Precise Hausdorff Distance Computation between Polygonal Meshes

Michael Bartoň[a], Iddo Hanniel[a,b], Gershon Elber[a], Myung-Soo Kim[c]

[a] *Technion – Israel Institute of Technology, Haifa 32000, Israel*
[b] *SolidWorks Corporation, 300 Baker Avenue, Concord, MA 01742 USA*
[c] *Seoul National University, Seoul 151-744, Korea*

Abstract

We present an exact algorithm for computing the precise Hausdorff distance between two general polyhedra represented as triangular meshes. The locus of candidate points, events where the Hausdorff distance may occur, is fully classified. These events include simple cases where foot points of vertices are examined as well as more complicated cases where extreme distance evaluation is needed on the intersection curve of one mesh with the medial axis of the other mesh. No explicit reconstruction of the medial axis is conducted and only bisectors of pairs of primitives (i.e. vertex, edge, or face) are analytically constructed and intersected with the other mesh, yielding a set of conic segments. For each conic segment, the distance functions to all primitives are constructed and the maximum value of their low envelope function may correspond to a candidate point for the Hausdorff distance. The algorithm is fully implemented and several experimental results are also presented.

Key words: Hausdorff distance, polygonal mesh, geometric algorithm, bisector surface, medial axis, low envelope

1. Introduction

Measuring the distance between different geometric objects is often needed as a first step for other geometric computations such as collision detection, robot motion planning and haptic rendering [1, 2, 3]. Thus the development of efficient algorithms for distance computation is very important for improving the performance of related geometric problems. Among many different distance measures, the Euclidean distance and recently the Hausdorff distance have attracted considerable research attention in geometric modeling, computational geometry, and computer graphics [3]–[15].

An extensive family of applications for the Hausdorff distance may also be found in visualisation, namely in the case of surface remeshing [16, 17, 18]. Typically, having a dense starting mesh, its simplification is required to be within a certain distance measure. The one-sided Hausdorff distance from A to

B, $h(A,B)$, measuring the maximum deviation of an object A from the other object B, is defined as

$$h(A,B) = \max_{a \in A} d(a,B),$$

where $d(a,B) = \min_{b \in B} \|a-b\|$ is the minimum Euclidean distance from a to B. Since the one-sided distance is non-symmetric, the Hausdorff distance $H(A,B)$ is defined as

$$H(A,B) = \max\left(h(A,B), h(B,A)\right),$$

which introduces all metric properties. Consequently, the Hausdorff distance measures the shape similarity between two geometric shapes [9].

For planar convex polygons, Atallah [12] developed a linear-time algorithm for computing the Hausdorff distance. Llanas [13] considered two approximate algorithms for non-convex polytopes. Cignoni et al. [14] reported an efficient algorithm for approximating the Hausdorff distance between a non-convex polygonal mesh and a set of point samples on a nearby polygonal mesh of similar shape. Another algorithm which approximates the Hausdorff distance within a user defined tolerance is proposed in [26]. An octree-based bi-hierarchical search that quickly finds regions of higher geometrical distances is performed. These $3D$ regions are than further subdivided and simple accept/reject tests purge regions that can not possess the Hausdorff distance.

In a recent work, Tang et al. [15] presented a real-time algorithm for approximating the Hausdorff distance between two triangular meshes within a given error bound.

There are also some previous results on the more general case for freeform shapes. Jüttler [19] presented a method to estimate bounds on the Hausdorff distance between two freeform/implicit curves. Alt and Scharf [10, 11] developed an exact algorithm for the Hausdorff distance between planar rational curves. In a recent work, Elber and Grandine [20] extended this result to a more general problem for rational curves/surfaces. The performance of the proposed algorithm is quite efficient for the case of planar/space curves. However, for the Hausdorff distance between a freeform surface and a curve or another surface, the algorithm is too slow.

IDDO: Please, also explain in more detail lower envelope in this paragraph.

In this work, we consider a slightly simpler problem for two polygonal meshes and present an exact algorithm for the Hausdorff distance computation. From a theoretical point of view (in the sense of computational geometry), Alt et al. [21] also considered this problem. They first explained a simple approach based on the lower envelope of squared distance function surfaces over each triangle. The worst case time complexity is $\mathcal{O}(n^5)$, where n is the number of triangles. (Employing a very theoretical machinery, they also derived a randomized algorithm of $\mathcal{O}(n^{3+\epsilon})$ expected time complexity, for any $\epsilon > 0$. However, their randomized algorithm has never been implemented due to its non-trivial intricacy.) Even their simple approach has never been implemented. Compared with lower envelopes of surfaces, it is considerably easier to implement an algorithm that is based on a low envelope of curves. In this paper, we present such

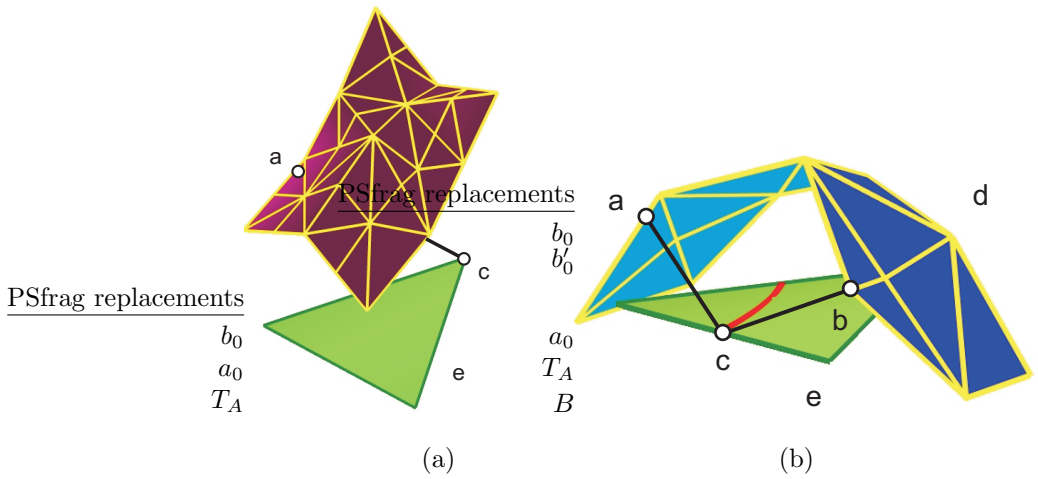

Figure 1: Triangle T_A of mesh A where the one-sided distance $h(A, B)$ is attained. (a) The footpoint $b_0 \in B$ is either unique (Case 1) or (b) there exist at least two footpoints (Case 2). In the latter case, a_0 lies on the intersection curve (red) between T_A and (a part of) the medial axis of B.

an algorithm with its worst-case time complexity $\mathcal{O}(n^4 \log n)$ with a complete implementation.

Without loss of generality, we assume that

$$h(A, B) \geq h(B, A),$$

and the Hausdorff distance $H(A, B)$ is realized at two points $a_0 \in A$ and $b_0 \in B$ such that

$$H(A, B) = \max_{a \in A} d(a, B) = d(a_0, B) = \min_{b \in B} \|a_0 - b\| = \|a_0 - b_0\|. \qquad (1)$$

Note that the point b_0 is the *footpoint* of a_0 to the other object B. There are two cases to consider: (1) the footpoint b_0 is unique or (2) there are multiple (typically two) footpoints $b_0, b_0' \in B$, i.e., the point a_0 is on the medial axis of B, see Fig. 1. Recall the definition of *medial axis* of an object as a locus of all points having more than one closest point on the object's boundary.

An important advantage in considering the special case of two polyhedra is that, in the first case, the Hausdorff distance is realized at a mesh vertex a_0 as the minimum distance to the other polygonal mesh B. (There are some degenerate cases where a_0 can be located in an edge/face interior; but these cases can easily be reduced to a case where $a_0 \in A$ is a vertex or $a_0 \in A$ is located on a medial axis of the other mesh as discussed in Section 2.) Moreover, in the second case, the point a_0 of one mesh is on a medial axis of the other mesh B. Since the medial axis of a polygonal mesh consists of quadratic and linear surfaces, the problem essentially reduces to solving quadratic equations and searching some

extreme distance points on conic curve segments. Unlike [14, 15], the presented method does not require any point sampling or subdivision of input triangles; and hence the *precise*[1] Hausdorff distance is computed.

The rest of the paper is organized as follows. Section 2 fully classifies the locus of candidate points, events where the Hausdorff distance may occur. Consequently, in Section 3, we develop the algorithm that inspects those candidate locations and computes the precise Hausdorff distance. The complexity of the proposed algorithm is analyzed in Section 4. Several examples are given in Section 5 and finally the paper is concluded in Section 6.

2. Candidate points

Let A and B be two triangular meshes and let $h(A, B)$ be the one-sided Hausdorff distance from A to B. In this section, we introduce a set of *candidate points*, a locus of points in A where $h(A, B)$ may be attained, and we prove that this set is sufficiently described by one of two cases:

- **Case 1:** a_0 is a vertex of A and the maximum of the Euclidean distance is attained between a_0 and an object - face, edge or vertex - of B.
- **Case 2:** a_0 is a point on the intersection curve between a triangle of A and a medial axis of B. This intersection curve is (a segment of) a conic section and the point a_0 maximizes the distance between this conic section and the mesh B.

Before we come to the proof, we need to recall some basic notions from the Voronoi theory [22].

2.1. Preliminaries

Definition 1. *Given a set of* $(n+1)$ *objects* O^0, O^1, ... , O^n *in* $\mathbb{R}^3$, *the* Voronoi cell *(VC) of an object* O^i, *denoted as a* base object, *is the set of all points in* $\mathbb{R}^3$ *closer to* O^i *than to* O^j, $\forall\ j \neq i$. *The* Voronoi Diagram *(VD) is then the union of the Voronoi cells of all* $(n+1)$ *objects.*

Definition 2. *The locus of points that are equidistant from* O^i *and* O^j, *for some* $j \neq i$, *is called (the* O^iO^j*)* bisector.

In general, bisectors in $\mathbb{R}^3$ are surfaces. For some special mutual limit positions of objects O^i and O^j, the bisector may degenerate either to a curve, or to the whole $\mathbb{R}^3$ (if $O^i \equiv O^j$). Note that the *medial axis*, the union of all boundaries of the Voronoi cells, consist of *bisector surfaces* [27].

In the case of simple objects like polygonal meshes, the geometric entities dealt with are points, lines and planes in $\mathbb{R}^3$. Therefore, all the bisectors between these entities are quadric surfaces and thus their intersections with a plane are

[1] By precise we mean up to floating point numbers.

conic section curves in $\mathbb{R}^3$. The fact that these bisectors are quadrics is obvious from the prescription of the (squared) distance function, which measures the distance from a point, a line or a plane, respectively (hereafter bold letters denote vector values):

- **The squared distance function from a point:**

$$d^2 = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2, \tag{2}$$

  where $\boldsymbol{p_0} = (x_0, y_0, z_0)$ is the point from which we compute the distance, and d is the computed distance of the Euclidean point $\boldsymbol{p} = (x, y, z) \in \mathbb{R}^3$ from $\boldsymbol{p_0}$.

- **The squared distance function from a plane:**

$$d^2 = \frac{(\alpha x + \beta y + \gamma z + \delta)^2}{\alpha^2 + \beta^2 + \gamma^2}, \tag{3}$$

  where $(\alpha, \beta, \gamma, \delta)$ are the plane's coefficients.

- **The squared distance function from a line:**

$$d^2 = \langle \boldsymbol{p} - \boldsymbol{c_0}, \boldsymbol{p} - \boldsymbol{c_0} \rangle - \frac{\langle \boldsymbol{p} - \boldsymbol{c_0}, \boldsymbol{d_0} \rangle^2}{\langle \boldsymbol{d_0}, \boldsymbol{d_0} \rangle}, \tag{4}$$

  where $\boldsymbol{c_0} + t\boldsymbol{d_0}$ is the *parametric* line equation and d is the distance of the point $\boldsymbol{p}$ from the line.

Lemma 1. *The bisector surfaces between points, lines and planes in $\mathbb{R}^3$ are quadric surfaces.*

Proof. By definition, the bisector is the locus of all equidistant points. Thus, the result of equating two squared distance functions and eliminating d^2 gives the implicit representation of the bisector. Since both sides of the equation hold only, at most, degree-2 polynomials (in x, y, z), a (at most) degree-2 implicit surface is obtained. □

Note that planes are degenerate quadric surfaces and indeed the plane-plane and point-point bisectors are, in fact, planes. Hereafter, we denote by a mesh-bisector, the bisectors between points, lines and/or planes.

Corollary 1. *The intersection of a plane and a mesh-bisector is a conic section curve.*

Proof. Since the bisector surface is a quadric, its intersection with a plane is a conic section curve. □

Remark 1. *In Section 3, we will measure the squared distance $d^2(t)$ from a conic curve segment $C_{ij}^k(t)$ (the intersection between a triangle T_A^k and an $O_B^i O_B^j$ bisector quadric) to an object O_B^l (a point, a line, or a plane). The squared distance function is computed by replacing the point $\mathbf{p}$ of (2)–(4) by a moving point $\mathbf{p}(t) = C_{ij}^k(t) = (x(t), y(t), z(t))$ along the conic section curve. Since the conic curve can be represented as a quadratic rational curve, the corresponding squared functions will be quartic rational functions in t.*

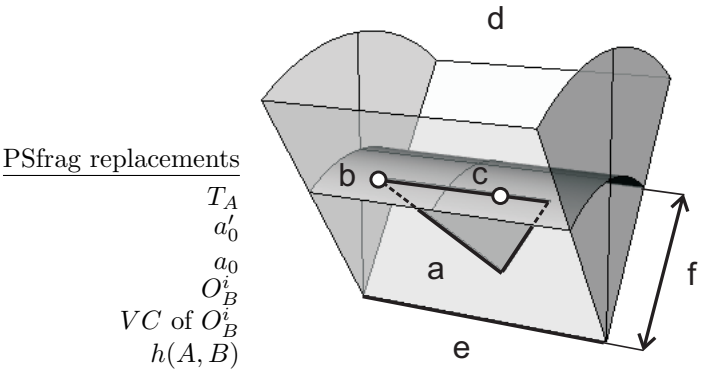

Figure 2: The Voronoi cell of edge $O^i_B \in B$ contains a_0, the point where the one-sided Hausdorff distance $h(A,B)$ is attained. Hence, the part of mesh A which is contained in this cell, $T_A = A \cap VC$, lies inside the cylinder of radius $h(A,B)$ whose axis is O^i_B.

2.2. Sufficient Subset of Candidate Points

In this subsection, we show that the one-sided Hausdorff distance can be computed by only checking two sets of candidate points; either vertices of A (Case 1) or the intersections of A with the medial axis of B (Case 2). An examination of these two cases detects some points $a_0 \in A$, $b_0 \in B$ such that $h(A,B) = d(a_0, b_0)$. We show that the examination of the candidate points described by these two cases is sufficient to compute the precise Hausdorff distance.

Example 1. Points defined in Cases 1 and 2 form a set of candidates at each of which the distance needs to be calculated. Note that the set of points where the Hausdorff distance may be attained can be an infinite set. Consider a mesh A consisting of only one triangle $T_A = A$ which is completely contained in the VC of an edge $O^i_B \in B$ such that the farthest edge of T_A is parallel to O^i_B, as depicted in Fig. 2. Then, every point of that edge is a candidate point. In this particular example, the Hausdorff distance is revealed at vertex a'_0 (Case 1).

Theorem 1. *The one-sided Hausdorff distance $h(A,B)$ from A to B can be attained either at a vertex of A (Case 1) or on the intersection of A with a medial axis of B (Case 2).*

Proof: Consider the VD of a mesh B. By definition, the VC of an object $O^i_B \subset B$ (face, edge or vertex) is the region in $\mathbb{R}^3$ where all points are closer to O^i_B than to any other object in B. Let us denote by $a_0 \in A$, the point where $h(A,B)$ is attained and let $a_0 \in VC$ of O^i_B for some i. Since the VCs are disjoint in their interior and their union covers $\mathbb{R}^3$, a_0 lies

- either strictly in the interior of the VC of O_B^i,
- or on its boundary.

If a_0 lies on the boundary, a_0 lies on a medial axis of B, since the boundary of any VC consists of bisector surfaces between two objects of B (Case 2).

What remains to prove is that, if a_0 lies in the interior of the VC of some O_B^i, then a_0 is

(i) directly a vertex of A, or if not then

(ii) there exists a vertex $a_0' \in A$ lying in the *interior* of VC such that $h(A,B) = d(a_0,B) = d(a_0',B)$, or if not then

(iii) there exists $a_0'' \in A$ lying on the *boundary* of the VC such that $h(A,B) = d(a_0,B) = d(a_0'',B)$.

Assume a_0 is an interior point of the VC of O_B^i. Since a_0 is the point where the one-sided Hausdorff distance $h(A,B)$ is attained, $d(a_0,B) \geq d(a,B)$ for any $a \in A$. Depending on the type of O_B^i (a vertex, an edge, a face), three cases need to be considered:

- **Vertex:** a_0 is on a sphere of radius $h(A,B)$ centered at a vertex O_B^i, and all the other points of $A \cap VC$ are contained in this sphere. In particular, the triangle of A to which a_0 belongs is contained inside the sphere and this can only happen if a_0 is a vertex of the triangle; case (i) above.
- **Edge:** a_0 lies on a cylinder of radius $h(A,B)$ whose axis is along an edge O_B^i, and all other points of $A \cap VC$ are contained in this cylinder. In particular, the triangle T_A to which a_0 belongs is contained in the cylinder, see Fig. 2. Assume that a_0 is not a vertex of the triangle. Since it is on the cylinder and no point of T_A is outside the cylinder, a_0 must be on a triangle edge of A that is parallel to the cylinder axis (and there are an infinite number of points on the triangle edge with the same distance from O_B^i). This edge is either entirely contained in VC (which immediately gives (ii)) or if not then it intersects a boundary of VC at some point a_0'' ($\Rightarrow$ (iii)).
- **Face:** In a similar manner, a_0 is on a plane parallel to the face plane O_B^i and offset by $h(A,B)$, and all the other points of $A \cap VC$ are contained between these two planes. If a_0 is not a vertex of a triangle then it is either a point on an edge of the triangle and the edge must be parallel to the plane of the triangle O_B^i, or it is an inner triangle point and the whole triangle T_A must be parallel to the plane of the face O_B^i. In both cases, a similar argument to the edge case shows that either there is a vertex inside the VC ($\Rightarrow$ (ii)), or the distance $h(A,B)$ is attained at an intersection of the triangle with the VC boundary ($\Rightarrow$ (iii)). $\square$

Thus, in order to compute the one-sided Hausdorff distance $h(A, B)$, it is sufficient to check only the point sets defined in Cases 1 and 2 since the union of these two sets always contains a candidate point at which the sought after Hausdorff distance is attained. Note that there may exist some (other) $a'_0 \in A$ and $b'_0 \in B$ such that $h(A, B) = d(a_0, b_0) = d(a'_0, b'_0)$ and the point a'_0 belongs to none of the sets described by Cases 1 and 2, see Example 1.

3. Algorithm

In this section, we present an algorithm for computing the precise Hausdorff distance between two triangular meshes. Moreover, in order to reduce the computational cost, we also introduce some local optimizations.

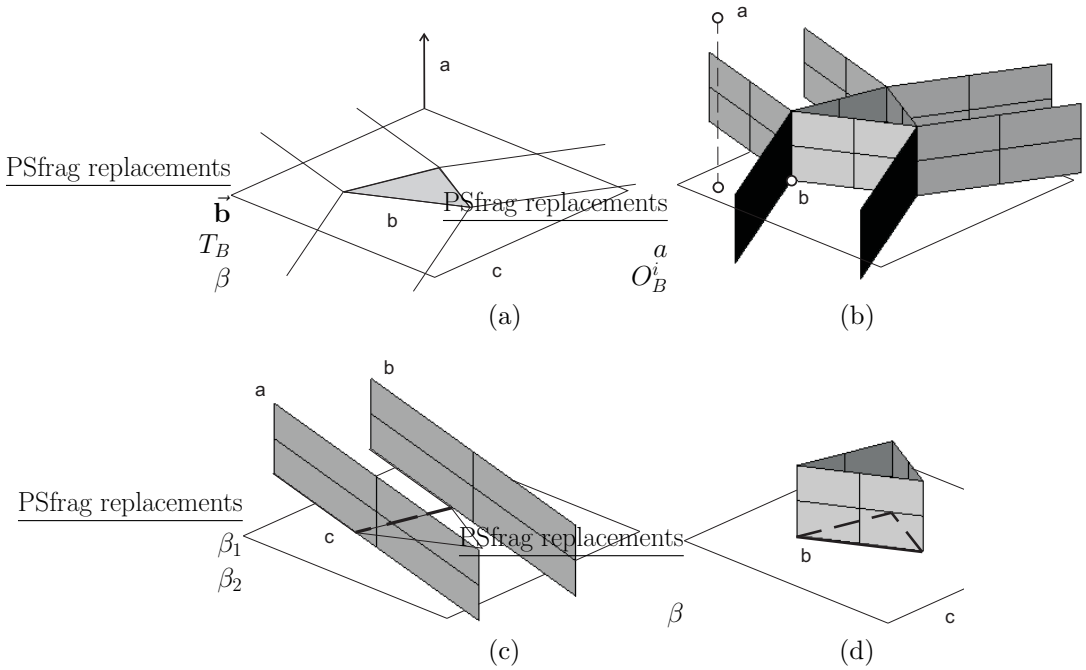

Figure 3: The Voronoi diagram for a triangle T_B in plane β (a) is extruded in the normal direction of β, $\vec{\mathbf{b}}$, to get Voronoi cells in 3D. (b) Point a belongs to the cell of vertex O_B^i. Hence, $d(a, T_B) = d(a, O_B^i)$. (c) The bounding panel of the edge is delimited by a pair of planes β_1 and β_2 that are perpendicular to it and pass through its endpoints. (d) The bounding panel of the face is defined by a triplet of planes passing through its edges and perpendicular to β.

Remark 2. *The computation of the Voronoi diagrams for meshes is difficult in practice. Instead of the precise construction of VD, we handle only the bisector surfaces $\mathbf{B}_B^{ij}$, the mesh-bisectors for pairs of objects O_B^i, O_B^j. Because only some*

sub-part of $\mathbf{B}_B^{ij}$ *contributes to the* VD*, the* bounding panels *of objects* O_B^i, O_B^j *are considered:*

Definition 3. *The* bounding panel of edge O_B^i *is a subset of* $\mathbb{R}^3$ *bounded by two planes that pass through the endpoints of* O_B^i *while also being orthogonal to the edge. The* bounding panel of face O_B^j *is a locus of all points* $x \in \mathbb{R}^3$ *such that* $x_\beta \in O_B^j$*, where* x_β *denotes the orthogonal projection of* x *to* β*, the plane of face* O_B^j*.*

The bounding panel of $O_B^i \in B$ determines a part of $\mathbb{R}^3$, where the particular object may become a Hausdorff distance candidate. Apparently, this easy-to-compute bounding region is a superset of the VC of O_B^i, see Fig. 3(c) and (d). Bounding panels are constructed in order to purge those objects of A, where the one-sided Hausdorff distance can not be attained. Note that bounding panels are, as well as VC, open sets and their boundaries are the corresponding planes.

Based on observations discussed in Section 2.2, we formulate an algorithm, which checks the set of candidate locations (Cases 1 and 2) and returns the precise Hausdorff distance. We now describe several elementary steps, *building blocks* of the algorithm, in more detail:

PSfrag replacements

V_A^1
V_A^2
u
v
$\mathbf{B}_B^{ij}$
$C(u,v)$
$C_{ij}^k(t)$

Figure 4: The uv-space of triangle $V_A^1V_A^2V_A^3$ is intersected with mesh-bisector $\mathbf{B}_B^{ij}$ resulting in implicit conic section $C(u,v) = 0$ (red). This curve is parameterized and trimmed by the triangle, yielding curve $C_{ij}^k(t)$ (superscript k corresponds to the numbering of triangles of mesh A).

- The (minimum) distance between a point a and a triangle $T_B \subset B$, is accomplished by projecting a into the plane β, $T_B \subset \beta$, see Fig. 3. The footpoint is evaluated at nine (normalized) implicit linear functions, representing the boundaries of the VC of the triangle, and the object $O_B^i \subset T_B$ such that $d(a, T_B) = d(a, O_B^i)$ is found. Consequently, Algorithm 1 summarizes the point–mesh distance computation.
- The intersection of an implicit quadric $\mathbf{B}_B^{ij} = 0$ (the mesh-bisector of objects O_B^i and O_B^j) in $\mathbb{R}^3$ with the plane of a triangle $T_A = V_A^1V_A^2V_A^3 \subset A$,

$$P(u,v) = V_A^1 + u(V_A^2 - V_A^1) + v(V_A^3 - V_A^1),\ u, v \in \mathbb{R}, \tag{5}$$

Algorithm 1 { Minimal point–mesh distance}

1: **INPUT**: Triangular mesh B consisting of n triangles and point a
2: **for** every triangular face T_B^i of B **do**
3: $d_i = d(a, T_B^i)$, see Fig. 3(b);
4: **end for**
5: $d(a, B) = \min_{i=1,\dots,n} d_i$;
6: **OUTPUT**: $d(a, B)$.

is an implicit conic section $C(u, v) = 0$ in the parametric (u, v)-plane. This curve is parameterized according to [23] and trimmed by the triangle T_A, see Fig. 4. Note that multiple branches may be obtained and each branch can be represented as a quadratic rational plane curve.

- TODO: Iddo, could you please add a paragraph (probably here) on LEs?

- **Curve-mesh (minimal) distance:** Consider a quadratic rational curve $C_{ij}^k(t)$, $t \in [t_{min}, t_{max}]$, a branch of the intersection curve of a mesh-bisector $\mathbf{B}_B^{ij}$ with a triangle $T_A^k \subset A$. To detect a candidate point $C_{ij}^k(t^\star)$ (Case 2) on the curve $C_{ij}^k(t)$, a parameter $t^\star \in [t_{min}, t_{max}]$, which maximizes $d^2(C_{ij}^k(t), B)$, is sought.

  For the computation of $d^2(C_{ij}^k(t), B)$, we consider the *lower envelope* $LE_{ij}^k(t)$ of the set of squared distance functions $d^2(C_{ij}^k(t), O_B^l)$ between (a *trimmed* sub-segment of) the curve C_{ij}^k and all objects $O_B^l \subset B$, whose bounding panels have non-empty intersection with C_{ij}^k, see Fig. 5. Then, the *candidate parameter* $t^\star$ is a parameter value, where

  1. the maximum of $LE_{ij}^k(t)$ is attained:

  $$LE_{ij}^k(t^\star) \geq LE_{ij}^k(t), \quad \forall t \in [t_{min}, t_{max}] \tag{6}$$

  and

  2. $t^\star$ lies in the part of $LE_{ij}^k(t)$ which corresponds to the squared distance function from the curve $C_{ij}^k(t)$ to the objects O_B^i and O_B^j. (That is, the objects of B which are the closest to the candidate point $C_{ij}^k(t^\star)$ must be O_B^i and O_B^j.) Otherwise, the candidate point is not *relevant* since it is closer to other objects O_B^l than to O_B^i and O_B^j, see Fig. 5; consequently the curve $C_{ij}^k(t)$ can be ignored in the remaining computation.

- **Local optimization using the bounding panel of an object:** Consider a mesh–bisector's triangle intersection curve $C_{ij}^k(t)$, $t \in [t_{min}, t_{max}]$, and an object $O_B^l \subset B$. O_B^l may contribute to the lower envelope $LE_{ij}^k(t)$ only if some part of $C_{ij}^k(t)$ lies inside the VC of O_B^l (and the bounding

panel bounds VC, see Def. 3). Hence, consider a restriction of $C_{ij}^k(t)$, if any, to the bounding panel of O_B^l. Note that only this sub-segment of $C_{ij}^k(t)$, $t \in [t_1, t_2] \subset [t_{min}, t_{max}]$ (and consequently the squared distance function over this parametric domain) may contribute and needs to be taken into account when the lower envelope is computed.

Remark 3. *Note that the majority of the objects of B do not contribute to the lower envelope. Only those objects whose VCs contain (a part of) the intersection curve $C_{ij}^k(t)$ may contain footpoints from the curve. The curve* trimming process *described above is introduced in order to reduce the number of objects, and consequently, the number of square distance functions for which the lower envelope is computed.*

Algorithm 2 { One-sided Hausdorff distance between two triangular meshes}

INPUT: Two triangular meshes A and B
$d := 0$
for every vertex a of A **do**
  $candidate(a) = d(a, B)$; (*Algorithm 1, Case 1)*
  if $candidate(a) > d$; **then**
    $d \leftarrow candidate$;
  end if
end for
for every mesh-bisector $\mathbf{B}_B^{ij}$ of B **do**
  for every triangular face T_A^k of A **do**
    $C_{ij}^k(t) = \mathbf{B}_B^{ij} \bigcap T_A^k$; *(Case 2)*
    $candidate(C_{ij}^k) = \max_{t \in [t_{min}, t_{max}]} LE_{ij}^k(t)$;
    if $candidate(C_{ij}^k)$ is relevant && $candidate(C_{ij}^k) > d^2$ **then**
      $d^2 \leftarrow candidate(C_{ij}^k)$;
    end if
  end for
end for
OUTPUT: One-sided Hausdorff distance $h(A, B) = d$.

Algorithm 3 { Hausdorff distance between two triangular meshes}

INPUT: Two triangular meshes A and B
$H(A, B) := \max(h(A, B), h(B, A))$ (*Algorithm 2*)
OUTPUT: Hausdorff distance $H(A, B)$.

Summing up all the previous building blocks, the one-sided Hausdorff distance computation method is formulated in Algorithm 2. Finally, for the sake of formal completeness, the Hausdorff distance computation is given in Algorithm 3.

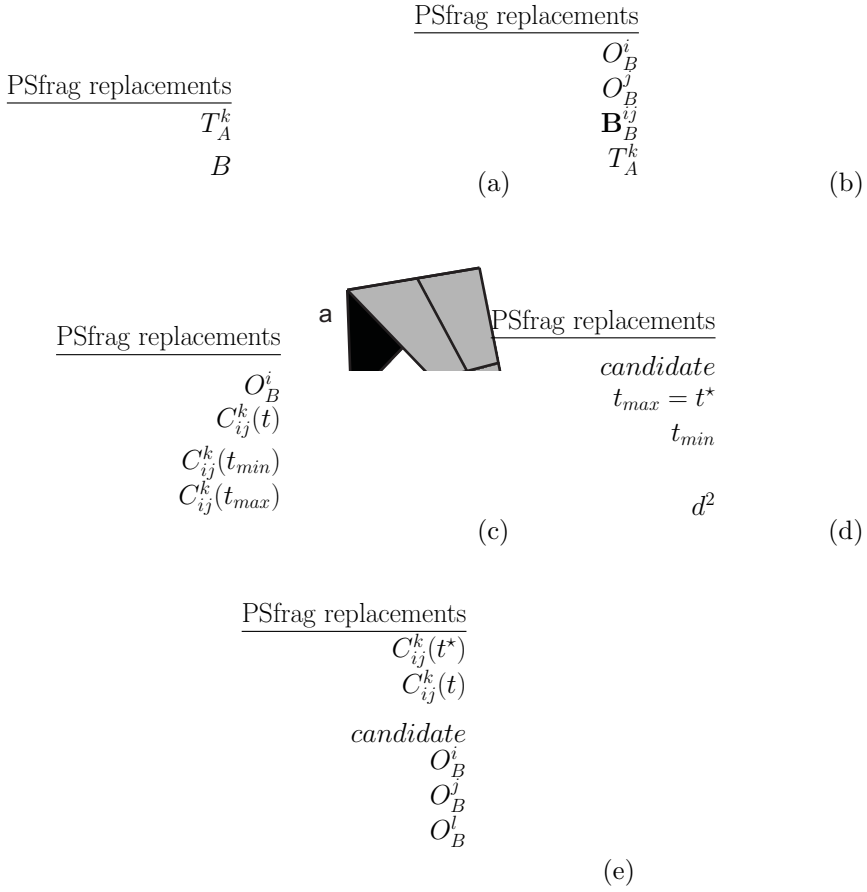

Figure 5: Hausdorff distance candidate point on a bisector-triangle intersection curve: (a) Mesh B and one triangle T_A^k of mesh A. (b) The implicit bisector $\mathbf{B}_B^{ij}$ between face O_B^i and vertex O_B^j of mesh B is intersected with T_A^k resulting in a conic section segment (light grey). (c) Parameter $t^\star$ of conic segment $C_{ij}^k(t)$ which maximizes $d(C_{ij}^k(t^\star), B)$ is sought. (d) Lower envelope $LE_{ij}^k(t)$ (thick black) of a set of quartic rational functions is constructed. These functions measure the squared distances between $C_{ij}^k(t)$ and objects of B, whose bounding panels have non-empty intersection with $C_{ij}^k(t)$. (e) Candidate parametric value $t^\star$ defines candidate point $C_{ij}^k(t^\star)$. Since the closest object of B (edge O_B^l) is neither O_B^i nor O_B^j, candidate $C_{ij}^k(t^\star)$ is irrelevant.

4. Complexity Analysis

In this section, we examine the complexity of Algorithm 2 (and consequently of Algorithm 3).

Let m be the number of objects (vertex, edge, face) of A and let n be the number of objects in B. In line 4 of Algorithm 2, $\mathcal{O}(m)$ vertex-mesh distance computations are needed, since every vertex-mesh distance computation takes $\mathcal{O}(n)$, the total complexity of Case 1 is $\mathcal{O}(mn)$. In lines 9 and 10, every mesh-bisector is intersected with every triangle of A which gives $\mathcal{O}(mn^2)$ intersection curves. Above the parametric domain of each of these curves, the lower envelope of (at most) n squared distance functions is constructed. The lower envelope of a set of n functions are (n, s)-Davenport-Schinzel sequences [24], where s is the maximal number of intersections between any pair of functions (a small constant in our case). The complexity of an (n, s)-Davenport-Schinzel, is denoted by $\lambda_s(n)$ and is an almost linear function[2]. Therefore, the computation of the lower envelope of n squared distance functions over the parametric domain takes $\mathcal{O}(\lambda_s(n) \log n)$ and finding its maximum requires $\mathcal{O}(\lambda_s(n))$ calculations. Thus, the total complexity is

$$\mathcal{C} = \underbrace{\mathcal{O}(mn)}_{\textit{Case 1}} + \underbrace{\mathcal{O}(mn^2)\mathcal{O}(\lambda_s(n) \log n + \lambda_s(n))}_{\textit{Case 2}} = \mathcal{O}(mn^2 \lambda_s(n) \log n). \tag{7}$$

For any practical purposes $\lambda_s(n)$ can be considered $\mathcal{O}(n)$. Therefore, the total complexity of Algorithm 2 can be considered to be $\mathcal{O}(mn^3 \log n)$. For the common case when $m = \mathcal{O}(n)$ we have the complexity $\mathcal{O}(n^4 \log n)$.

5. Examples

In this section we present several examples of computing the Hausdorff distance between two polygonal meshes. All examples were created using the GuIrit GUI interface of the Irit solid modeling environment [3], where the presented algorithm is implemented as a shared library.

All presented examples have been tested on a PC with an Intel(R) Pentium(R) CPU (2.8GHz), 1GB of RAM. Table 1 shows the parameters and timings for meshes shown in Figs. 6–11.

Examples of Figs. 7 and 8 show two relatively simple configurations, where the one-sided distance $h(A, B)$ is attained in the interior of a triangle of A. This event can occur, for instance, when the entire bisector-triangle intersection curve is fully contained in the interior of a triangle (Fig. 8) as well as when

[2] $\lambda_s(n)$ depends on $\alpha(n)$ — the inverse Ackerman function, which is an extremely slow-growing function that appears frequently in algorithms. If n is less than or equal to an exponential tower of 65536 2's, then $\alpha(n) \leq 4$ (see [25], for example)

[3] www.cs.technion.ac.il/∼irit

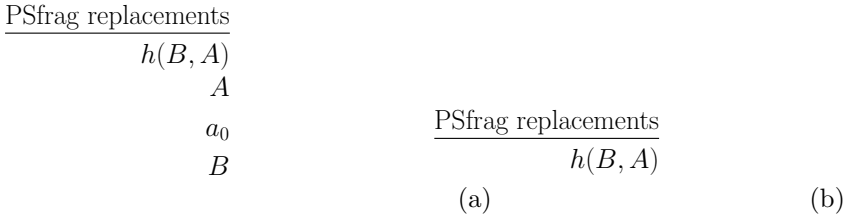

Figure 6: (a) Whereas $h(B, A)$ is attained at the top vertex of B (blue) with a unique footpoint on A (Case 1), the maximal candidate a_0 for $h(A, B)$ is located on the edge of one of the green triangles and therefore lies along a bisector-triangle intersection curve (Case 2); (b) the same example is shown from a different viewpoint.

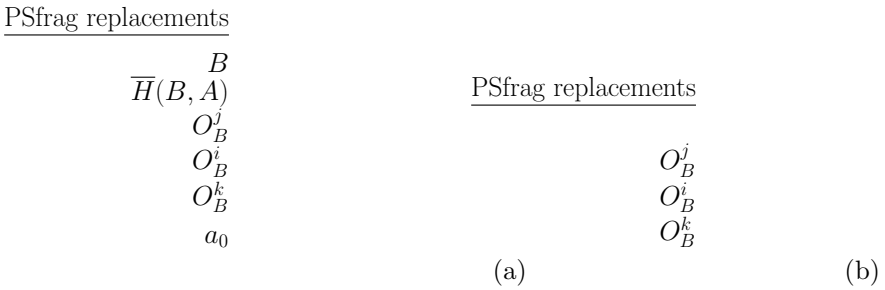

Figure 7: Trisector case: Mesh B consists of three disconnected triangles, A is a single green triangle. The bisector-triangle intersection curve (red) is at equal distance from two vertices O_B^i and O_B^j. The maximal candidate a_0 is located in the interior of the triangle A with the equal distance constraint: $d(a_0, O_B^i) = d(a_0, O_B^j) = d(a_0, O_B^k)$. The top view of the same situation is shown in (b).

the extremal distance value corresponds to the footpoints (at equal distance) to three different objects of mesh B (Fig. 7).

In general, as observed in Table 1, the candidate footpoints occur more frequently at the boundary of the bisector-triangle intersection curve, see Figs. 6

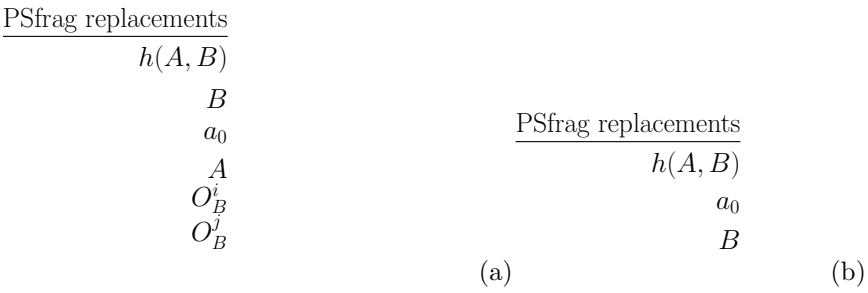

Figure 8: The bisector-triangle intersection curve (red) is at equal distance from vertex O_B^i and face O_B^j. The maximal candidate a_0 is "purely" determined by the distance function on the bisector curve; in contrast to Fig. 7, there is no other object $O_B^k \in B$ such that $d(a_0, O_B^i) = d(a_0, O_B^j) = d(a_0, O_B^k)$. A side view of the same situation is shown in (b).

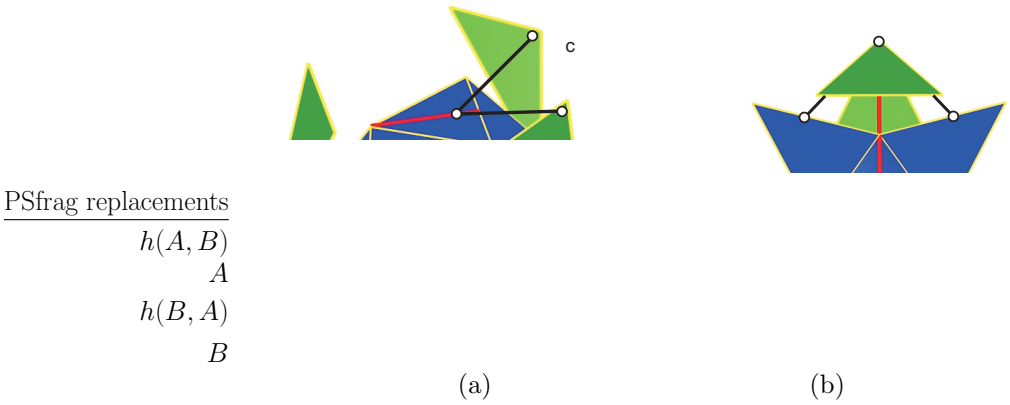

Figure 9: Both one-sided Hausdorff distances are attained along the bisector-triangle intersection curves (red). Each line is the intersection of a triangle with the edge-edge ($h(A,B)$) or the face-face ($h(B,A)$) bisectors. The top view of the same example is shown in (b).

and 10.

Table 1 also reports the number of relevant and irrelevant candidates arisen

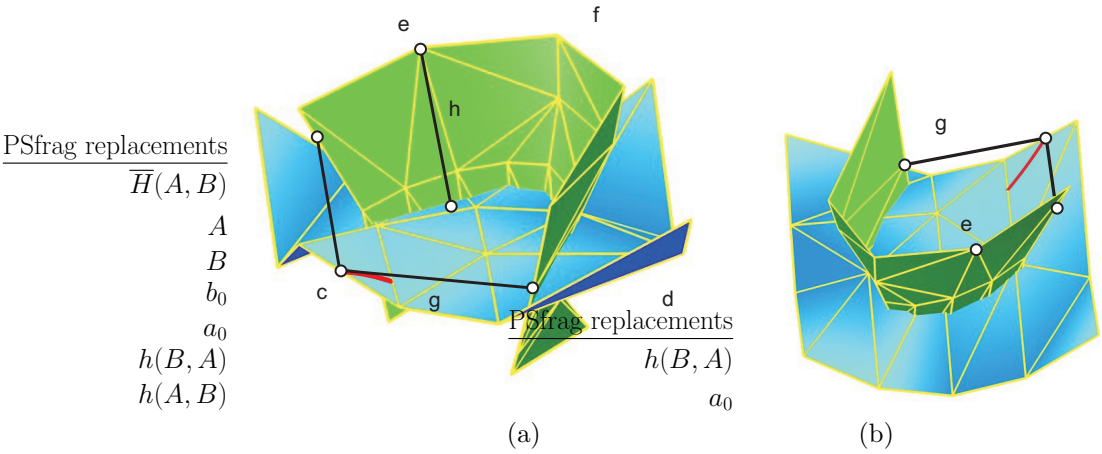

Figure 10: The one-sided Hausdorff distance $h(B, A)$ is attained at the boundary point b_0 of the bisector-triangle intersection curve (red), whereas $h(A, B)$ occurs at a vertex a_0.

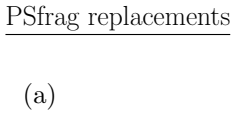

Figure 11: Time demanding example, where both one-sided Hausdorff distances are attained at vertices.

from the bisector-mesh intersection curves. These experimental data indicate the dominance of irrelevant candidates in the majority of practical situations. The irrelevancy of a candidate means that the corresponding bisector-mesh intersection curve thus constructed eventually turned out to be redundant after some later distance comparison with other objects of B. Note that in the current implementation all bisectors have been considered. Moreover, the computation of the candidate points (for the Hausdorff distance) on the bisector-mesh intersection curves is the most time-consuming part of the whole algorithm. Hence, an optimization procedure such as Tang et al. [15] for efficiently culling redun-

Table 1: T_A, T_B are the numbers of triangles, $\mathbf{B}_A$ vs. T_A reports the number of bisector-triangle intersection curves and consequently the number of candidates on these bisector curves. The candidates are further classified as relevant and irrelevant based on the discussion in Section 3. Analogously for $h(A, B)$. Moreover, for the bisector-triangle curves, we report the numbers of cases where the maximum of lower envelope appear at the boundary of the curve domain or in the interior.

	T_A, T_B	$\mathbf{B}_A$ vs. T_B		$\mathbf{B}_B$ vs. T_A		Max of LE		Time
		Rel.	*Irrel.*	*Rel.*	*Irrel.*	*Bndr.*	*Inter.*	*m:sec*
Fig. 7	(1,3)	0	3	13	65	35	46	0:01
Fig. 8	(1,3)	0	11	1	67	65	14	0:01
Fig. 9	(4,4)	10	190	13	411	313	311	0:04
Fig. 6	(16,17)	53	6,544	173	6,541	12,632	1,398	2:52
Fig. 10	(18,34)	305	32,228	223	10,801	39,770	3,787	13:45
Fig. 11	(67,49)	1,029	119,995	1,024	103,164	201,815	22,397	96:10

dant triangles at an early stage of the computation seems to be highly promising in reducing the number of objects whose bisectors may contain relevant candidates and consequently to significantly reduce the computation time of the entire process.

6. Conclusion

We have presented an algorithm for computing the precise Hausdorff distance between two general polyhedra represented as triangular meshes. As observed in Sections 4 and 5, due to the high complexity, the computational timing exceeds reasonable range even for meshes consisting of several tens of triangles (hundred objects). Note that the current implementation computes the mesh-bisectors for all pairs of objects, which involves highly redundant computations. In order to improve the asymptotic timing behavior of the method, some heuristic which purges away redundant objects is needed. Hence, as a future research, our algorithm could be extended to use an efficient culling technique in a preprocessing stage, such as the algorithm of Tang et al. [15], which is expected to significantly reduce the number of candidate objects and hopefully get closer to a real-time result.

Acknowledgment

This work was supported in part by the Israeli Ministry of Science Grant No. 3-4642, in part by the Israel Science Foundation (grant No. 346/07), and also in part by KICOS through the Korean-Israeli Binational Research Grant (K20717000006) provided by MEST in 2007.

References

[1] E. Gilbert, D. Johnson and S. Keerthi, A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Trans. Robot. Automat.*, vol. 4, pp. 193–203, Apr. 1988.

[2] M. Lin and J. Canny, A fast algorithm for incremental distance calculation. *IEEE Int. Conf. Robot. Automat.*, Sacramento, CA, pp. 1008–1014, Apr. 1991.

[3] D. Johnson, Minimum distance queries for haptic rendering. PhD thesis, Computer Science Department, University of Utah, 2005.

[4] C. Lennerz and E. Schomer, Efficient distance computation for quadric curves and surfaces, Proc. of geometric modeling and processing, pp. 60-69, 2002

[5] K.J. Kim, Minimum distance between a canal surface and a simple surface", *Computer-Aided Design*, Vol.35, No.10, pp. 871-879, 2003

[6] X.D. Chen, J.H. Yong, G.Q. Zheng, J.C. Paul and J.G. Sun, Computing minimum distance between two implicit algebraic surfaces, *Computer-Aided Design*, Vol.38, No.10, pp. 1053-1061, 2006

[7] X.D. Chen, L. Chen, Y. Wang, G. Xu and J.H. Yong, Computing the minimum distance between Bézier curves, *Journal of Computational and Applied Mathematics*, Vol.230, No.1, pp. 294-310, 2009

[8] M. Rabl and B. Jüttler, Fast distance computation using quadratically supported surfaces, *Proc. of Computational Kinematics (CK 2009)*, pp. 141-148, 2009

[9] H. Alt and L. Guibas, Discrete geometric shapes: Matching, interpolation, and approximation. *Handbook of Computational Geometry*, J.-R. Sack and J. Urrutia, Eds., Elsevier Science B.V. North-Holland, Amsterdam, 2000.

[10] H. Alt and L. Scharf, Computing the Hausdorff distance between sets of curves. *Procs of the 20th European Workshop on Computational Geometry (EWCG)*, Seville, Spain, pp. 233–236, 2004.

[11] H. Alt and L. Scharf, Computing the Hausdorff distance between curved objects. *Int. J. Comput. Geometry Appl.*, Vol.18, No.4, pp. 307-320, 2008.

[12] M. Atallah, A linear time algorithm for the Hausdorff distance between convex polygons. *Information Processing Letters*, vol. 17, pp. 207–209, 1983.

[13] B. Llanas, Efficient computation of the Hausdorff distance between polytopes by exterior random covering. *Computational Optimization and Applications*, vol. 30, pp. 161–194, 2005

[14] P. Cignoni, C. Rocchini, and R, Scopigno, Metro: Measuring error on simplipied surfaces. *Computer Graphics Forum*, vol. 17, no. 2, pp. 167–174, 1998.

[15] M. Tang, M. Lee and Y.J. Kim, Interactive Hausdorff distance computation for general polygonal models. *ACM Transactions on Graphics (SIGGRAPH '09)*, vol. 28. no. 3, 2009.

[16] M. Eck, T. DeRose, T. Duchamp, H. Hoppey, M. Lounsbery and W. Stuetzle, Multiresolution analysis of arbitrary meshes. *ACM SIGGRAPH '95*, Los Angeles, CA, USA, pp. 173–182, 1995.

[17] P. Alliez and C. Gotsman, Isotropic remeshing of surfaces: a local parametrization approach. *Proc. Int. Meshing Roundtable*, pp. 215-224, 2003.

[18] D. Manocha and C. Erikson, GAPS: General and automatic polygonal simplification. *In Proc. of ACM Symposium on Interactive 3D Graphics*, 79–88, 1998.

[19] B. Jüttler, Bounding the Hausdorff distance of implicitly defined and/or parametric curves. *Mathematical methods in CAGD*, Oslo 2000, pp. 1–10, 2000.

[20] G. Elber and T. Grandine, Hausdorff and minimal distances between parametric freeforms in R^2 and R^3. *Advances in Geometric Modeling and Processing*, F Chen and B Jüttler (Eds.), Procs of the 5th Int'l Conf, GMP 2008, Hangzhou, China, April 23-25, 2008. Lecture Notes in Computer Science 4975, Springer, pp. 191–204.

[21] H. Alt, P. Brass, M. Godau, C. Knauer and C. Wenk, Computing the Hausdorff distance of geometric patterns and shapes. *Discrete and Computational Geometry. The Goodman–Pollack Festschrift*, B. Aronov, S. Basu, J. Pach and M. Sharir, Eds., *Algorithms and Combinatorics*, Vol. 25, pp. 65–76, Springer-Verlag, Berlin, 2003.

[22] A. Okabe, B. Boots, K. Sugihara and S.N. Chiu, *Spatial tessellations: Concepts and applications of Voronoi diagrams.* Wiley publ., NYC, 2^{nd} edition, 2000.

[23] J. Hoschek, Bézier curves and surface patches on quadrics. *Mathematical methods in Computer aided Geometric Design II*, T. Lyche and L. Schumaker (eds.), pp 331-342, 1992.

[24] M. Sharir and P.K. Agarwal, *Davenport-Schinzel sequences and their geometric applications.* Cambridge University Press, New York, NY, USA, 1996.

[25] E.W. Weisstein, "Ackermann Function." From MathWorld—A Wolfram Web Resource. http://mathworld.wolfram.com/AckermannFunction.html.

[26] M. Guthe, P. Borodin and R. Klein, Fast and accurate Hausdorff distance calculation between meshes. *Journal of WSCG*, V. Skala (ed.), Vol. 13, pp 41-48, 2005.

[27] G. Elber, M. S. Kim, The bisector surface of freeform rational space curves. *Proceedings of the thirteenth annual symposium on computational geometry*, Nice, pp 473-474, 1997.