Computing Distance Functions from Generalized Sources on Weighted Polyhedral Surfaces

Marta Fort* J. Antoni Sellarès*

Abstract

We present algorithms for computing approximate distance functions and shortest paths from a generalized source (point, segment, polygonal chain or polygonal region) on a weighted non-convex polyhedral surface in which obstacles (represented by polygonal chains or polygons) are allowed. We also describe an algorithm for discretizing, by using graphics hardware capabalities, distance functions. Finally, we present algorithms for computing discrete k-order Voronoi diagrams.

1. Introduction

Computing shortest paths, and consequently distances, on polyhedral surfaces is a fundamental problem in computational geometry with important applications in geographical information systems, robotics and computer graphics. Shortest path distances computation often arise as a subroutine in the solution of other problems, for example Voronoi diagrams computation.

In this paper we present algorithms for computing approximate distance functions and shortest paths from a generalized source on a possibly non-convex weighted polyhedral surface with obstacles. We also describe an algorithm that uses hardware graphics capabilities to discretize the distance function. As an application, we present algorithms for computing discrete k-order Voronoi diagrams of a set of generalized sites on a weighted polyhedral surface with obstacles. We omit some proofs due to lack of space.

1.1. Preliminaries

Let $\mathcal{P}$ be a possibly non-convex polyhedral surface represented as a mesh consisting of n triangular faces $f_1, \cdots, f_n$ with associated positive weights $w_1, \cdots, w_n$, respectively. The weight associated with an edge, is the minimum of the weights of the two neighboring faces. From now on, a generalized element on $\mathcal{P}$ refers to a point, segment, polygonal chain or polygon. We model obstacles in $\mathcal{P}$ by a set of non-punctual generalized elements on $\mathcal{P}$. We only consider paths from generalized sources to points on $\mathcal{P}$ that stay on $\mathcal{P}$ and avoid the obstacles. The cost of a path Π on $\mathcal{P}$ is defined by $\|\Pi\| = \sum_{i=1}^{n} w_i|\Pi_i|$, where $|\Pi_i|$ denotes the Euclidean length of the path lying inside the face f_i. For a generalized source s and a point q on $\mathcal{P}$, the path of least cost between them is called shortest path. The cost of the shortest path is called the (shortest path) distance between s and q. The distance function defined by a generalized source s on $\mathcal{P}$ is a function d_s such that for any point $q \in \mathcal{P}$, $d_s(q)$ is the distance between s and q. Given $\varepsilon \in]0, 1[$, a path is called a $(1+\varepsilon)$ approximation of a shortest path between a generalized source and a point on $\mathcal{P}$ if its cost is at most $(1 + \varepsilon)$ times the cost of a shortest path.

Let S be a set of r generalized sites on the polyhedral surface $\mathcal{P}$ and let S' be a subset of k sites of S, $k \in \{1, \cdots, r-1\}$. The set of points of $\mathcal{P}$ closer to each site of S' than to any other site of S, is a possibly empty region called the k-order Voronoi region of S'. The set of k-order Voronoi regions of all subsets of k sites of S is called the k-order Voronoi diagram of S. When $k = 1$ and $k = r - 1$ the k-order Voronoi diagram is called the closest Voronoi and the furthest Voronoi diagram, respectively.

1.2. Related Work

Shortest Paths

Results on algorithms computing $(1 + \varepsilon)$ approximate shortest paths on weighted polyhedral surfaces involve geometric parameters that have been omitted for the sake of clarity. In 1991 Mitchell and Paparimitriou [MP] gave the characterization of the shortest paths on weighted surfaces and presented an algorithm that uses the continuous Dijkstra method to provide an $(1 + \varepsilon)$approximate shortest path that runs in $O(n^8 \log n/\varepsilon)$ time and $O(n^4)$ space. There exist several papers providing $(1 + \varepsilon)$ approximate short-

*Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain, {mfort,sellares}@ima.udg.edu. Partially supported by grant TIN2007-67982-C02-02.

978-0-7695-3243-1/08 $25.00 © 2008 IEEE
DOI 10.1109/ICCSA.2008.21

est paths that discretize the polyhedral surface reducing the problem to the determination of shortest paths on weighted graphs using Dijkstra algorithm [ALMS, AMS1, AMS2]. The discretization scheme places Steiner points on the triangle edges or on the bisectors of the triangle vertices. Between the formers, the best algorithm follows a logarithmic discretization scheme and has time complexity $O(n/\varepsilon \log 1/\varepsilon (1/\sqrt{\varepsilon} + \log n))$ [AMS1]. An algorithm of time complexity $O(n/\sqrt{\varepsilon} \log n/\varepsilon \log 1/\varepsilon)$ that uses the later approach is presented in [AMS2]. However, there exist an alternative strategy called Bushwhack, proposed by Sun and Reif [SR]. They use the shortest path properties smartly and obtain, by using the discretization strategy introduced in [AMS1], an approximate shortest path in $O(n/\varepsilon \log 1/\varepsilon \log n/\varepsilon)$ time.

Voronoi Diagrams

The computation of exact generalized Voronoi diagrams use to be complicated because it involves the manipulation of high-degree algebraic curves or surfaces and their intersections. Algorithms based on different distance functions have been proposed to compute 2D and 3D discretized Voronoi diagrams on a grid by using graphics hardware [FG, HKLMC].

Mount [Mou] shows that the closest Voronoi diagram of r sites on a non-weighted polyhedral surface with n faces, assuming $r \leq n$, has complexity $O(n^2)$ in the worst case and also gives an algorithm that computes the diagram in $O(n^2 \log n)$ time. Aronov at al. [AKOV] show that the furthest Voronoi diagram of the sites on the polyhedral surface has maximum combinatorial complexity $\Theta(rn^2)$, and present an algorithm that computes the diagram in $O(rn^2 \log^2 r \log n)$ expected time. In [FS] an algorithm that computes discrete generalized higher-order Voronoi diagrams on non weighted polyhedral surfaces is presented. To the best of our knowledge there exist no results related to the complexity and computation of a k-order Voronoi diagram on weighted surfaces.

Graphics Hardware

The increasing programmability and high computational rates of graphics processing units (GPUs) make them attractive as an alternative to CPUs for general-purpose computing. Recently, different algorithms and applications that exploit the inherent parallelism, easy programmability and vector processing capabilities of GPUs have been proposed [OLGHKLP, PBMH]. In computational geometry and GIS fields there exist several algorithms that have a fast hardware-based implementation [HKLMC, GHLM, KCC, KMV, AKMV, FEKKVS, MKV].

The graphics pipeline [OpenGL] is divided into several stages. The input is a list of 3D geometric primitives expressed as vertices defining points, lines, etc. with attributes associated. The output is an image in the frame buffer, a collection of several hardware buffers corresponding to two dimensional grids whose cells are called pixels. In the first stage of the pipeline, per-vertex operations take place, each input vertex is transformed from 3D coordinates to window coordinates. Next stage is rasterization, when it finishes we obtain a fragment, with its associated attributes, for each pixel location covered by a geometric primitive. Fragment attributes are obtained from the attributes associated to the vertices by linear interpolation. The third stage, the fragment stage, computes the colour for each pixel in the frame buffer, according to the fragments corresponding to it taking into account a series of tests such as the depth test and per-fragment operations. Information of a user defined rectangle of the colour buffer can be easily transferred to the CPU or directly to a texture. The unique programmable parts of the graphics pipeline are the vertex and fragment shaders which are executed on a per-vertex and per-fragment basis, respectively. They are used to change the vertex or fragment attributes.

1.3. Our Contribution

In this paper we present an algorithm for computing approximately shortest paths, and consequently distances, from a generalized source on a, possibly non-convex,weighted polyhedral surface $\mathcal{P}$ with obstacles. More specifically:

- We extend the discretization scheme introduced by Sun and Reif [SR] to handle the generalized source case in arbitrary position. As generalized sources we consider points, segments, polygonal lines and polygons. We prove that the discretization scheme provides $(1+\varepsilon)$-approximate shortest paths (Section 2).
- We obtain $(1+\varepsilon)$-approximate shortest path distances on the weighted polyhedral surface by using Bushwack strategy (Section 3).
- The algorithm is extended to the case of several sources and their distanced field is obtained, providing an implicit representation of the closest Voronoi diagram of a set of generalized sites on the surface (Section 4).
- We explain how the distance to arbitrary points of the surface and the actual shortest path is obtained (Section 5)

We also present two algorithms, based on distance functions and hardware graphics capabilities, for computing dis-

crete Voronoi diagrams of a set of m generalized sites on the polyhedral surface $\mathcal{P}$ with obstacles:

- We present an algorithm for discretizing, by using hardware graphics, the distance function defined by a generalized source on $\mathcal{P}$ (Section 6).
- We obtain discrete k-order Voronoi diagrams, $k = 1, \cdots, m-1$ (Section 7).

We present some experimental results (Section 8) and we finally provide some conclusions and future work (Section 9).

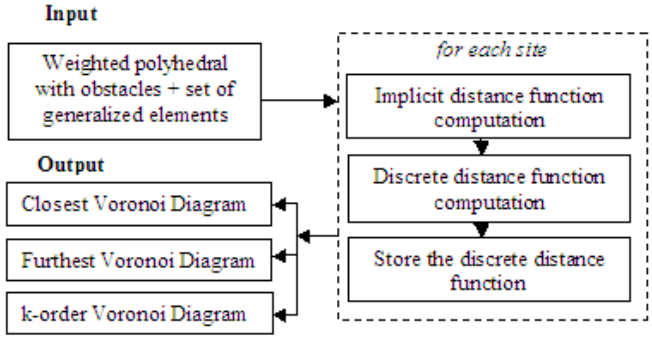

Figure 1. Shortest path $(1+\varepsilon)$-approximate distances and Voronoi diagrams computation of a set of generalized sites.

2. Discretization Scheme

We provide a discretization scheme to obtain $(1+\varepsilon)$-approximate distances from a point or segment source to a graph node. In fact, we define a logarithmic discretization scheme which places Steiner points on the edges of the triangulated polyhedral surface $\mathcal{P}$ according to a given parameter $0 < \varepsilon \leq 1$. The scheme we propose adapts the one provided by Sun and Reif [SR] to take into account a point or segment source s which is not necessarily a vertex nor an edge.

We start by providing some definitions and notation, we denote E the set of edges of $\mathcal{P}$, f a face, e an edge, v a vertex and x an arbitrary point. We define $F(f)$ as the union of the faces without empty intersection with f, face f included, $F(x)$ as the union of the faces containing point x. Let $E(x)$ denote the set of edges containing x and $S(x)$ the set of sources contained in $F(x) \setminus \{x\}$, in the current case $S(x)$ will be empty or $\{s\}$. Let D_x be the minimal distance between x and $S(x) \cup E \setminus E(x)$, $D_e = sup\{D_v | v \in e\}$ and $D_{v_e} = D(e)$. For each point x we define the radius $r'(x)$ to be $D_x/5$, and the weighted radius $r(x)$ of x to be $\frac{w_m}{w_M} r'(x)$, where w_m, w_M are the minimal and maximal weights of the faces in $F(x)$. Notice that these radius take into account not only the proximity of the edges in $E \setminus E(x)$ but also the proximity of s to x, when x is in $F(f_s)$. Finally $V(x)$ is the vicinity of point x. Vicinity $V(x)$ is defined to have radius $r_\varepsilon(x) = \varepsilon r(x)$, it contains all the points around x at distance at most $r_\varepsilon(x)$. When instead of a point x a source s is considered $F(s)$, $S(s)$, $E(s)$, D_s, $r'(s)$, $r(s)$, $V(s)$ and $r_\varepsilon(s)$ are analogously defined.

Steiner points are placed on the edges of $\mathcal{P}$ considering that source s and each vertex v has a vicinity. For a given vertex v_i Steiner points $v_{i,1}, v_{i,2}, \ldots, v_{i,k_i}$ are on edge $e = \overline{v_i v_j}$ and outside the vicinities, they are chosen according to the following criteria:

- When e is not in $F(f_s)$, the first node is placed so that $|\overline{v_i v_{i,0}}| = r_\varepsilon(v_i)$, the rest using the equality $|\overline{v_{i,k} v_{i,k+1}}| = \varepsilon\, D_{v_{i,k}}$, until placing v_{i,k_i} where $\overline{v_{i,k_i} v_i} + \varepsilon\, D_{v_{i,k_i}} \geq |\overline{v_i v_e}|$.
- When $e \in F(s)$ we take into account the vicinity of s, $V(s)$. If $e \cap V(s) = \emptyset$ we proceed as in the previous case. Otherwise, when $e \cap V(s) \neq \emptyset$, $e \setminus V(s)$ defines two subedges $\overline{v_i v}$ and $\overline{v v_j}$, since we are considering v_i we choose Steiner points on $e' = \overline{v_i v}$. Steiner point v_{i_0} is placed so that $|\overline{v_i v_{i,0}}| = r_\varepsilon(v_i)$, Steiner points for $i = 1 \ldots j_i$ so that $|\overline{v_{i,k} v_{i,k+1}}| = \varepsilon\, D_{v_{i,k}}$ until $k = j_i$ where $\overline{v_{i,j_i} v} + \varepsilon D_{v_{i,j_i}} \geq |\overline{v_i, v_{e'}}|$. The rest of points are placed considering endpoint v, Steiner point $v_{i,k_i} = r_\varepsilon(v)$, for $k = k_i \ldots j_i + 1$ according to $|\overline{v_{i,k} v_{i,k+1}}| = D_{v_{i,k}}$ until $\overline{v_{i,j_i+1} v} + \varepsilon D_{v_{i,j_i+1}} \geq |\overline{v, v_{e'}}|$.

Lemma 2.1 *The number of Steiner points placed on each edge is in* $O\left(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon}\right)$.

Proof. The proof is omitted for lack of space.

We build a graph whose nodes are the vertices of $\mathcal{P}$ and the Steiner points. Graph edges consist of face-crossing segments joining pairs of Steiner points of the same face, and edge-using segments joining consecutive nodes along an edge, and each vertex v to the first node of the edges where v is incident to. Notice that when s is a vertex this scheme is the logarithmic scheme provided by Sun and Reif [SR].

2.1. Approximation Analysis

We are interested in bounding the committed error when using the previous graph to compute weighted distances from a point or segment source s. With this purpose we present different Lemmas.

From now one we denote by $\tilde{w}$ and w' the largest and smallest weight of the faces of $\mathcal{P}$. We assume $\varepsilon \leq 1$ and $w' \geq 1$, it is not a restriction because it can be achieved by adding one to all the face weights.

Lemma 2.2 *For any path p from source $s \in f_s$ to node $t \in f_t$ which does not intersect the vertex vicinities of f_t there is a normalized path $\hat{p}$ so that $||\hat{p}|| = (1 + \frac{\varepsilon}{2})||p||$.*

Proof. Assume that p passes through a vertex vicinity, $V(v)$. We distinguish between two situations depending on whether v is a vertex of f_s or not (See Figure 2).

A) Let us assume that v is a vertex of f_s. We denote u_1, u_2, the first bending point of p in $V(v)$ and u_2'' the last bending point of p in $F(v)$ (See Figure 2 a)). By using the definition of D_v and the fact that the radius of $V(v)$ is $\frac{\varepsilon}{5}D_v$ we obtain:

On the one hand that $|p[u_2'', u_2]| + |\overline{u_2 v}| \geq |\overline{u_2'' v}| \geq D_v$ and $|p[u_2'', u_2]| \geq D_v - \varepsilon\frac{D_v}{5}$ for being $|\overline{u_2 v}| \leq \varepsilon\frac{D_v}{5}$. Therefore,

$$\frac{|\overline{u_2 v}|}{|p[u_2'', u_2]|} \leq \frac{\varepsilon \cdot D_v/5}{D_v - \varepsilon \cdot D_v/5} =$$

$$= \frac{\varepsilon}{5 - \varepsilon} \leq \frac{\varepsilon}{4} \qquad (1)$$

On the other hand that $|p[s, u_1]| + |\overline{u_1 v}| \geq |\overline{sv}| \geq D_v$ and $p[s, u_1]| \geq D_v - \varepsilon\frac{D_v}{5}$ for being $|\overline{u_1 v}| \leq \varepsilon\frac{D_v}{5}$. Therefore,

$$\frac{|\overline{u_1 v}|}{|p[s, u_1]|} \leq \frac{\varepsilon \cdot D_v/5}{D_v - \varepsilon \cdot D_v/5} =$$

$$= \frac{\varepsilon}{5 - \varepsilon} \leq \frac{\varepsilon}{4} \qquad (2)$$

We denote r_1(/r_2) the region with minimum weight traversed by $p[s, u_1]$ (/$p[u_2, u_2'']$) and u_1'(/u_2') the last point of p in r_1(/r_2), notice that u_1' may be s. Finally we denote w_{r_1}(/w_{r_2}) the weight of r_1(/r_2) (See Figure 2 a)). Let us consider the normalized path $\hat{p}[s, u_2''] = \{p[s, u_1'], \overline{u_1' v}, \overline{v u_2'}, p[u_2', u_2'']\}$. By comparing the costs of $\hat{p}[s, u_2'']$ and $p[s, u_2'']$ and using inequalities (1) and (2) we obtain that:

$$||\hat{p}[s, u_2'']|| - ||p[s, u_2'']|| =$$

$$= w_{r_1}|\overline{u_1' v}| + w_{r_2}|\overline{v u_2'}| - ||p[u_1', u_1]|| -$$

$$-||p[u_1, u_2]|| - ||p[u_2, u_2']|| \leq$$

$$\leq (w_{r_1}|\overline{u_1' v}| - ||p[u_1', u_1]||) + (w_{r_2}|\overline{v u_2'}| - ||p[u_2, u_2']||) \leq$$

$$\leq w_{r_1}|\overline{u_1 v}| + w_{r_2}|\overline{v u_2}| \leq$$

$$\leq w_{r_1}\frac{\varepsilon}{4}|p[s, u_1]| + w_{r_2}\frac{\varepsilon}{4}|p[u_2, u_2'']| \leq$$

$$\leq \frac{\varepsilon}{4}||p[s, u_1]|| + \frac{\varepsilon}{4}||p[u_2, u_2'']|| \leq \frac{\varepsilon}{4}||p[s, u_2'']||$$

Therefore,

$$||\hat{p}[s, u_2'']|| \leq (1 + \frac{\varepsilon}{4})||p[s, u_2'']|| \qquad (3)$$

B) Now, we assume that v is not a vertex of f_s. We proceed in a similar way, let u_1'', and u_2'' be the first and last bending points of p in $F(v)$ (See Figure 2 b)). Point u_1'' plays the role of s, we define $\hat{p}[u_1'', u_2''] = \{p[u_1'', u_1'], \overline{u_1' v}, \overline{v u_2'}, p[u_2', u_2'']\}$ (Figure 2 b)). It can be proven that

$$|\overline{u_1 v}|/|p[u_1'', u_1]| \leq \frac{\varepsilon}{4} \qquad (4)$$

(equivalent to (1)), again using the same reasoning we obtain a result equivalent to (3):

$$||\hat{p}[u_1'', u_2'']|| \leq (1 + \frac{\varepsilon}{4})||p[u_1'', u_2'']|| \qquad (5)$$

Assuse that p passes through l vertex vicinities, $V(v_1)$, $V(v_1)$, ..., $V(v_l)$. For each vertex we replace the subpath p_i of p that passes through $V(v_i)$ for the normalized path $\hat{p}_i$. Using the correspondent inequality ((3) or (5)) for each vicinity we obtain that $||\hat{p}|| \leq ||p|| + \frac{\varepsilon}{4}$

$$\sum_{i=1}^{l} ||p_i|| \leq (1 + \frac{\varepsilon}{2})||p||.$$

■

Lemma 2.3 *For any path p from source $s \in f_s$ to node $t \in f_t$ there is a normalized path $\hat{p}$ so that $||\hat{p}|| = (1 + \frac{\varepsilon}{2})||p||$.*

Proof. We omit the proof due to lack of space, however, it is similar to the proof of Lemma 2.2.

Theorem 2.1 *The obtained graph contains a $(1 + 3\varepsilon)$-approximation of the shortest path Π from point source s to an arbitrary node t.*

Proof. According to Lemma 2.3 it exists a normalized path $\hat{\Pi}$ such that $||\hat{\Pi}|| \leq (1 + \frac{1}{2}\varepsilon)||\Pi||$.

Let $\overline{v} = \overline{v_1 v_2}$ be a segment of $\hat{\Pi}$ contained in a face f. Endpoint v_1(/v_2) of $\overline{v_1 v_2}$ is on a segment $\overline{u_{1,1}, u_{1,2}}$(/$\overline{u_{2,1}, u_{12,2}}$) which is delimited by either two Steiner points or a vertex and a Steiner point, they are named a *pure Steiner segment* and a *half Steiner segment*, respectively. Segments conforming $\hat{\Pi}$ belong to one of the following three categories: (1) Both endpoints are on pure Steiner segments; (2) An endpoint is on a pure Steiner segment and the other on a half Steiner segment; (3) Both endpoints are on half Steiner segments. For each of the three

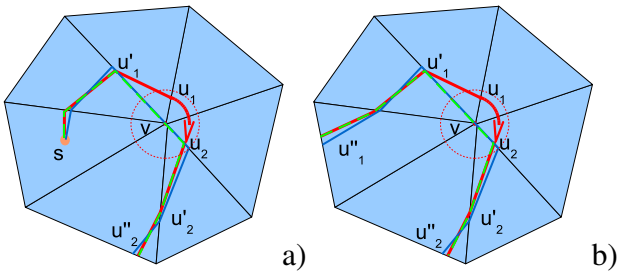

Figure 2. A subpath of path $\pi_{s,t}$ from $s \in f_s$ to $t \in f_t$ in red, in dashed green a normalized path and in blue a path on the graph. Path $\pi_{s,t}$ goes through vertex vicinity $V(v)$, where v is: a) a vertex of the face containing s; b) a vertex of a face not containing s nor t.

cases it can be proven that $|\overline{u_{1,i}, u_{2,j}}| \leq (1+2\varepsilon)|\overline{v_1 v_2}|$ for $i, j \in \{1, 2\}$:

$$|\overline{v_1 v_2}| \leq |\overline{v_1 u_{1i}}| + |\overline{u_{1i} u_{2j}}| + |\overline{u_{2j} v_2}| \leq$$

$$\leq \varepsilon|\overline{v_1 v_2}| + |\overline{u_{1i} u_{2j}}| + \varepsilon|\overline{v_1 v_2}| = |\overline{u_{1i} u_{2j}}| + 2\varepsilon|\overline{v_1 v_2}|.$$

Consequently, we can construct a path Π' such that

$$||\Pi'|| \leq (1+2\varepsilon)||\hat{\Pi}|| \leq (1+2\varepsilon)(1+\frac{\varepsilon}{2})||\Pi|| \leq (1+\frac{5\varepsilon}{2})||\Pi||.$$

■

3. Distance Functions

We also show that the Bushwack strategy can be used for the case of a point source in general position and for a segment source. We consider polygonal line and polygon sources and end allowing generalized obstacles on the surface.

3.1. From Point Source to Node

Let us consider an arbitrary point source s in a face of $\mathcal{P}$, we provide a way to obtain an approximate shortest paths from s to any node of the graph considering the discrete graph presented in Section 2 by adapting the Bushwack strategy.

Distance Propagation

It is well known that two shortest path originated to the same source point cannot intersect in the interior of any face, we can use Bushwhack strategy to compute distances from a point source s.

Now s is not a vertex, thus, we have to adapt the initialization step where intervals $I_{s,-,e}$ associated to the source s are created. These intervals encode distance function d_s of source s on the edges e of the triangle(s) containing s. Bushwack strategy propagates the distance function across mesh triangles in a lazy and best-first propagation scheme. When a node v on an edge e is first visited, several intervals $I_{v,e,e'}$ are created, with e' opposite to v when v is a vertex or adjacent to e otherwise. Interval $I_{v,e,e'}$ contains those contiguous nodes of e' whose shortest path from s to $v' \in I(v, e, e')$ may use node v before arriving v' via an edge-using or face-crossing segment contained on the face determined by e' and v. According to Lemma 2.1 and Theorem 2.1 and the Bushwack strategy complexity when there exist m nodes per edge, we provide the following Theorem.

Theorem 3.1 *$(1 + 3\varepsilon)$-approximate distances from a point source s to the graph nodes can be obtained in $O(mn \log(mn))$ time and $O(mn)$ space, where $m \in O(\frac{1}{\varepsilon} \log \frac{1}{\varepsilon})$.* ■

3.2. From Segment Source to Node

Let us now consider a segment source s on $\mathcal{P}$, we use the previous discretization scheme. We want to note that D_s is the minimal distance between s and the edges of $\mathcal{P}$ that do not intersect s. It is used to define the radius of $V(s)$, the vicinity of s, which is $r_\varepsilon(s) = \varepsilon \frac{w_m}{w_M} \frac{D_s}{5}$. Given point x, D_x is the minimal distance between x and $S(x) \cup E \setminus E(x)$, where $S(x) = \{s\} \setminus \{x\}$ when $s \in F(x)$ or $S(x) = \emptyset$, otherwise. Consequently the discretization scheme takes into account the proximity of segment s and provides a $(1+3\varepsilon)$-approximation of the optimal path from the segment source s to a node t.

Distance Propagation

Again, to be able to compute shortest paths by using Bushwhack strategy, we have to prove that two shortest paths from s to arbitrary points on $\mathcal{P}$ do not intersect in the interior of a face (Lemma 3.1) and adapt the initialization step where we create intervals on the edges of the face(s) containing s. Notice that these intervals will contain both edge-using and face-crossing segments from s to the nodes, each node will be joined to the point of s defining minimum distance. Consequently in the faces containing f we can find two different types of edges: those emanating from the endpoints of s and the rest that are perpendicular to segment s (See Figure 3).

Lemma 3.1 *Let s be a segment source and p, p' two points of $\mathcal{P}$. Denote $\Pi_{s,p}$ and $\Pi_{s,p'}$ the shortest paths from s to p and to p', respectively. Shortest paths $\Pi_{s,p}$ and $\Pi_{s,p'}$ do not intersect in the interior of any face, except for in s.*

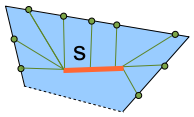

Figure 3. Shortest paths from the orange segment source s to some green nodes.

Proof. We omit the proof due to lack of space.

According to Lemma 3.1, Theorem 2.1 and Lemma 2.1 and using the Bushwack strategy complexity, we can conlcude the following Theorem.

Theorem 3.2 *We can obtain a $(1+3\varepsilon)$-approximate distance defined by a segment source s in $O(mn\log(mn))$ time and $O(mn)$ space, where $m \in O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$.* ■

3.3. From Polygonal Line Source to Node

The distance function defined by a polygonal chain s defined by r' segments, is obtained by considering s as the union of all the segments s_i conforming it. In this case we define a vicinity around s using the already provided definitions.

The discretization scheme follows the same guidelines, however, now $V(s)$ may intersect two or more times the same edge e. Consequently when we consider $e \setminus V(s)$ we can obtain more than two sub-edges and some of them are not incident to a vertex of $\mathcal{P}$. In subedges $e' = \overline{uv}$ not incident to a vertex we place one Steiner point at each endpoint u and v and then we use the logarithmic scheme along e' using first u and next v, in both cases until we get point $v_{e'}$ (See Figure 4). All the Steiner points on these edges should be placed the first time that e is considered. Thus we can provide the following Lemma and Theorem.

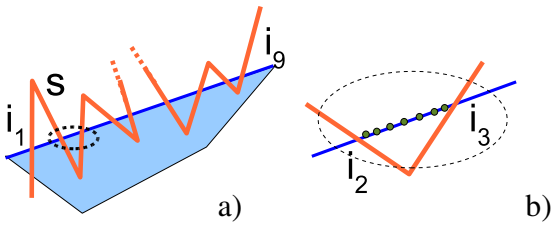

Figure 4. a) Polygonal line source intersecting an edge e, in dark blue, nine times. b) Detail of subedge $\overline{i_2 i_3}$ with its corresponding Steiner points.

Lemma 3.2 *The number of Steiner points generated on an edge intersected r' times by the Vicinity of s is $O(r'\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ Steiner points.* ■

Theorem 3.3 *The graph contains a $(1+3\varepsilon)$-approximation of any optimal path from a polygonal line source s to a node t.* ■

A polygonal region s is a connected region of $\mathcal{P}$ whose boundary, ∂s, is a closed polygonal chain. The distance defined by s is 0 in s and the distance function defined by ∂s in $\mathcal{P} \setminus s$. Therefore, its distance can be computed considering the polygonal line source defining its boundary and propagating the distance to $\mathcal{P} \setminus s$ and giving distance 0 in s. It is summarized in next result.

Property 3.1 *Distances from a polygonal region r can be computed by considering its boundary polygonal line $\partial(s)$.*

Distance Propagation

Let us see that any two shortest paths from s to two different points of $\mathcal{P}$ can not intersect in the interior of any face, except for in a point of s.

Lemma 3.3 *Let s be a polygonal line source and p, p' two points of $\mathcal{P}$. The shortest paths from p, p' to s do not intersect in the interior of any face, except for perhaps in s.*

Proof. Due to the lack of space, we omit the proof.

Therefore, we can use Bushwack strategy by adapting the initialization step where we consider each segment s_i conforming the polygonal line s independently. Let $s_1, \ldots, s_k$ be the segments conforming s, we create intervals $I_{s_i,-,e'}$ containing the nodes closer to s_i than to any other already considered segment s_j, $j < i$. When a new segment s_j is considered the previously computed intervals may be modified as it happens during the propagation step.

This method can also be used to compute distances from a polygonal regions according to Property 3.1. When considering a polygonal region, in the initialization step we will only define intervals in $\mathcal{P} \setminus s$.

Thus by using Lemma 3.3, Lemma 3.2 and making the logical assumption that the polygonal s intersects each edge a constant number of times, we can state the following Theorem.

Theorem 3.4 *We can obtain a $(1+3\varepsilon)$-approximate distance defined by a polygonal line or polygon source s in $O(mn\log(mn))$ and $O(mn)$ time and space complexity, where $m \in O(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$.* ■

3.4. Polygonal Obstacles

Given the polyhedral surface $\mathcal{P}$, with obstacles represented as several surface faces, edges and vertices, we

can adapt the algorithm to compute approximate weighted shortest paths that can go along obstacle edges, but not through them. Obstacle edges have their nodes some how duplicated, we use a copy of the nodes for each face. The shortest path arriving at a node on an obstacle edge can not go through the edge, it can only be propagated along the edge or back to the face it comes from. Thus the only thing that we have to change are the edges of the graph.

These modifications do not affect the discretization scheme nor the proofs of the provided Lemmas. Consequently we can provide the following theorem which summarizes the results obtained until now, assuming $0 < \varepsilon < 1$ and that source s intersects each edge a constant number of times.

Theorem 3.5 *Let $\mathcal{P}$ be a weighted triangulated polyhedral surface with generalized obstacles and s be a generalized source on $\mathcal{P}$, a $(1+3\varepsilon)$-approximate distance function from s can be obtained by using Bushwack strategy in $O(mn\log(mn))$ time with $m \in O(\frac{1}{\varepsilon}\ \log\frac{1}{\varepsilon})$.* ■

4. Implicit Distance Field Computation

When we consider a set of sites S we can use the graph to obtain their distance field, which for any node gives the approximate shortest path distance to its nearest site of S.

The scheme provided in Section 2 contemplates the case when more than one site is considered. Thus, it can be used to obtain a graph where we will obtain a $(1+3\varepsilon)$ - approximation of the distance field. In this case we have to proceed as it is explained for the case of polygonal sites in Section 3.3. When we place Steiner points on an edge e that is intersected by $V(S) = \cup_{s\in S}V(s)$ we have to handle each sub-edge of e apart and use the logarithmic scheme more than twice. Consequently the number of Steiner points, may increase.

Lemma 4.1 *The number of Steiner points generated on an edge intersected r' times by the Vecinity of the sources in S contains $O(r'\frac{1}{\varepsilon}\log\frac{1}{\varepsilon})$ Steiner points.* ■

The results related to the ε-approximation provided in previous sections remain true, consequently, we can state the following Theorem.

Theorem 4.1 *The graph allows the computation of a $(1+3\varepsilon)$-approximate distance field defined by a set of generalizes sites S.* ■

4.1. Distance Propagation

Bushwhack algorithm can be used to obtain the distance field of a set of generalized sites as it is proven in the following lemma.

Lemma 4.2 *Let us consider a set of generalized sites S and two points of $\mathcal{P}$, p and p'. Denote $\Pi_{S,p}$ the shortest path that joins p to the closest site of S to p, and $\Pi_{S,p'}$ the one joining S with p'. Then $\Pi_{S,p}$ and $\Pi_{S,p'}$ do not intersect in the interior of a face.*

Proof. We omit the proof due to space constraints.

We adapt Bushwack strategy to compute the distance field by propagating the distance function of all the sites at once. It is achieved by considering all the sites in the initialization step. We consider the first site and define intervals on the faces it intersects. Next we consider the second site and again define its intervals taking into account the previously defined ones which can be modified. Since intervals emanating from different sites are stored in the same list, we propagate their distance field. As it is typically done by Bushwack, at each step the shortest path with minimal cost is propagated. The difference is that, now, paths may come from different initial sites. The time and space complexity does not increase, and is again $O(mn\log(mn))$ and $O(nm)$, whenever each edge of e is intersected by S a constant number of times.

Theorem 4.2 *A $(1+\varepsilon)$-approximate distance filed defined by a set S of generalized sites on a triangulated weighted polyhedral surface $\mathcal{P}$ with obstacles can be obtained by using Bushwack in $O(mn\log(mn))$ time where $m \in O(\frac{1}{\varepsilon}\ \log\frac{1}{\varepsilon})$.* ■

5. Distance and Shortest Path Computation

When the propagation of the distance function from a generalized source to the nodes has concluded, the distance to any point on $\mathcal{P}$ can be obtained. If we are given a set of sites S, we can compute the shortest path distance to the closest site by using the distance field defined by S.

5.1. Influence Regions

The approximate distance function in the interior of the faces is computed by propagating the shortest paths arriving at the nodes of the discrete graph to the face points. Given a node v contained on edge e, Bushwhack algorithm defines intervals $I_{v,e,e'}$ associated to v, e and the edges $e' \neq e$ of the face(s) containing v or opposite to v when v is a vertex. These intervals contain the nodes (Steiner points and vertices) that, according to the already visited nodes, may be reached by a shortest path that leaves from v. Now, for each interval $I_{v,e,e'}$ we define the *influence region* of I, denoted R_I, on the face f containing e, e' and v, as the set of all points of f that, according to $I_{v,e,e'}$ can be reached by a shortest path emanating from v.

To compute R_I for a given interval $I_{v,e,e'}$ on face f with edges e, e' and e'', we consider: a) the already visited nodes v_l, v_r, of e placed contiguously on the left and right of v, respectively; b) the nodes v'_l, v'_r of e' placed contiguously on the left and right of $I_{v,e,e'}$, if $I_{v,e,e'}$ contains the leftmost(/rightmost) vertex of e', v'_l(/v'_r) is the corresponding vertex of e'. Region R_I is the polygonal region of vertices $\{v_l, v'_l, v'_r, v_r\}$. Consequently R_I is a region of at most four vertices. Notice that the influence region R_I of an interval can be computed in $O(1)$ during Bushwack algorithm.

Property 5.1 *The influence region R_I of an interval can be computed in $O(1)$ with the information computed during Bushwack algorithm.*

5.2. Distance Computation

The shortest path distance from any point $q \in f$ to the source s can be obtained by finding the node v_m on the edges of f defining the minimum distance value. If $d_{s,v}$ denotes the distance function defined by s and node v, we have $d_{s,v}(q) = d_s(v) + w|\overline{vq}|$, where w is the weight of f. Notice that to determine v_m, those nodes whose influence region does not contain q can be directly discarded. The cost of the shortest path from a point of $\mathcal{P}$ to its nearest site can be obtained by standard methods.

Approximation Analysis

We use a graph that provides $(1 + 3\varepsilon)$-approximate distances from s to the nodes, let us now bound the error produced when obtaining the distance from s to an arbitrary point t of $\mathcal{P}$.

Proposition 5.1 *We can obtain $(1+4\varepsilon)$-approximate shortest paths from a generalized source s to an arbitrary target point $t \in \mathcal{P}$.*

Proof. The proof is only hinted due to lack of space. It uses Lemma 2.3 and shows that any normalized path Π' can be $(1+3\varepsilon)$-approximated on the graph. It is done by bounding the length of the subpath from the last bending point to the target by $\varepsilon||\Pi'||$. ■

When a set of sites S is given, the distance to the closest site can be computed by using the $(1 + 4\varepsilon)$-approximate distance filed. By using this result and a proof similar to that of the previous proposition, we can provide the following theorem.

Theorem 5.1 *Given a set of generalizes sites S, we obtain a $(1 + 4\varepsilon)$-approximation of their distance field. When $S = \{s\}$ we obtain a $(1+4\varepsilon)$-approximation of its distance function.* ■

Distance fields are mainly important because they define Voronoi diagrams. When distances are exactly computed the points that are equidistant from two sites conform the Voronoi diagram bisectors. Since now we are working with approximate distances, we should study where the exact bisectors are when the approximated distance is used. Let d_s denote the $(1 + 4\varepsilon)$-approximate distance function and $\hat{d}_s$ the exact distance function for a source s. Let p be a point of an exact bisector determined by sites s_0 and s_1, thus, $\hat{d}_{s_0}(p) = \hat{d}_{s_1}(p)$. Therefore the following proposition can be given.

Proposition 5.2 *The error produced on a Voronoi diagram bisector tends to 0 when ε tends to 0.*

Proof. It can be shown that $|d_{s_0}(p) - d_{s_1}(p)| < 6\,\varepsilon\,\hat{d}_{s_0}(p)$. ■

5.3. Shortest Path Computation

After computing the distance function for a source s, we can obtain the shortest path from s to an arbitrary point q on a face f of $\mathcal{P}$. It suffices to use a backtracing technique and store, during Bushwhack algorithm, in each node v a pointer to the previous node in the path that goes from s to v. Once the distance function is computed, the path is obtained by first determining the node v of f providing the minimum distance at q. From node v we go back to node v' stored in the pointer associated to v. We keep jumping until we arrive at source s. The shortest path can be obtained in $O(m + \overline{n})$ time where $\overline{n}$ is the number of segments conforming the path. To obtain the path we store the pointers to the nodes.

When a set of sources S is considered, the shortest path to the closest site can be obtained by using the same strategy by using the distance field instead of the distance function.

6. Discrete Distance Functions

In this Section we overview our algorithm to compute, by using graphics hardware, discrete distance functions on a weighted polyhedral surface. For more details see [FS], where a similar process is used for computing discrete distance functions on non weighted polyhedral surfaces.

Given a point q and a node v belonging to the same face f of the polyhedral surface $\mathcal{P}$ with weight w, the distance vector $\overrightarrow{d_q}$ of q with respect to v is the vector joining v to q. Observe that the cost of the path from s to q given by the shortest path from s to v and a straight line from v to q is $d_v(q) = D_s(v) + w|\overline{vq}|$. Notice that $|\overline{vq}|$ can be computed as the length of $\overrightarrow{d_q}$. Distance vector properties allow us to compute $d_v(q)$ interpolating the distance vectors of the vertices of $R_{v,f}$.

A *planar parametrization* of $\mathcal{P}$ is a bijective function that maps $\mathcal{P}$ into a bounded region of the plane $\mathcal{R}$. By using the

parameterization of $\mathcal{P}$, we discretize the region $\mathcal{R}$ of the xy-plane as a rectangular grid of size $W \times H$, which induces a discretization on the triangles of $\mathcal{P}$. When the parameterization maintains the triangles shape, the discretization error in the xy-plane matches with the discretization error on $\mathcal{P}$ [CGAL, CHM, FG].

The parameterization and the surface discretization are used to obtain a discrete representation on $\mathcal{R}$ of the distance function defined by weighted shortest paths on $\mathcal{P}$. We keep track of the explicit representation of the distance function while it is propagated along the surface $\mathcal{P}$ with the Bushwhack algorithm without increasing its computational cost. When a node v is first visited we compute its intervals, the influence regions R_I and the distance function it defines to all the grid points of R_I. Since grid points can be contained in the influence region of different nodes, during the process we store in each grid point of the xy-plane the minimum of the obtained distances.

The distance to the grid points of R_I is computed by using distance vectors, OpenGL and graphics hardware. The OpenGL pipeline triangulates the input polygons, processes the triangle vertices and rasterizes the triangles into fragments with linear interpolation. When R_I is first computed it is painted with OpenGL and mapped to $\mathcal{R}$ with the planar parameterization by using a vertex shader. After the rasterization we have a set of fragments representing 2D grid points $\{q \in \mathcal{R}\}$. We use a fragment shader to compute at each fragment its corresponding $d_v(q)$ as $D_s(v) + w\|\overrightarrow{d_q}\|$, which is normalized into $[0, 1]$ and set as the fragment depth value. Distance $D_s(v)$ and weight w are sent to the fragment shader and $\overrightarrow{d_q}$ is obtained during the rasterization process from the distance vectors to the vertices of $R_{v,f}$. Finally, to store the minimal distance to each point we use the depth test. Since $R_{v,f} \subset f$ and we have at most $m + m'$ nodes per edge, each face can be painted $O(m+m')$ times. The extra time needed to paint all $R_{v,f}$ is $O((m+m')HW)$, consequently, the complexity of the Bushwack algorithm is not increased.

6.1. Visualization on the polyhedral surface

Once we have obtained a discrete representation of any of the mentioned Voronoi diagrams in the color buffer, we can transfer the values of this buffer to a texture. The texture is an explicit discrete representation of the Voronoi diagram and by using texturing methods, with the already used planar parametrization, the Voronoi diagram can be visualized on the polyhedral surface.

Distance functions can also be visualized on the polyhedral surface by transferring the values of the depth buffer to a depth texture. Since distances vary from 0 to 1 they can be used to weight the color of the pixels. If the white color is used, pixels are painted in a green gradation from black (distance 0) to light green (distance 1) according to the distance function values.

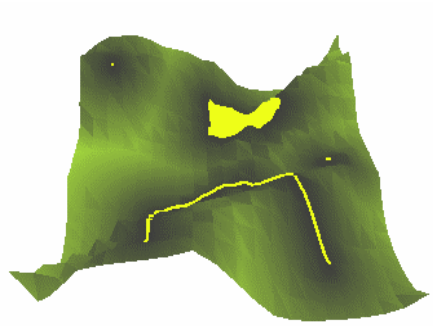

Figure 5. The distance field defined by a set of four generalized sites.

7. Discrete k-Order Voronoi Diagrams

We can obtain a discrete representation of the closest Voronoi diagram by using the algorithm to obtain the implicit distance field explained in Section 4 combined with the process given in Section 6. We paint the influence regions associated to different sources in different colors. As a result we obtain the Voronoi diagram represented on a texture representing $\mathcal{R}$, that can be visualized on the polyhedral surface by using texturing methods. Moreover, if we compute the set of distance functions for all r sources, we can obtain (and visualize on the polyhedral surface) any high-order Voronoi diagram [FS]. The closest and furthest Voronoi diagrams are obtained as the lower and upper envelope of the distance functions and the k-order Voronoi diagrams, $k \in \{2, \cdots, r-2\}$, are obtained using a "depth peeling" technique.

The time and space complexities of the algorithm that obtains the closest Voronoi diagram by using the Bushwack algorithm are $O((nm + n'm')\log(nm + n'm'))$ and $O(nm + n'm)$, respectively. The time needed to compute the r distance functions and store them in a texture is $O(r(nm + n'm')\log(nm + n'm'))$. Finally, the extra time needed to obtain the closest and furthest Voronoi diagrams from the distance functions is $O(rHW)$, and for the $k-$order Voronoi diagram is $O(rkHW)$.

The results obtained are approximated, and two different types of error can be seen. One is the discretization error, which depends on the discretization size. It can be reduced using the fact that the bigger the grid size $W \times H$, the smaller the error produced. The other is due to floating errors, which are specially related to the depth buffer and depth texture precision. The 32-bit precision is sufficient to store the normalized distances which take values in the interval $[0, 1]$. This can be specially seen when computing k-order Voronoi diagrams, where many distances have to be compared. In the rest of the applications it is not visible.

8. Experimental results

We have implemented the proposed methods using C++ and OpenGL for the special case of polyhedral terrains. All the images have been carried out on a Intel(R) Pentium(R) D at 3GHz with 1GB of RAM and a GeForce 7800 GTX/PCI-e/SSE2 graphics board.

In Table 1 we present some experimental results, obtained by considering terrains without obstacles, a set S of six sites (one point, two segments, two polygonal lines and one polygon), $\varepsilon = 0.5$, and a grid to discretize the domain of size 500×500. We present execution times for weighted terrains with weights randomly generated between zero and five. In the table we specify the number of terrain faces, n, and the number of faces intersected by the sites n'. We provide the total number of Steiner points. Next, we give the time needed to compute the distance field using the Bushwack strategy and finally the time needed to compute the six distance functions. Notice that the time needed to obtain the six distance functions is about six times that needed to obtain the distance field. The extra time needed to obtain, from the already computed distance fields, the closest or furthest Voronoi diagram is 0.08(s), the 4th nearest site to each point is 0.3(s) and the 5th nearest site is 0.35(s).

n	n'	N. Steiner Points	D. Field	D. Functions
800	73	24449	5.6 (s)	30 (s)
5000	145	135670	28 (s)	147 (s)
10000	221	260599	54 (s)	281 (s)
20000	271	533677	102 (s)	582 (s)
45000	240	1228163	268 (s)	1408 (s)

Table 1. Distance functions computation.

Figure 6 show some examples of Voronoi diagrams for generalized sources on polyhedral terrains with n =800 faces obtained with our implementation. The generalized sources, except for the polygon sources interior, are painted on the terrain surface and the remaining points of the surface are colored according to the Voronoi region they belong to.

The error produced by the 32-bit precision of the depth buffer can be seen in Figure 6 c). Isolated pixels are painted in the color of the regions adjacent to the region they belong to.

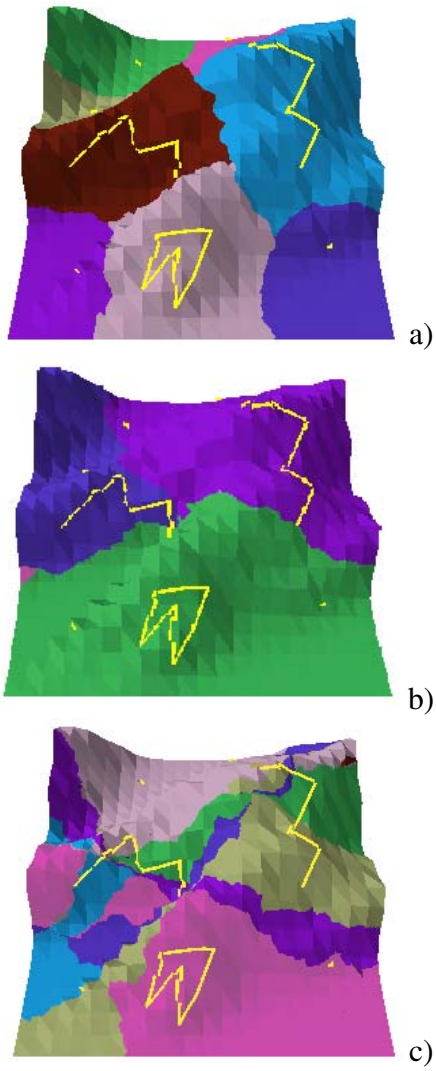

Figure 6. A set of eight generalized sources on a weighted terrain with their: a) Closest Voronoi diagram. b) Furhtest Voronoi Diagram. c) 6th-nearest diagram.

9. Conclusions and Future Work

We have presented a discretization scheme that provides $(1+\varepsilon)$-approximate shortest paths on polyhedral surfaces. We have proved that the Bushwhack strategy can be used to obtain the shortest paths on the graph. The algorithm is extended to obtain the distance field of a set of generalized sites. Next, a way to obtain an explicit representation of the distance function is provided. As applications we provide a way to directly obtain a discrete Voronoi diagram from the implicit distance field, and a more general technique, which from the distance functions of all the sources in S the closest, furthest or any $k-$order Voronoi diagram can be obtained. Finally some experimental results obtained with our implementation that works for polyhedral terrains with generalized sources and obstacles are presented.

As future work we will solve some facility location problems such as the 1-Center, 1-Median. The 1-Center is the point minimizing the maximal distance of a point on the polyhedral surface to a set of sites, and the 1-Median is the point minimizing the sum of distances of a point on the terrain to the sites.

References

[ALMS] Aleksandrov, L., Lanthier, M., Maheshwari, A., Sack, J.-R.: An ε - approximation algorithm for weighted shortest paths on polyhedral surfaces. In Scandinavian Workshop on Algorithm Theory'98 (1998) 11–22.

[AKMV] P. K. Agarwal, P.K., Krishnan, S., Mustafa, N., Venkatasubramanian, S.: Streaming geometric optimization using graphics hardware. 11th European Symp. on Algorithms (2003) 544–555.

[AMS1] Aleksandrov, L., Maheshwari, A., Sack, J.R: Approximation algorithms for geometric shortest path problems. STOC '00: Proceedings of the 32nd annual ACM symposium on Theory of computing (2000) 286–295.

[AMS2] Aleksandrov, L., Maheshwari, A., Sack, J.R.: Determining approximate shortest paths on weighted polyhedral surfaces. J. ACM, **52**(1) (2005) 25–53.

[AKOV] Aronov, B., van Kreveld, M. J., van Oostrum, R., Varadarajan, K. R.: Facility location on a polyhedral surface. Discrete & Computational Geometry, **30(3)** (2003) 357–372.

[CHM] , Carr, N., Hart, J., Maillot J.: The solid map: Methods for generating a 2-D texture map for solid texturing. Proc. Western Computer Graphics Symposium (2000) 179–190.

[CGAL] Computational Geometry Algorithms Library User and Reference Manual, Chapter 32 Planar Parameterization of Triangulated Surface Meshes, http://www.cgal.org/Manual/3.2/dochtml/cgalmanual /contents.html (accessed 10 February 2008)

[FEKKVS] FAN, Q., EFRAT, A., KOLTUN, V., KRISHNAN, S. and VENKATASUBRAMANIAN, S., 2005, Hardware-Assisted Natural Neighbor Interpolation, *Proc. 7th Workshop on Algorithms Engineering and Experimentation*.

[FG] Fisher, I., Gotsman, C.: Fast approximation of high order voronoi diagrams and distance transforms on the GPU. Journal of Graphics Tools. **11(4)** (2006) 39–60.

[FG] Floater, M., Hormann, K.: Surface Parameterization: a Tutorial and Survey. Advances in multiresolution for geometric modelling (2005) 157–186.

[FS] Fort, M., Sellarès, J.A.: Generalized Higher-Order Voronoi Diagrams on Polyhedral Surfaces. 4th Int. Symp. on Voronoi Diagrams in Science and Engineering, (2007) 74–83.

[GHLM] GUESGEN, H.W., HERTZBERG J., LOBB R. and MANTLER, A., 2001, First steps towards buffering fuzzy maps with graphics hardware, *Proc. FOIS Workshop on Spatial Vagueness, Uncertainty and Granularity*.

[HKLMC] HOFF III, K.E., KEYSER, J., LIN, M. C., MANOCHA, D. and CULVER, T., 1999, Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware. SIGGRAPH, pp 277–286.

[KCC] V. Koltun, Y. Chrysanthou, D. Cohen-Or, Hardware-accelerated from-region visibility using a dual ray space, *ACM. Proceedings of the 12th EW on Rendering Techniques*, (2001), pages 205–216.

[KMV] KRISHNAN, S., MUSTAFA, N. H. and VENKATASUBRAMANIAN S., 2002, Hardware-Assisted Computation of Depth Contours. *13th ACM-SIAM Symposium on Discrete Algorithms*, pp 558-567.

[MP] Mitchell, J.S.B., Papadimitriou, C.H.: The weighted region problem: finding shortest paths through a weighted planar subdivision. J. ACM. **38(1)** (1991) 18–73.

[Mou] Mount, D.M.: Voronoi diagrams on the surface of a polyhedron. Technical report, University of Maryland. (1985).

[MKV] MUSTAFA, N., KRISHNAN, S., VARADHAN, G. and VENKATASUBRAMANIAN S., 2006, Dynamic simplification and visualization of large maps, *International Journal of Geographical Information Science*, **20(3)**, pp 273–302.

[OpenGL] SEGAL, M. and AKELEY, K., The OpenGL Graphics System: A Specification, (2004) http://www.opengl.org/documentation/specs/version2.1 /glspec21.pdf (accessed 10 February 2008)

[OLGHKLP] OWENS, J.D., LUEBKE, D., GOVINDARAJU, N., HARRIS, M., KRGER, J., LEFOHN A.E. and PURCELL, T.J., 2007, A Survey of General-Purpose Computation on Graphics Hardware, *Computer Graphics Forum*, **26(1)**, pp 80–113.

[PBMH] PURCELL, T., BUCK, I., MARK, W. and HANRAHAN, P., 2002, Ray tracing on programmable graphics hardware, *ACM Transactions on Graphics*, **21(3)**, pp 703–712.

[SR] Sun, Z., Reif, J.: On finding approximate optimal paths in weighted regions. Journal of Algorithms. **58(1)** (2006) 1–32.