradigm of exact computation; we have implemented all predicates using the `Filtered_exact` number type of CGAL (3.2.1) templated with CGAL interval arithmetic (based on `double` number type) and the CORE library [4]. This means that the predicates are first evaluated using interval arithmetic, and only when this fails are they evaluated using the CORE exact number type. Using filtered exact computation ensures that no predicate is ever incorrectly evaluated; however, this comes at a cost: on random inputs, the computation is no more than four times slower than when using the `double` number type.

General position assumption. By polytopes in general position we mean that no predicate evaluates to zero.[5] In particular, this guarantees that each event corresponds to a

[4] Note that both [17] and [2] report algorithms having time complexity $O(n^2k^2 \log n)$ instead of $O(n^2k^2 \log k)$; the reason for this is that, in [17], the skeleton is reconstructed which is not the case here and, in [2], the event queue may be of size $\Theta(n)$ because the polytopes may intersect.

[5] Note that filtered-exact arithmetic is still needed under this general position assumption since predicates could still be evaluated incorrectly with a fixed-precision floating-point arithmetic.

unique position of the sweep plane.[6] It also implies more familiar assumptions such as that no two polytope faces are coplanar; see [2] for more details. To the best of our knowledge, there exists no implementation of the 3D visibility skeleton that handles degeneracies, including the implementation of Durand in which degeneracies were avoided by perturbing the input scenes by hand (Duguet [7] proposed a method for handling degeneracies but, as previously noted, only for computing a section of the visibility skeleton, that is a set of maximal free line segments that are supported by concurrent lines.) Our code represents an improvement in the sense that we systematically detect all degeneracies although the code to handle them remains unwritten.

Software validation. We verified the correctness of our implementation by comparing its output with that of an implementation of the brute force algorithm, the latter being straightforward, having only about a thousand lines of code. We ran tests on twenty input scenes of up to 100 polytopes with up to 1 000 edges and obtained the same results for both implementations, that is, the same list of vertices.

4 Experiments

The model. The input scenes are generated in two phases. First, we generate a set of disjoint spheres and, in phase two, we generate one convex polytope in each sphere.

In phase one, for a given number of spheres k and scene density μ, we generate k unit spheres in a spherical universe of center O and radius R, where $R = \sqrt[3]{k/\mu}$ (that is, $\mu = k\frac{4}{3}\pi/\frac{4}{3}\pi R^3$). The centers of the spheres are chosen, one by one, uniformly from the ball centered at O of radius $R-1$. When a newly generated sphere intersects any existing one, the new sphere is discarded. Note that the spheres are not uniformly distributed since the new sphere is not chosen independently of the previous ones.

In phase two, for each sphere, a set of vertices is generated uniformly on the surface of the sphere and their convex hull is computed. This defines one polytope for each sphere and guarantees that they are disjoint. Notice that the density of the polytopes in the scene is somewhat less than the density μ of the spheres of phase one.

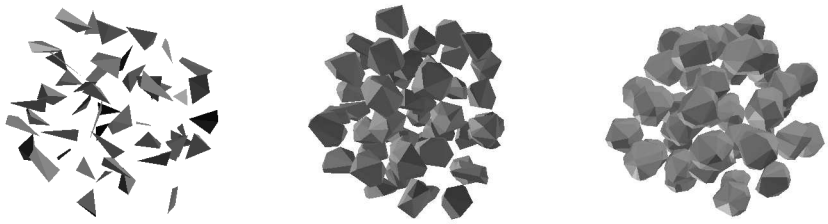
We emphasize that our objective is not to study uniformly distributed disjoint polytopes approximating spheres. We have chosen this scene model because it provides a simple way to generate large scenes containing disjoint polytopes. Furthermore, it allows us to compare our results with the theoretical results of [6].

The experiments. We consider scenes of polytopes, as defined above, depending on three parameters, the number k of polytopes, the total number n of polytope edges, and the scene density μ. We perform three suites of experiments in which we measure the number of visibility skeleton vertices.

In Suite I, we fix the scene density μ and the number n/k of edges per polytope. For different values of k, we generate scenes of k polytopes each having n/k edges.

It appears that, even in the random setting of our experiments, predicates are evaluated incorrectly about 0.1% of the times when using the `double` number type.

[6] Actually, this guarantees that no two *unrelated* events correspond to the same position of the sweep plane. The situation where three bitangents become aligned induces three events corresponding to an alignment of any two of these bitangents; these three events are in the same sweep plane yet the situation is generic.



(a) (b) (c)

Fig. 1. Three sample scenes with scene density $\mu = 0.3$ and $k = 50$ polytopes whose number of edges, n/k, is approximately equal to (a) 6, (b) 42, and (c) 84.

We perform experiments for $\mu = 0.3, 0.05$ and 0.01 and for $n/k \approx 6, 42$ and 84.[7] (A sample scene with $k = 50$ is shown in Fig. 1.) For each value of n/k, we vary the number k of polytopes as follows: (a) when $n/k \approx 6$, we vary k from 10 to 190 (giving $n \in [75, 1\,425]$), (b) when $n/k \approx 42$, we vary k from 10 to 130 (giving $n \in [400, 5\,200]$), and (c) when $n/k \approx 84$, k varies from 10 to 110 (giving $n \in [850, 9\,400]$). As we will see, the number of visibility skeleton vertices appears to be roughly $C_\mu k\sqrt{nk}$ in these experiments where C_μ is a constant that depends on the density.

In Suite II of our experiments, we also fix the scene density μ to 0.3 and vary the number n/k of edges per polytope for fixed numbers of polytopes. Namely, we consider $k = 30, 60,$ and 90 and vary n/k from 6 to 102. As we will see, these experiments confirm that when n/k varies (in the given range), the complexity observed in the first set of experiments holds.

Note that a scene with density $\mu = 0.3$ is very dense (see Fig. 1 and recall Kepler's Theorem that the density of any sphere packing in 3D space is at most $\pi/3\sqrt{2} \approx 0.74$). Density $\mu = 0.3$ is close to the highest density we can reach in a reasonable amount of time with our scene generation scheme.

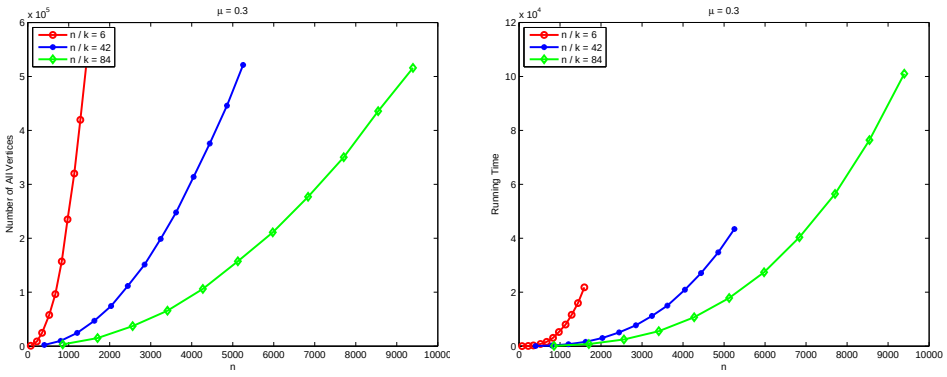
Machine characteristics. All the experiments were done on an *i*686 machine with a *Pentium* 2.80 GHz CPU running Linux, with 2 GB of main memory. Running time was measured with the *getrusage()* command and the *ru_utime* attribute.

4.1 Experimental Results and Analysis

Number of skeleton vertices in terms of n. We present, in terms of the total number n of edges in the scene, the results of Suite I for density $\mu = 0.3$. For $n/k \approx 6, 42$ and 84, respectively, Fig. 2(a) shows the total number of visibility skeleton vertices; Fig. 2(b) shows the running time of our implementation in seconds.

For a given value of n/k, the number of skeleton vertices appears to be quadratic in n (see Figs. 2(a) and 3(a)) and the running time appears to be in $\Theta(n^{2.5}\log n)$ (see Figs. 2(b) and 4(b)). Notice also that for a fixed n, both the running time and the size of the output drop drastically when the number k of polytopes decreases (see Fig. 2). These observations are consistent with the theoretical bounds; the worst-case time complexity of the algorithm is in $O(n^2k^2\log k)$ and the worst-case output size is in $\Theta(n^2k^2)$ [2]. The rest of this section analyses the running time and output size in terms of n and k.

[7] In fact, we generate polytopes whose numbers of vertices range in $[4, 6]$, $[15, 17]$ and $[30, 32]$, respectively. The number of edges per polytope is thus not actually fixed but varies slightly; the polytopes we generated have, on average, 7.5, 40, and 85 edges, respectively.



(a) (b)

Fig. 2. Suite I ($\mu = 0.3$): (a) total number of skeleton vertices and (b) running time (in seconds; 2.88×10^4 seconds = 8 hours) in terms of n, the number of edges in the scene.

Number of skeleton vertices in terms of n and k. Fig. 3 shows the number of skeleton vertices in terms of $k\sqrt{nk} = k^2\sqrt{n/k}$. The number of these skeleton vertices appears to be linear in this parameter with a constant that depends on the scene density μ. For $\mu = 0.3, 0.05$, and 0.01, the constant is roughly 5, 4, and 3. The constant appears to decrease in terms of μ which is consistent with the intuition that the constant goes to zero as the scene density goes to zero (since the probability that there exists a line transversal to three polytopes goes to zero as the density goes to zero and that the number of vertices of type *VV* is asymptotically negligible).

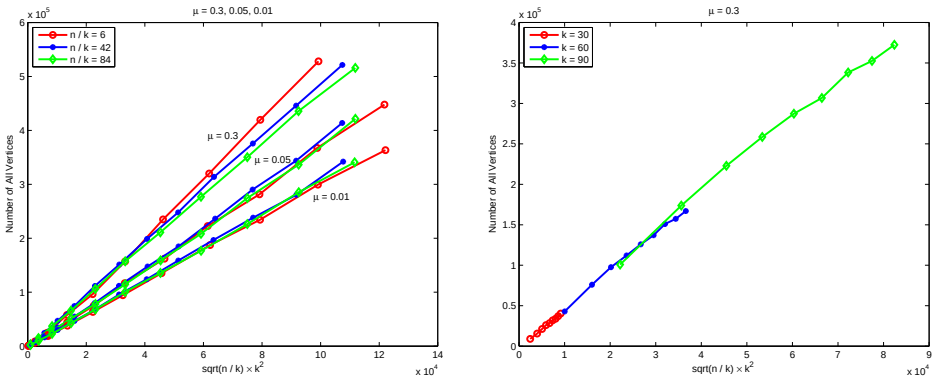
Note that, for any fixed density μ and any given value of $k^2\sqrt{n/k}$, the number of these skeleton vertices varies very little in terms of the polytope complexity n/k (Fig. 3(a)) and in terms of the number of polytopes (Fig. 3(b)). This suggests that $C_\mu k^2\sqrt{n/k}$ is a good predictor of the number of these skeleton vertices regardless of the polytope complexity, at least for the scene density μ and the ranges of n/k used here.

Our experiments thus indicate that, in our setting, the number of skeleton vertices is roughly $C_\mu k^2\sqrt{n/k}$, where C_μ is a constant that depends on the density μ of the scene. The experiments hint that this constant is small and is a decreasing function of μ.

This observed complexity is, as expected, much smaller than worst-case bounds. Recall that, for k polytopes with n edges in total, the worst-case number of skeleton vertices is $\Theta(n^2k^2)$ [2]. Also, if the silhouettes of the polytopes have size $\sqrt{n/k}$ in the worst case, the worst-case number of skeleton vertices is $O(nk^3\sqrt{n/k})$ [15, §6.7]. These worst-case bounds are much larger than our observed size (by a factor $n\sqrt{nk}$ and nk).

We analyze below the observed complexity of $C_\mu k^2\sqrt{n/k}$ in terms of (i) k when the complexity of the polytopes is constant, and (ii) the silhouette size of the polytopes when the number k of polytopes is constant.

Analysis of the number of skeleton vertices in terms of k. If each polytope has constant complexity (*i.e.*, n/k in $\Theta(1)$), our experiments exhibit a quadratic growth (in terms of k) of the number of skeleton vertices. This is consistent with previous experiments [8, 14] in which the scenes consist of polygons of constant complexity and is also consistent with the best known theoretical *expected* upper bound of $O(k^2)$ [6] corresponding to our setting. However, this contradicts the intuitive linear bound of $\Theta(k)$ when n/k is



(a) (b)

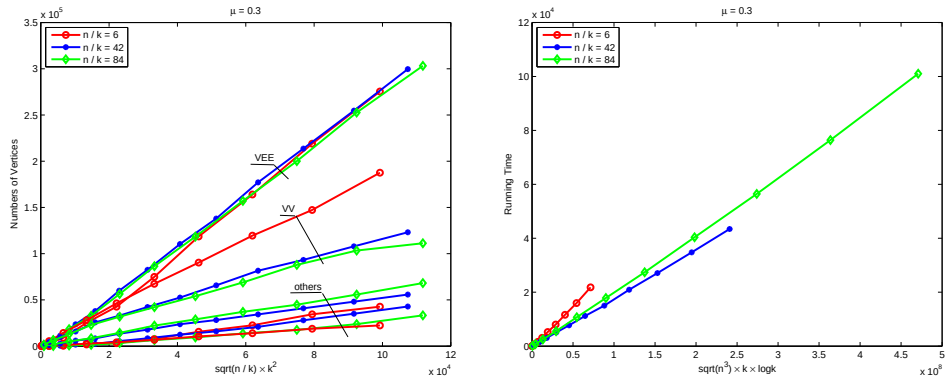
Fig. 3. Total number of skeleton vertices in terms of $k^2\sqrt{n/k}$ when (a) the polytopes have a constant (n/k) number of edges (Suite I) and (b) the number k of polytopes is constant (Suite II).

constant; indeed, recall that for k randomly distributed congruent spheres, the expected number of visibility events is linear and, for constant-size polytopes of bounded aspect ratio inside such spheres, the expected number of visibility events is linear for events that occur sufficiently inside the universe but it is only upper bounded by $O(k^2)$ for events near the boundary of the universe [6]. It is possible that the expected upper bound of $O(k^2)$ is tight but it is also possible that our experiments did not reach the asymptotic behavior of the complexity. If this is the case, it is then reasonable to believe that our experimental estimate of the complexity is an overestimate.

Analysis of the number of skeleton vertices in terms of the silhouette size of the input polytopes. If we fix the number k of polytopes and vary the total number n of edges, our experiments show that the number of skeleton vertices depends linearly on $\sqrt{n/k}$. We argue below that this means that, in our setting, when k is fixed, the number of skeleton vertices depends linearly on the silhouette size of the input polytopes and explain why.

Recall that, for any polyhedron of size $\Theta(m)$, the size of its silhouette viewed from a random point is $O(\sqrt{m})$ under some reasonable hypotheses [16] (see also [18] for the special case of polyhedra that approximate spheres). Since the vertices of the polytopes we consider are randomly distributed on a sphere, it is reasonable to assume that the size of the silhouette does not depend much on the choice of the viewpoint. In other words, for any polytope with n/k edges we consider, it is reasonable to assume that its silhouette has size $O(\sqrt{n/k})$ from any viewpoint. Hence, when k is fixed, the number of skeleton vertices depends linearly on the silhouette size of the input polytopes.

We offer the following intuitive explanation of this observation. Consider the arcs of type *EEE* of the skeleton. The endpoints of these arcs are vertices of type *VEE*, *FEE* or *EEEE*. When the number k of polytopes is fixed and the number n of edges tends to infinity, the polytopes tend to spheres and the segments corresponding to vertices of type *EEEE* converge to segments that are tangent to four spheres; hence, in our setting, the number of *EEEE* vertices converges to a constant. Similarly, for those VEE vertices that correspond to intersections of two arcs of types *VE* and *EEE* (thus corresponding to segments tangent to three polytopes while lying in planes that are tangent to two of them). Moreover, in the successive refinements of polytopes as n increases, each



(a) (b)

Fig. 4. Suite I (k polytopes having a constant, n/k, number of edges; $\mu = 0.3$): (a) proportion of number of vertices in terms of $k^2\sqrt{n/k}$ and (b) running time (in seconds) in terms of $n^{1.5}k\log k$.

EEE arc incident to an *EEEE* vertex or one of the above *VEE* vertices will become a sequence of *EEE* arcs joined at *VEE* vertices (that is, subdivision vertices such that the sets of supports are invariant along the subdivided arcs). For such a sequence of arcs, the number of these *VEE* vertices is the number of polytope vertices encountered by a maximal free line segment as it slides from the segment corresponding to one end of the sequence to the other, while remaining tangent to the three polytopes (nearly spheres) involved. The number of such polytope vertices is, intuitively, at most the worst-case size of the silhouette of each polytope, which we have assumed to be in $O(\sqrt{n/k})$. As a polytope gets more complex and tends to a sphere, the subset of lines in the line space that are tangent to the polytope on its vertices tends towards the subset of lines that are tangent to the sphere. This is also the case for lines tangent to the polytope on its faces. For this reason, the number of *FEE* vertices is asymptotically the same as that of *VEE* vertices. Thus, intuitively, we can expect that, for fixed k, (i) the number of vertices of type *EEEE* converges to a constant as n goes to infinity, (ii) the number of vertices of type *VEE* or *FEE* is in $O(\sqrt{n/k})$ times the number of *EEEE* vertices, and thus that (iii) the number of *VEE* and *FEE* vertices is in $O(\sqrt{n})$ (for k fixed). In experiments not reported here, we have observed (ii) and (iii), but not (i). In Figure 4(a), the number of *VEE* vertices is much larger than the number of *EEEE* vertices, which is consistant with the previous discussion, while the convergence of *EEEE* is not seen in our experimental range.

Finally, the number of *VV* vertices is, intuitively, bounded by the product of the number of pairs of polytopes that are mutually visible (which is asymptotically linear in k for any given density) and the size of the polytope silhouettes. Hence the number of *VV* vertices is presumably in $\Theta(k\sqrt{n/k})$. Note that in our experiments (Fig. 4(a)) we observe a complexity of roughly $\Theta(k^2\sqrt{n/k})$ because the number of visible pairs of polytopes is quadratic in k in our experimental range.

Running time in terms of n and k. We study the running time of our implementation in terms of $n\sqrt{n}k\log k$ for experiments in Suite I, and show the results for scene density $\mu = 0.3$ in Fig. 4(b) (our results for other densities are omitted here). We observe that for a fixed polytope complexity n/k, the running time seems linear in $n\sqrt{n}k\log k$. More

precisely, we observe a running time of $C'_\mu n\sqrt{n}k\log k$ seconds with C'_μ no more than $3\cdot 10^{-4}$ for the considered densities. Note, however, that for density 0.3, the data we obtained from groups $n/k \approx 42$ and 84 fit the estimated time complexity well, whereas the data from the group $n/k \approx 6$ is a constant factor away.

The observed running time can be intuitively explained as follows. Note first that $n\sqrt{n}k\log k$ is equal to $n\ \sqrt{n/k}\ k\sqrt{k}\ \log k$. We dissect this expression as follows. First, n is the number of sweeps performed by the algorithm. We observe that the factor $k\sqrt{k}$ is linearly related to the average over all sweeps of the maximal number of bitangents encountered in the sweep plane during a sweep (we omit here the presentation of these experiments). This is reasonable since the number of bitangents in the sweep plane varies from $\Theta(k^2)$, the trivial worst-case bound, to $\Theta(k)$, the expected bound in the right setting [13]. Furthermore, the factor $\sqrt{n/k}$ naturally relates to the number of updates caused by each bitangent during the sweep; indeed, following a bitangent during a sweep, the bitangent will encounter vertices on each of the two polytopes supporting it; the number of these vertices on each polytope is related and, intuitively, is less than the worst-case size of the silhouettes of the polytopes which, as we argued above, is in $O(\sqrt{n/k})$ in our setting. Finally, $\log k$ is the complexity of each update of the event list.

5 Conclusion

We have presented here an implementation of the sweep-plane algorithm to compute the visibility skeleton. Our experiments suggest that, in our setting, the number of vertices of the 3D visibility skeleton is $C_\mu k\sqrt{nk}$ and is dominated by type *VEE* vertices. The constant C_μ, which depends on the scene density, is no more than 5 for n and k in our experimental range, and for the various densities that we studied.

This is the first prediction of the actual size of the 3D visibility skeleton for reasonably large n, and expressed in terms of both n and k. Assuming that the size of the silhouette of a polytope with n/k edges is $O(\sqrt{n/k})$, our results show that the size of the visibility skeleton is linear in the size of the silhouette and quadratic in the number of polytopes. Surprisingly, the constant C_μ is rather small; this indicates that there is no large constant hidden in the big-Oh notation.

The experiments also suggest that the expected running time of our implementation of the sweep plane algorithm is $C'_\mu n\sqrt{n}k\log k$ seconds, where C'_μ depends on the scene density but is, on our machine, no more than $3\cdot 10^{-4}$ for the considered densities.

Our results indicate that the visibility skeleton is of reasonable size and can be computed exactly in a reasonable length of time. Further work includes completing the implementation for degenerate situations. A major challenge is to extend the sweep algorithm to handle general polyhedra.

References

1. P. Angelier and M. Pocchiola. CGAL-based implementation of visibility complexes. Technical Report ECG-TR-241207-01, Effective Computational Geometry for Curves and Surfaces (ECG), 2003.
2. H. Brönnimann, O. Devillers, V. Dujmovic, H. Everett, M. Glisse, X. Goaoc, S. Lazard, H.-S. Na, and S. Whitesides. Lines and free line segments tangent to arbitrary three-dimensional convex polyhedra. *SIAM Journal on Computing*, 37(2):522–551, 2007.

3. CGAL: Computational Geometry Algorithms Library. http://www.cgal.org.
4. The CORE library. http://cs.nyu.edu/exact/.
5. J. Demouth. *Événements visuels et limites d'ombres*. PhD thesis, Université Nancy 2, 2008. To appear.
6. O. Devillers, V. Dujmović, H. Everett, X. Goaoc, S. Lazard, H.-S. Na, and S. Petitjean. The expected number of 3D visibility events is linear. *SIAM Journal on Computing*, 32(6):1586–1620, 2003.
7. F. Duguet and G. Drettakis. Robust epsilon visibility. In John Hughes, editor, *Proceedings of ACM SIGGRAPH 2002*, pages 567–575. ACM Press / ACM SIGGRAPH, July 2002.
8. F. Durand, G. Drettakis, and C. Puech. The visibility skeleton: a powerful and efficient multi-purpose global visibility tool. *Computer Graphics Proceedings, Annual Conference Series*, 31:89–100, 1997. Proceedings of Siggraph'97.
9. F. Durand, G. Drettakis, and C. Puech. Fast and accurate hierarchical radiosity using global visibility. *ACM Transactions on Graphics.*, 18(2):128–170, 1999.
10. F. Durand, G. Drettakis, and C. Puech. The 3D visibility complex. *ACM Transactions on Graphics*, 21(2):176–206, 2002.
11. A. Efrat, L. Guibas, O. Hall-Holt, and L. Zhang. On incremental rendering of silhouette maps of a polyhedral scene. *Comput. Geom.: Theory and App.*, 38(3):129–138, 2007.
12. H. Everett, S. Lazard, B. Lenhart, and L. Zhang. On the degree of standard geometric predicates for line transversals in 3D. *Comput. Geom.: Theory and App.*, 2008. To appear.
13. H. Everett, S. Lazard, S. Petitjean, and L. Zhang. On the expected size of the 2D visibility complex. *Int. J. of Comput. Geom. and App.*, 17(4):361–382, 2007.
14. L. Glaves. An exploration of the 3D visibility complex. Master's thesis, Polytechnic University, Brooklyn, NY., 2007.
15. M. Glisse. *Combinatoire des droites et segments pour la visibilité 3D*. PhD thesis, Université Nancy 2, Oct. 2007.
16. M. Glisse and S. Lazard. An upper bound on the average size of silhouettes. *Discrete and Computational Geometry*, 2008. To appear.
17. X. Goaoc. *Structures de visibilité globales : tailles, calculs et dégénérescences*. PhD thesis, Université Nancy 2, May 2004.
18. L. Kettner and E. Welzl. Contour edge analysis for polyhedron projections. In W. Strasser, R. Klein, and R. Rau, editors, *Geometric Modeling: Theory and Practice*, pages 379–394. Springer, 1997.
19. H. Plantinga and C. Dyer. Visibility, occlusion, and the aspect graph. *International Journal of Computer Vision*, 5(2):137–160, 1990.
20. O. A. Platonova. Singularities of relative position of a surface and a line. *Uspekhi Mat. Nauk*, 36(1):221–222, 1981. Also in *Russian Math. Surveys* 36(1):248-249, 1981.
21. M. Pocchiola and G. Vegter. The visibility complex. *Int. J. of Comput. Geom. and App.*, 6(3):279–308, 1996.
22. A. Schröder. *Globale Sichtbarkeitsalgorithmend*. PhD thesis, Philipps-Universität Marburg, Jun. 2003.
23. L. Zhang, H. Everett, S. Lazard, and S. Whitesides. Towards an implementation of the 3D visibility skeleton. In Proceedings of the *23rd ACM Annual Symposium on Computational Geometry (SoCG'07)*, pages 131–132, S. Korea, 2007. Video.