Computing Elevation Maxima by Searching the Gauss Sphere

Bei Wang, SCI Institute, University of Utah
Herbert Edelsbrunner, Duke University, Geomagic and IST Austria
Dmitriy Morozov, Stanford University

The elevation function on a smoothly embedded 2-manifold in $\mathbb{R}^3$ reflects the multiscale topography of cavities and protrusions as local maxima. The function has been useful in identifying coarse docking configurations for protein pairs. Transporting the concept from the smooth to the piecewise linear category, this paper describes an algorithm for finding all local maxima. While its worst-case running time is the same as of the algorithm used in prior work, its performance in practice is orders of magnitudes superior. We cast light on this improvement by relating the running time to the total absolute Gaussian curvature of the 2-manifold.

Categories and Subject Descriptors: F.2.2. [**Nonnumerical Algorithms and Problems**]: Geometrical problems and computations

General Terms: Design, Algorithms, Performance, Experimentation

Additional Key Words and Phrases: Elevation function, persistent homology, Gauss map, absolute Gaussian curvature, algorithms, protein surfaces, computational experiments

ACM Reference Format:
Wang, B., Edelsbrunner, H., Morozov, D., 2011. Computing Elevation Maxima by Searching the Gauss Sphere. ACM J. Exp. Algor. 9, 4, Article 39 (March 2010), 12 pages.
DOI = 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

1. INTRODUCTION

This paper introduces a new algorithm for computing all local maxima of the elevation function defined on a 2-manifold embedded in $\mathbb{R}^3$. This function has been introduced by Agarwal et al. [Agarwal et al. 2006] for the purpose of improving the prediction of protein interaction through docking. The approach identifies protrusions (knobs) and cavities (wells) on the two surfaces and matches them up. This idea goes back to Connolly [Connolly. 1986] who used a function that maps each point of the protein surface to the fraction of a fixed-radius sphere centered at the point that lies outside the protein volume. As shown by Cazals et al. [Cazals et al. 2003], this function resembles the mean curvature at the point in the limit, when the radius approaches zero. The fixed radius makes a choice of the scale the function reflects.

The elevation function introduced in [Agarwal et al. 2006] serves the same purpose, but in contrast to Connolly's function, the elevation is scale independent and marks small as well as large protrusions of varying shape and direction. Its construction is based on the persistence structure of the 2-parameter family of height functions, as explained in the next section. The task at hand is then the computation of all local maxima for two proteins and the use of the type, size, and location of the marked topographic features to identify promising positions for interaction. The experimental study in [Wang et al. 2005] shows that this approach is effective in finding initial positions that can

This research is partially supported by the Defense Advanced Research Projects Agency (DARPA) under grants HR0011-05-1-0007 and HR0011-05-1-0057. Most of the work was done while the authors were at Duke University.
Author's addresses: B. Wang, SCI Institute, University of Utah; D. Morozov, Computer Science and Math Department, Stanford University; H. Edelsbrunner, Computer Science Department, Duke University, Geomagic and IST Austria.

DOI 10.1145/0000000.0000000 http://doi.acm.org/10.1145/0000000.0000000

then be refined by local optimization. The computationally most expensive step in this study is the determination of the elevation maxima. Using the algorithm in [Agarwal et al. 2006], the running time for a triangulated 2-manifold with m edges is proportional to $m^5 \log_2 m$. Since typical proteins give rise to surfaces with hundreds of thousands of edges, the quintic dependence on m is a serious drawback that limits the practical deployment of the method.

In this paper, we give a new algorithm that is faster for triangulated surfaces approximating smooth surfaces that we typically find in practice. They are characterized by having dihedral angles at edges that are close to half the full angle (molecular skin surface [Edelsbrunner 1999]). We relate the running time of our algorithm to the total absolute Gaussian curvature of the surface and this way determine that we can expect roughly a ten-thousand fold improvement over the running time of the old algorithm. We note, however, that we offer no improvement in the worst-case performance.

Since we incorporate the surface complexity in terms of total absolute Gaussian curvature into the analysis of the algorithm, it is worth mentioning that there is a large literature on the notion of curvatures for triangulated surfaces. We refer to [Banchoff 1970] and [Morvan 2008; Cohen-Steiner and Morvan 2006] for details.

Outline. In Section 2, we introduce the geometric and topological background needed to understand the elevation function. We do this in two steps, discussing the mathematically cleaner smooth case in Section 2.1 and the computationally more useful piecewise linear (PL) case in Section 2.2. In Section 3, we present the algorithm for computing all elevation maxima, along with some implementation details and the analysis. In Section 4, we present our experimental results, employing our software to compute elevation maxima for a number of triangulated protein surfaces. We gather statistics on critical regions, pairwise intersections, and elevation maxima. We use these statistics as evidence that our assumption is a reasonable approximation of the reality for our data and that the new algorithm runs about four orders of magnitude faster than the old one.

2. PRELIMINARIES

In this section, we introduce the necessary background for understanding the elevation function and our algorithm for computing its local maxima. We begin with the smooth case, which we use as guiding intuition in our subsequent treatment of the piecewise linear case.

2.1. The Smooth Case

We begin with brief introduction of Morse functions and persistent homology, then use these concepts to define the elevation function. Finally, we discuss the Gaussian curvature of the 2-manifold.

Morse functions. The class of smooth, real-valued functions is a challenging object that simplifies considerably if we add genericity as a requirement. Letting $f : \mathbb{M} \to \mathbb{R}$ be a smooth function on a 2-manifold, a point $x \in \mathbb{M}$ is *critical* if the derivative at x equals zero. The value of f at a critical point is a *critical value*. All other points are *regular points* and all other values are *regular values* of f. A critical point is *non-degenerate* if the Hessian, that is, the matrix of second partial derivatives at the point is invertible. In this case, the matrix has two non-zero eigenvalues, $\lambda_1 \neq \lambda_2$, and the *index* of the non-degenerate critical point is the number of negative eigenvalues. A non-degenerate critical point of index 0 is a *minimum*, of index 1 is a *saddle*, and of index 2 is a *maximum*, see Figure 1. Finally, f is a *Morse function* if all its critical points are non-degenerate and its values at the critical points are distinct. Given a value $a \in \mathbb{R}$, the corresponding *sublevel set* consists of all points with value at most a, $\mathbb{M}_a = f^{-1}(-\infty, a]$. Sweeping the manifold in the direction of increasing function value, we get a 1-parameter family of sublevel sets. The topology of the sublevel set changes precisely when the sweep passes through a critical point. Let $t_1 < t_2 < ... < t_n$ be the ordered sequence of critical values and $-\infty = s_0 < s_1 < ... < s_n = \infty$ a sequence of interleaved values, that is, $s_i < t_{i+1} < s_{i+1}$, for all i. By assumption of f being Morse, we get from the sublevel set at s_i to the one at s_{i+1} by passing exactly one non-degenerate critical point. The change can be characterized in terms of the dimension of the handle we attach to go from $\mathbb{M}_{s_i}$ to $\mathbb{M}_{s_{i+1}}$.

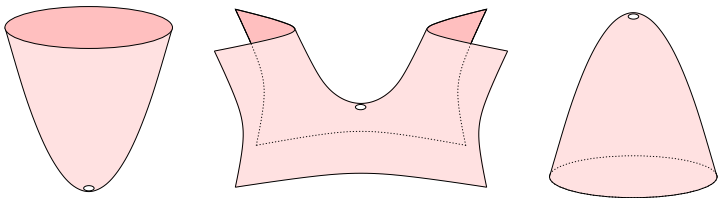

Fig. 1: From left to right: a minimum, a saddle, and a maximum.

For index 0, we add a 0-handle, that is, an isolated point which we then thicken to a disk. For index 1, we add a 1-handle, that is an interval attached to the boundary of the sublevel set at its endpoints which we then thicken to a strip. Finally, for index 2, we add a 2-handle, that is, a disk attached to the boundary of the sublevel set along its boundary circle.

Persistent homology. Looking at the homology groups [Munkres 1984] of the sequence of sublevel sets, we use the concept of persistence to measure the lengths of the intervals along which homology classes exist [Edelsbrunner et al. 2002]. Since sublevel sets between two contiguous critical values are indistinguishable, we may consider the finite sequence

$$\emptyset = \mathbb{M}_0 \subseteq \mathbb{M}_1 \subseteq \ldots \subseteq \mathbb{M}_n = \mathbb{M},$$

where we simplify notation by setting $\mathbb{M}_i = \mathbb{M}_{s_i}$. Fixing a dimension p ($p \geq 0$), each sublevel set has a p-th homology group and the sequence is connected from left to right by homomorphisms induced by inclusion, which we denote as $f_p^{i,j} : \mathsf{H}_p(\mathbb{M}_i) \to \mathsf{H}_p(\mathbb{M}_j)$. We have a *birth* at $\mathbb{M}_i$ if the map $f_p^{i-1,i}$ is not surjective, and we have a *death* at $\mathbb{M}_j$ if the map $f_p^{j-1,j}$ is not injective. Furthermore, the death at $\mathbb{M}_j$ corresponds to the birth at $\mathbb{M}_i$ if there is homology class γ in $\mathsf{H}_p(\mathbb{M}_i)$ that is not in the image of $f_p^{i-1,i}$, its image in $\mathsf{H}_p(\mathbb{M}_{j-1})$ is still not in the image of $f_p^{i-1,j-1}$, but its image in $\mathsf{H}_p(\mathbb{M}_j)$ is in the image of $f_p^{i-1,j}$. We call $f(t_j) - f(t_i)$ the *persistence* of this birth-death pair. As explained in [Cohen-Steiner et al. 2007], this method gives a pairing between births and deaths that has many interesting properties. Each death corresponds to a unique birth but not every birth corresponds to a death. To remedy this shortcoming, we extend the sequence of homology groups for extended persistence as described in [Cohen-Steiner et al. 2009]. Writing $\mathbb{M}^a = f^{-1}[a, \infty)$ for the *superlevel set* of a, we go up with absolute homology groups of sublevel sets, as before, and we come back down with relative homology groups,

$$\begin{aligned} 0 = \mathsf{H}_p(\mathbb{M}_0) \to \mathsf{H}_p(\mathbb{M}_1) \to \ldots \to \mathsf{H}_p(\mathbb{M}_n) \\ \to \mathsf{H}_p(\mathbb{M}, \mathbb{M}^n) \to \ldots \to \mathsf{H}_p(\mathbb{M}, \mathbb{M}^0) = 0, \end{aligned}$$

where we simplify notation by setting $\mathbb{M}^i = \mathbb{M}^{s_i}$, $\mathbb{M}^0 = \mathbb{M}$ and $\mathbb{M}^n = \emptyset$. Now every birth corresponds to a death. In fact, we have two events at every critical point, one going up and one coming down, but duality implies that we just get each pair twice, see [Cohen-Steiner et al. 2009]. As a consequence of duality, the birth-death pairs we get for the negative function, $-f$, are the same. This turns out to be important in the definition of the elevation function.

For 2-manifolds, there is a more elementary way to introduce extended persistence using the Reeb graph of the function. Instead of giving details, we refer to [Agarwal et al. 2006] and we mention that this approach leads to a fast algorithm. It consists of constructing the Reeb graph in a sweep [Cole-McLaughlin et al. 2004] followed by deconstructing it in another sweep using cutting and linking trees [Agarwal et al. 2006; Georgiadis et al. 2006]. We run this algorithm for a piecewise linear function on a triangulated 2-manifold. Letting m be the number of edges in the triangulation, as before, the running time computing the extended persistence for a given height function is bounded by some constant times $m \log_2 m$.

Elevation. To define elevation, we assume the 2-manifold $\mathbb{M}$ is smoothly embedded in $\mathbb{R}^3$. For a direction $u \in \mathbb{S}^2$, we consider the height function $h_u : \mathbb{M} \to \mathbb{R}$ defined by $h_u(x) = \langle x, u \rangle$. Generically, h_u is a Morse function, but for some directions u it is not, either because a critical point is degenerate or because two or more critical points map to the same height value. Considering the entire sphere of directions, we get a 2-parameter family of height functions.

For each $u \in \mathbb{S}^2$, we pair up births with deaths using the extended sequence of homology groups defined by the sublevel and the superlevel sets of h_u. In the Morse function case, each birth-death pair identifies two critical points, x and y, one giving birth and the other giving death, and we define the elevation at these two points as their persistence or, equivalently, the absolute height difference in the direction u, $E(x) = E(y) = |h_u(x) - h_u(y)|$. Each point of $\mathbb{M}$ is critical in two directions, u and $-u$, and is thus assigned two values, the absolute height difference to the paired critical point in the two directions. Since $h_{-u} = -h_u$, the paired point is the same so we get a unique value at every point. This is the *elevation function* of the 2-manifold, $E : \mathbb{M} \to \mathbb{R}$.

To get a feeling for this function, we consider a protrusion (a mountain) of the 2-manifold. To measure the height of the mountain, we measure from the top down, to the first saddle that separates it from an even higher mountain. We can do this in various directions, so we do it to maximize the height. This might be in a direction along which the first saddle is ambiguous. Perhaps there are three such saddles at the same height value in this direction, similar to the third type in Figure 2 in which we have a saddle with the same height difference to three minima. In this direction, we have two violations of genericity required for Morse functions, because there are three critical points with the same height value. Indeed, local maxima of E tend to arise along non-generic directions. An exception is the 1-legged maximum defined by only two critical points (with one leg between them). Besides this case, we have 2-legged maxima defined by three critical points, and 3- and 4-legged maxima defined by four critical points each; see Figure 2.

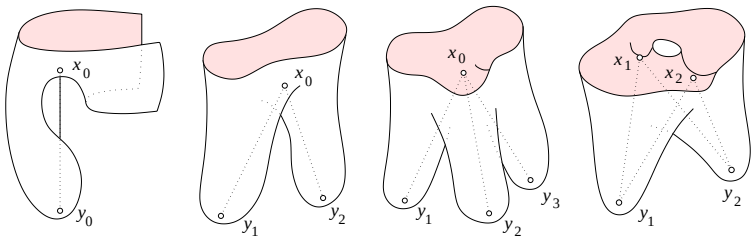

Fig. 2: The four generic types of local maxima of the elevation function. From left to right: the 1-, 2-, 3- and 4-legged maximum.

Curvature. We will later discover that the running time of our algorithm for finding all local maxima relates to the total absolute curvature of the surface. We introduce this concept using the *Gauss map*, $N : \mathbb{M} \to \mathbb{S}^2$, defined by mapping a point x of $\mathbb{M}$ to the outer unit normal, $N(x)$, at x. Assuming $\mathbb{M}$ is smoothly embedded in $\mathbb{R}^3$, the Gauss map is continuous and surjective but not necessarily injective. Indeed, the preimage of $u \in \mathbb{S}^2$ consists of all critical points of h_u with outer normal u, as opposed to $-u$. The multiplicity of N at u and $-u$ together is thus the number of critical points of h_u. We will see shortly that the total coverage of $\mathbb{S}^2$ is exactly the total absolute Gaussian curvature of $\mathbb{M}$.

Letting x be a point of $\mathbb{M}$ and $r > 0$ a radius, we define the *absolute Gaussian curvature* at x by taking the limit of a fraction of areas, $g(x) = \lim_{r \to 0} \frac{Area(N(A_r))}{Area(A_r)}$, where A_r is the neighborhood of points at distance at most r from x on $\mathbb{M}$. The *total absolute Gaussian curvature* is the integral of the local quantity, $G(\mathbb{M}) = \int_{x \in \mathbb{M}} g(x) \mathrm{d}x$. It should be clear that $G(\mathbb{M})$ is the area of the total coverage of $\mathbb{S}^2$, taking multiplicity into account. For a given direction, the multiplicity is $|N^{-1}(u)|$.

Hence, $G(\mathbb{M}) = \int_{u\in\mathbb{S}^2} |N^{-1}(u)|\mathrm{d}u$. Writing c_{avg} for the average number of critical points of the height functions, we thus have the total absolute Gaussian curvature equal to one half times the area of the sphere times that average, $G(\mathbb{M}) = 2\pi c_{\mathrm{avg}}$. This integral geometry formula for the curvature will come handy in the analysis of our algorithm. For more information on the integral geometry formulation of curvature see Santaló [Santaló 2004].

2.2. The PL Case

We do all computations on a piecewise linear approximation of the smooth 2-manifold. To transport the smooth concepts to the PL category, we think of the PL surface as being approximated by a smooth surface. Tightening the approximation, we get a series and take the limit. This is the general intuition we have in the background guiding the formulation of definitions in the PL case.

Triangulated surfaces. A *triangulation* of a 2-manifold $\mathbb{M}$ is a simplicial complex, K, whose underlying space is homeomorphic to $\mathbb{M}$: $|K| \approx \mathbb{M}$. It consists of vertices, edges, and triangles. To put K into $\mathbb{R}^3$, it suffices to map each vertex to a point; the edges and triangles are the convex hulls (of the images) of their vertices. This is a *geometric realization* if the triangles meet in shared edges and vertices but not in any other point sets. We call the result a *triangulated surface*, implicitly assuming that it is geometrically realized in $\mathbb{R}^3$. The *star* of a vertex is the set of simplices that contain it, and the *link* consists of all faces of simplices in the star that do not belong to the star, $\mathrm{St}\, v_i = \{\sigma \in K \mid v_i \in \sigma\}$; $\mathrm{Lk}\, v_i = \{\tau \subseteq \sigma \in \mathrm{St}\, v_i \mid \tau \notin \mathrm{St}\, v_i\}$. A PL function $f : |K| \to \mathbb{R}$ is determined by its values at the vertices. Assuming $f(v_i) \neq f(v_j)$ whenever $i \neq j$, we define the *lower link* as the subset of simplices in the link where f is smaller than at the vertex, $\mathrm{Lk}_- v_i = \{\sigma \in \mathrm{Lk}\, v_i \mid x \in \sigma \Rightarrow f(x) < f(v_i)\}$. Finally, v_i is *regular* if its lower link is contractible, and *critical*, otherwise. Since K triangulates a 2-manifold, every link is a circle and the only contractible closed subsets are points and closed paths. The lower link of a regular vertex is thus a single vertex or a path connecting two vertices. A *minimum* is characterized by $\mathrm{Lk}_- v_i = \emptyset$ and a *maximum* by $\mathrm{Lk}_- v_i = \mathrm{Lk}\, v_i$. In the remaining case, the lower link consists of $k + 1 \geq 2$ paths and we call v_i a *k-fold saddle*, or a *simple saddle* if $k = 1$.

In contrast to the smooth case, it is not possible to turn a k-fold into a simple saddle by a small perturbation. We therefore treat them directly, without reduction to simple cases. As an example, consider the Euler-Poincaré Theorem which relates the topology of the 2-manifold with the critical point structure of its functions. Denote the *index* of a simple critical point by $\mathrm{index}\,(v_i)$, recalling that $\mathrm{index}\,(v_i) = 0$ if v_i is a minimum, 1 if v_i is a simple saddle, 2 if v_i is a maximum. Assuming K is connected, it is characterized by its *genus* and we have $2 - 2 \cdot \mathrm{genus} = n - m + l = \sum_i (-1)^{\mathrm{index}\,(v_i)}$, where n, m, l are the number of vertices, edges, triangles in K and a k-fold saddle is represented by k simple saddles in the sum.

Critical regions. Another significant complication we encounter in the PL case is that a vertex is generally critical for an entire region of directions. Letting $h_u : |K| \to \mathbb{R}$ be the height function defined by $h_u(x) = \langle x, u\rangle$, the *critical region* of a vertex is the closure of the set of directions along which v_i is critical,

$$R_i \;=\; \mathrm{cl}\,\{u \in \mathbb{S}^2 \mid v_i \text{ is critical point of } h_u\}.$$

We construct it from the closed polygonal curve defined by the star of v_i. Specifically, we map each triangle in the star to its outer normal direction, a point on $\mathbb{S}^2$, and we connect the directions of two neighboring triangles by the shorter of the two connecting great-circle arcs. This gives a closed polygonal curve, π_i, which may or may not have self-intersections. To cope with the former, more complicated case, we orient π_i and define the *winding number* of a direction $u \in \mathbb{S}^2$ not on the curve as the number of times the curve goes around the directed line defined by u. Viewed along u, we count a counterclockwise turn as $+1$ and a clockwise turn as -1. Taking the sum we get the winding number, which are denoted as $w(u, \pi_i)$. For a detailed study on the polyhedron Gauss map, refer to [Alboul and Echeverria. 2005]. Examples are shown in Figure 3. The winding number of u relates to the type of the vertex in the height function defined by u. Specifically, if v_i is regular then the

winding number of u is 0, if v_i is a simple critical point then the winding number is $(-1)^{\text{index}\,(v_i)}$, and if v_i is a k-fold saddle then the winding number is $-k$.

Curvature. Thinking of a vertex as a tiny region in an approximating smooth surface, we define its *Gaussian curvature* as the area of its critical region weighted by the winding number. More useful in this paper is its *absolute Gaussian curvature* defined as the area weighted by the absolute winding number, $g(v_i) = \int_{u\in\mathbb{S}^2} |w(u, \pi_i)|\mathrm{d}u$. The *total absolute Gaussian curvature* is then the sum over all vertices, $G(K) = \sum_i g(v_i)$. Equivalently, it is the area of the sphere times half the average number of critical vertices, taking multiplicities into account, as usual. The average is taken over all height functions, and we count half the critical vertices because v_i is critical for $u \in \mathbb{S}^2$ as well as $-u \in \mathbb{S}^2$.

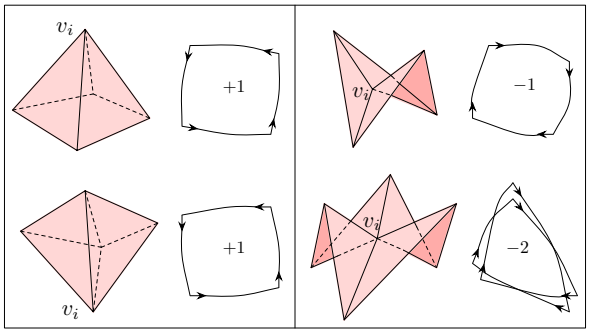

Fig. 3: Left: for a direction u with winding number $+1$ the corresponding vertex appears either as a maximum or a minimum. Right: for winding number -1 we have a simple saddle and for -2 we have a 2-fold or monkey saddle for the height function defined by the corresponding direction.

3. COMPUTATION

In this section, we describe how we compute the elevation maxima for a given triangulated surface in $\mathbb{R}^3$. The algorithm is straightforward and the only new insight is in the analysis, relating the running time with the total absolute Gaussian curvature of the surface.

Types and filters. Recall that there are four types of elevation maxima for a generic smooth surface, as illustrated in Figure 2. We have the same four cases for a generic triangulated surface K in $\mathbb{R}^3$. Each maximum is given by a set of two, three, or four points. We consider the case in which all these points are vertices of K. The cases in which some of the points in V lies on edges of K are similar. Let V be a set of vertices. A necessary requirement for V to define a maximum is that its vertices are critical for a common direction. More specifically, we need them critical in a particular direction that is determined by V. This direction, $u_V = (y - x)/||y - x||$, is slightly different for each type.

1. 1-legged case, $V = \{x, y\}$. Here, u_V is the direction defined by the two points.
2. 2-legged case, $V = \{x, y_1, y_2\}$. Letting y be the orthogonal projection of x onto the line passing through y_1 and y_2, u_V is defined if y lies between y_1 and y_2.
3. 3-legged case, $V = \{x, y_1, y_2, y_3\}$. Letting y be the orthogonal projection of x onto the plane passing through y_1, y_2, y_3, u_V is defined if y lies in the triangle they span.
4. 4-legged case, $V = \{x_1, x_2, y_1, y_2\}$. Letting x and y be the feet of the shortest line segment connecting the line passing through x_1 and x_2 with the line passing through y_1 and y_2, u_V is defined if x lies between x_1 and x_2 and y lies between y_1 and y_2.

PROJECTION FILTER. The direction u_V defined by the points in V is defined and belongs to the common intersection of critical regions, $u_V \in \bigcap_{v_i \in V} R_i$.

The cases in which some of the points in V lie on edges of K are similar. Given the vertices and edges, there is generically at most one choice of points and therefore at most one direction u_V that passes the Projection Filter. We generate these mixed sets by substituting edges connecting adjacent vertices in vertex-only sets. For example, the four vertices of a 4-legged case may give rise to a mixed set containing two vertices and one edge, specifying a 2-legged case, or a set of two edges, specifying a 1-legged case.

Note that the non-empty intersection of the critical regions is a necessary but not a sufficient condition for the set V to pass the Projection Filter. In turn, passing the Projection Filter is a necessary but not sufficient condition for the direction u_V to be an elevation maximum. For that, the set needs to satisfy another condition. To describe it, we write x_0 for x.

PERSISTENCE FILTER. For each pair x_i and y_j in V, there is an arbitrarily small perturbation u of u_V such that x_i, y_j is a birth-death pair for the height function h_u.

Algorithm. We compute the elevation maxima in three steps, starting with 2-, 3-, 4-tuplets V whose points have pairwise overlapping critical regions. The next two steps narrow down the selection using first the Projection and the Persistence Filter.

STEP 0. Compute the critical regions of the vertices of K. Letting the critical regions be the nodes of the intersection graph, R, we draw an arc if the two regions have a non-empty common intersection. For $k = 2, 3, 4$, let Q_k be the set of k-cliques, that is, the k-tuplets of nodes connected by all $\binom{k}{2}$ arcs. Let $S_0 = \bigcup_k Q_k$.
STEP 1. Subject each pair, triplet, and quadruplet in S_0 to the Projection Filter and let $S_1 \subseteq S_0$ be the collection that passes the filter.
STEP 2. Subject each pair, triplet, and quadruplet in S_1 to the Persistence Filter and let $S_2 \subseteq S_1$ be the collection that passes the filter.

Step 1 and 2 are the same as in [Agarwal et al. 2006], so we focus on the implementation of Step 0 in which we compute the 2-, 3-, 4-tuplets with pairwise intersecting critical regions.

Implementation. We break down Step 0 into three smaller steps, constructing the critical regions, finding the intersecting pairs, and computing the cliques of size 2, 3, 4 in the intersection graph. Implementation is done with Perl, C and CGAL [CGAL]. All computations are exact except estimating the area and the bounding box of a critical region.

STEP 0.1. Recall that each critical region, R_i, is given by a closed polygon with m_i edges on the sphere. Those edges may intersect, and we take time $O(m_i^2)$ to construct the decomposition of the sphere [de Berg et al. 1997], including winding numbers for all subregions. Reflecting R_i centrally through the origin in $\mathbb{R}^3$, we get the region $-R_i$ of inward normals along which v_i is critical. Constructing all critical regions takes time proportional to $\sum_i m_i^2$.
STEP 0.2. Most critical regions are small and simple. This suggests we use a bounding volume approach to find the intersecting pairs. Specifically, we find an axis-parallel box B_i in $\mathbb{R}^3$ that encloses the region R_i on $\mathbb{S}^2 \subseteq \mathbb{R}^3$. We do this in two steps, first computing the smallest enclosing sphere of R_i and second the smallest axis-aligned box that contains the sphere. Assuming that R_i fits inside a hemisphere of $\mathbb{S}^2$, the smallest enclosing sphere of its vertices also encloses R_i. To compensate for round-off errors, we increase the sphere slightly and compute the box B_i to enclose the enlarged sphere. Computing the smallest enclosing sphere of R_i takes randomized time $O(m_i)$, see [Welzl 1991]. Given the boxes B_i, we find the overlapping pairs using the segment-tree streaming algorithm as described in [Zomorodian and Edelsbrunner 2002]. Writing b_i for the number of boxes that overlap B_i, we have a total of $b = \frac{1}{2}\sum_i b_i$ of overlapping pairs. The streaming algorithm takes time proportional to $n \log_2^3 n + b$ to find them. For each pair of

overlapping boxes, we check whether or not the critical region they enclose have a non-empty intersection. Standard computational geometry methods allow us to determine whether or not R_i and R_j intersect in time $O(m_{ij} \log m_{ij})$, where $m_{ij} = m_i^2 + m_j^2$ [de Berg et al. 1997].
STEP 0.3. The result of Steps 0.1 and 0.2 is a graph R. Its n nodes are the critical regions, and its q arcs are the pairs of critical regions with non-empty overlap. Writing $q = \frac{1}{2}\sum_i q_i$, where q_i is the degree of the i-th node, we compute the cliques of size 2, 3, 4 by checking all pairs and triplets of neighbors. Finding the cliques that include R_i thus takes time $O(\binom{q_i}{1} + \binom{q_i}{2} + \binom{q_i}{3})$.

Analysis. The time for Step 0 is dominated by the requirement for Step 0.2, which is some constant times $T_{\text{new}} = \sum_i \binom{q_i}{1} + \binom{q_i}{2} + \binom{q_i}{3}$. The time for Step 1 is some constant times $|S_0| \leq T_{\text{new}}$ and that for Step 2 is some constant times $T = |S_1| n \log_2 n$. This adds up to some constant times $T_{\text{new}} + T$, as compared to $T_{\text{old}} + T$ for the algorithm in [Agarwal et al. 2006], where $T_{\text{old}} = \binom{n}{2} + \binom{n}{3} + \binom{n}{4}$. Any improvement thus hinges on two properties, namely that T_{old} is significantly larger than T_{new} as well as T. We now show that the first property holds under grossly simplifying assumptions, and we provide evidence in the next section that both properties hold for data we encounter in practice.

CAP ASSUMPTION. The critical regions are spherical caps, all of the same size, and their centers are uniformly distributed on $\mathbb{S}^2$.

Recall that the areas of the critical regions add up to the total absolute Gaussian curvature, $\sum_i Area(R_i) = G(K)$. This sum is also half the area of the sphere times the average number of critical points of the height functions, $G(K) = 2\pi c_{\text{avg}}$. It follows the area of a single critical region is $Area(R_i) = 2\pi c_{\text{avg}}/n$, and because the cap is smaller than the flat disk of the same radius, its radius squared is $\rho^2 > 2c_{\text{avg}}/n$. Two caps overlap if and only if the center of one is contained in the cap of radius 2ρ around the center of the other. The area of the enlarged cap is less than four times $Area(R_i)$. Hence the probability for a region R_j to overlap R_i is $\text{Prob}[R_i \cap R_j \neq \emptyset] \leq 4Area(R_i)/4\pi = 2c_{\text{avg}}/n$. Since expectations are additive even if the events are not independent, the expected number of k-tuplets of neighbors is $\text{Exp}[\binom{q_i}{k}] \leq \binom{n-1}{k} Area(R_i)^k/\pi^k \leq 2^k c_{\text{avg}}^k/k!$. Adding the expectations for $k = 1, 2, 3$ and all i gives

$$\text{Exp}[T_{\text{new}}] \leq n \cdot (2c_{\text{avg}} + 2c_{\text{avg}}^2 + \frac{4}{3}c_{\text{avg}}^3).$$

Recall that $c_{\text{avg}} = G(K)/2\pi$. It follows the average number of k-tuplets of critical regions overlapping a given one depends on the shape of the smooth surface and not on the size of the approximating triangulated surface. Similarly, the time for Step 0 depends on the shape and otherwise only linearly on the number of vertices in the triangulation.

4. EXPERIMENTS

In this section, we present the results of some of our computational experiments. Running our software on triangulated surfaces representing biomolecular structures, we gather statistics on critical regions, pairwise intersections, and elevation maxima. We use these statistics as evidence that the Cap Assumption is a reasonable approximation of the reality for our data and that the new algorithm runs about four orders of magnitude faster than the old one.

Input data. We use two types of triangulated surfaces approximating smooth models of biomolecular structures all listed in Table I. The first type is the molecular skin which uses hyperboloid and concave sphere patches to blend between the spheres that represent the atoms of a molecule [Edelsbrunner 1999]. An algorithm that constructs an approximating triangulated surface with guaranteed bounds on two- and three-dimensional angles is described in [Cheng et al. 2001] and software written by Ho-lun Cheng is available at [Biogeometry 2005]. For a representative of our data set, see Figure 4. The second type is the molecular surfaces generated by Chimera [Petterson et al. 2004].

The MSMS algorithm used in Chimera [Sanner and Olson 1996] constructs a triangulation of the solvent excluded surfaces initially computed by Connolly [Connolly 1983].

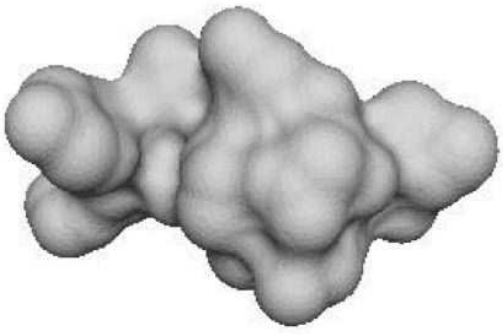

Fig. 4: Representative of our data sets, a triangulated surface approximating a peptide within the 1BRS protein.

Critical point statistics. For each data set, we estimate the minimum, average, and maximum number of critical points of the height functions, which we sample at one thousand directions chosen from $\mathbb{S}^2$. The results are shown in Table II left. Comparing the estimated with the actual average, which we get using $c_{\text{avg}} = G(K)/2\pi = \sum_i Area(R_i)/2\pi$, we see that the error is small. For example, for data set 4, the estimated c_{avg} is 29.92 while the actual average is 29.94. Since all our skin triangulations approximate a smooth surface to about the same accuracy, for different surfaces, the average number of critical points scales linearly with n. Indeed, c_{avg}/n is between 0.003 and 0.005 for all our skin data sets.

As mentioned earlier, each vertex of K is critical for a region of directions, in fact two antipodal regions. Most of these regions are simple, that is, defined by a polygon without self-intersections. As shown in the last column in Table II, the percentage of non-simple polygons is indeed rather small. Besides checking for self-intersections, we measure the complexity of a critical region by counting the triangles we need to triangulate it on the sphere. The minimum, average, and maximum of this number are given in the right half of Table II.

Intersection statistics. The following statistics were collected for the finer molecular skin surfaces only. Recall that we compute the pairs of intersecting critical regions in two steps, first finding the intersections among the bounding boxes and second among the critical regions. Table III gives the statistics for both.

Table I: The triangulated surfaces used in our computational experiments together with their numbers of vertices, edges, and triangles. Top: molecular skin surfaces. Bottom: molecular Chimera surfaces.

id	name	n	m	ℓ
0	1BRS-5to6	1,370	4,104	2,736
1	1CLU-DBG	3,149	9,441	6,294
2	1BRS-A-5to10	4,248	12,738	8,492
3	1BRS-A-30to40	6,114	18,336	12,224
4	1BRS-A-17to25	7,799	23,391	15,594
5	1BRS-A-5to10	836	2,502	1,668
6	1BRS-A-30to40	1,372	4,110	2,740
7	1BRS-A-17to25	1,595	4,119	3,186

Given a pair of intersecting boxes, we test whether or not the corresponding critical regions intersect by checking the overlap among the triangles in their triangulations. The average number of

Table II: Left: estimated minimum, average, and maximum of the number of critical points of the height functions. Right: minimum, average, and maximum of the number of triangles needed to triangulate the critical regions. Top: molecular skin surfaces. Bottom: molecular Chimera surfaces.

id	$c_{\min}$	c_{avg}	$c_{\max}$	$\frac{c_{\text{avg}}}{n}$	$r_{\min}$	r_{avg}	$r_{\max}$	%
0	2	6.41	16	0.0047	2	3.99	8	12
1	2	13.50	44	0.0043	2	4.01	12	15
2	6	17.07	34	0.0040	2	4.01	10	17
3	10	25.14	46	0.0041	2	4.01	10	16
4	12	29.92	64	0.0038	2	4.01	10	20
5	6	16.01	32	0.0192	2	4.08	11	29
6	10	27.13	46	0.0198	2	4.13	15	30
7	14	31.02	54	0.0194	2	4.09	10	33

Table III: Left: the minimum, average, and maximum number of boxes intersecting a given box. Right: the minimum, average, and maximum number of critical regions intersection a given critical region.

id	$b_{\min}$	b_{avg}	$b_{\max}$	$\frac{b_{\text{avg}}}{n}$	$q_{\min}$	q_{avg}	$q_{\max}$	$\frac{q_{\text{avg}}}{n}$
0	12	94	207	0.069	9	40	97	0.029
1	27	204	626	0.065	11	82	250	0.026
2	52	236	556	0.056	20	92	201	0.022
3	95	243	859	0.040	29	134	330	0.022
4	99	423	1,276	0.054	35	160	543	0.021

Table IV: Left: the number of cliques before and after the Projection Filter and the Persistence Filter. Right: dominant terms in the running time of the old and the new algorithms.

id	$\lvert S_0\rvert/10^3$	$\lvert S_1\rvert$	$T_{\text{old}}/10^{10}$	$T_{\text{new}}/10^6$	$T/10^6$
0	1,608	2,373	15	24	33
1	32,119	20,521	410	508	749
2	43,572	17,175	1,356	720	882
3	198,023	56,797	5,820	3,327	4,368
4	433,116	94,300	15,411	7,354	9,508

triangle-triangle checks is consistently between 11 and 12, which justifies the use of this brute-force over a more sophisticated method.

Similar to the number of critical points, we expect that the average number of boxes intersecting a given box and the average number of critical regions intersecting a given critical region scale linearly with n. Indeed, b_{avg}/n is between 0.04 and 0.07 and q_{avg}/n is between 0.02 and 0.03 for all our skin data sets. The latter is about six times the average number of critical points; compare this with the factor two we got under the Cap Assumption. The observed relation between these two quantities is only about three times as loose, which is reasonable considering that real data necessarily violates the Cap Assumption to some extent (due to irregular shapes and different orientations of the critical regions). The new algorithm starts with T_{new} tuplets. A back-of-the-envelope calculation suggests that T_{new} is roughly $n\binom{q_{\text{avg}}}{3}$, which is roughly a factor of ten thousand smaller than $\binom{n}{4}$, independent of the value of n. We thus might expect the new algorithm runs about four orders of magnitude faster than the old one.

Running time. Recall that S_0 is the set of cliques of size 2, 3, or 4 in the intersection graph of the critical regions. The subset $S_1 \subseteq S_0$ contains all cliques that pass the Projection Filter, and the subset $S_2 \subseteq S_1$ contains all cliques that also pass the Persistence Filter. The sizes of the first two sets are given in the left of Table IV.

Most relevant to the running time of the algorithms for computing elevation maxima is S_1. Indeed, both the old and the new algorithm start with sets of 2-, 3-, and 4-tuplets that contain the cliques in S_0 and much more. As shown in Table IV on the right, the overestimate by the old algorithm is about ten thousand times that of the new algorithm. Furthermore, in the new algorithm, the time

for Step 0 and Steps 1 and 2 is fairly balanced. This implies a speed-up of about four orders of magnitude, which is consistent with back-of-the-envelope calculation mentioned above.

5. CONCLUSION

The main result of this paper is a new algorithm for computing all elevation maxima of a triangulated surface in $\mathbb{R}^3$. We provide experimental evidence that for practical data, the new algorithm runs about four orders of magnitude faster than the old one. The improvement is achieved by making the running time dependent on the total absolute Gaussian curvature of the surface and to a lesser extent on the number of vertices in the approximating triangulation. Now, the total absolute Gaussian curvature has different definitions for smooth and for piecewise linear surfaces. It appears that $G(K)$ approaches $G(\mathbb{M})$ as K is refined and forms a progressively more accurate approximation of the smooth surface $\mathbb{M}$. However, we do not have a proof and we do not know under what conditions this is true.

There is room for performance improvement, one promising direction is to parallelize the computations. It would also be interesting to sample the elevation maxima if this can be done faster than computing all. For example, is it possible to compute all elevation maxima larger than some threshold without spending the time to determine (and discard) the elevation maxima that do not exceed that threshold?

REFERENCES

AGARWAL, P. K., EDELSBRUNNER, H., HARER, J., AND WANG, Y. 2006. Extreme elevation on a 2-manifold. *Discrete Comput. Geom. 36*, 553–572.

ALBOUL, L. AND ECHEVERRIA., G. 2005. Polyhedral Gauss maps and curvature characterization of triangle meshes. *Lecture Notes in Computer Science 3605*, 14–33.

BANCHOFF, T. F. 1970. Critical points and curvature for embedded polyhedral surfaces. *Amer. Math. Monthly 77*, 475–485.

BIOGEOMETRY. 2005. The biogeometry project. http://biogeometry.duke.edu/.

CAZALS, F., CHAZAL, F., AND LEWINER., T. 2003. Molecular shape analysis based upon the Morse-Smale complex and the connolly function. *Proc. 19th Ann. Sympos. Comput. Geom.*, 351–360.

CGAL. Computational geometry algorithms library. http://www.cgal.org.

CHENG, H. L., DEY, T. K., EDELSBRUNNER, H., AND SULLIVAN, J. 2001. Dynamic skin triangulation. *Discrete Comput. Geom. 25*, 525–568.

COHEN-STEINER, D., EDELSBRUNNER, H., AND HARER., J. 2007. Stability of persistence diagrams. *Discrete Comput. Geom. 37*, 103–120.

COHEN-STEINER, D., EDELSBRUNNER, H., AND HARER, J. 2009. Extending persistence using Poincaré and Lefschetz duality. *Found. Comput. Math. 9*, 79–103.

COHEN-STEINER, D. AND MORVAN, J. 2006. Second fundamental measure of geometric sets and local approximation of curvatures. *J. Differential Geom. 74*, 3, 363–394.

COLE-MCLAUGHLIN, K., EDELSBRUNNER, H., HARER, J., NATARAJAN, V., AND PASCUCCI, V. 2004. Loops in Reeb graphs of 2-manifolds. *Discrete Comput. Geom. 32*, 231–244.

CONNOLLY, M. L. 1983. Analytic molecular surface calculation. *J. Appl. Crystallogr 6*, 548–558.

CONNOLLY., M. L. 1986. Shape complementarity at the hemoglobin albl subunit interface. *Biopolymers 25*, 1229–1247.

DE BERG, M., VAN KREVELD, M., OVERMARS, M., AND SCHWARZKOPF, O. 1997. *Computational Geometry - Algorithms and Applications*. Springer-Verlag, Berlin, Germany.

EDELSBRUNNER, H. 1999. Deformable smooth surface design. *Discrete Comput. Geom. 21*, 87–115.

EDELSBRUNNER, H., LETSCHER, D., AND ZOMORODIAN, A. 2002. Topological persistence and simplification. *Discrete Comput. Geom. 28*, 511–533.

GEORGIADIS, L., TARJAN, R., AND WERNECK, R. F. 2006. Design of data structure for mergeable trees. *Proc. 17th Ann. ACM-SIAM Sympos. Discrete Algorithm*, 394–403.

MORVAN, J. 2008. *Generalized Curvatures*. Springer-Verlag, Berlin, Germany.

MUNKRES, J. R. 1984. *Elements of Algebraic Topology*. Addison-Wesley, Redwood City, CA, USA.

PETTERSON, E. F., GODDARD, T. D., HUANG, C. C., GOUCH, G. S., GREENBLATT, D. M., MENG, E. C., AND FERRIN, T. E. 2004. Ucsf chimera - a visualization system for exploratory research and analysis. *J. Comput. Chem. 25*, 1605–1612.

SANNER, M. F. AND OLSON, A. J. 1996. Reduced surface: an efficient way to compute molecular surfaces. *Biopolymers 38,* 305–320.

SANTALÓ, L. 2004. *Integral geometry and geometric probability*. Cambridge University Press, Cambridge, UK.

WANG, Y., AGARWAL, P. K., BROWN, P., EDELSBRUNNER, H., AND RUDOLPH, J. 2005. Course and reliable geometric alignment for protein docking. *Proc. Pacific Sympos. Biocomputing*, 64–75.

WELZL, E. 1991. Smallest enclosing disks (balls and ellipsoids). *Lecture Notes in Computer Science 555*, 359–370.

ZOMORODIAN, A. AND EDELSBRUNNER, H. 2002. Fast software for box intersections. *Internat. J. Comput. Geom. Appl. 12*, 143–172.

Received Nov 2009; revised March 2010; accepted March 2011