Robust Construction of the Three-Dimensional Flow Complex

Frédéric Cazals, Aditya Parameswaran, Sylvain Pion

Robust Construction of the Three-Dimensional Flow Complex

F. Cazals
INRIA Sophia-Antipolis, France
Frederic.Cazals@inria.fr

A. Parameswaran
Stanford University, USA
adityagp@cse.iitb.ac.in

S. Pion
INRIA Sophia-Antipolis, France
Sylvain.Pion@inria.fr

ABSTRACT

The Delaunay triangulation and its dual the Voronoi diagram are ubiquitous geometric complexes. From a topological standpoint, the connection has recently been made between these cell complexes and the Morse theory of distance functions. In particular, in the generic setting, algorithms have been proposed to compute the flow complex —the stable and unstable manifolds associated to the critical points of the distance function to a point set. As algorithms ignoring degenerate cases and numerical issues are bound to fail on general inputs, this paper develops the first complete and robust algorithm to compute the flow complex.

First, we present complete algorithms for the flow operator, unraveling a delicate interplay between the degenerate cases of Delaunay and those which are flow specific. Second, we sketch how the flow operator unifies the construction of stable and unstable manifolds. Third, we discuss numerical issues related to predicates on cascaded constructions. Finally, we report experimental results with CGAL's filtered kernel, showing that the construction of the flow complex incurs a small overhead w.r.t. the Delaunay triangulation when moderate cascading occurs. These observations provide important insights on the relevance of the flow complex for (surface) reconstruction and medial axis approximation, and should foster flow complex based algorithms.

In a broader perspective and to the best of our knowledge, this paper is the first one reporting on the effective implementation of a geometric algorithm featuring cascading.

Categories and Subject Descriptors: G.m [Mathematics of Computing]: Misc

General Terms: Algorithms, Reliability.

Keywords: Euclidean distance function, Morse-Smale complex; lazy constructions, cascading; surface reconstruction, medial axis approximation.

SCG'08, June 9–11, 2008, College Park, Maryland, USA.
.

1. INTRODUCTION

1.1 Morse Theory of Distance Functions

Given a collection of geometric objects, the loci of point equidistant from these objects is a fundamental construction appearing under various names in differential geometry (cut locus, medial axis) [1, 3], mathematical morphology (medial axis, skeleton) [27], non-smooth analysis (singular set, central set) [22], Computational Geometry (Voronoi diagram), etc. The construction is theoretically fundamental, and has countless practical applications [25].

An equally important construction is the collection of level sets of the distance function to these objects. Investigating the evolution of these level sets when the distance increases can be casted into the framework of Morse theory, which is concerned with the study of functions defined on manifolds [24]. In the setting of Computational Geometry (CG) and Geometric Modeling, the framework of Morse theory has been approached from several directions. The development of α-shapes [11] and the investigation of topological properties of collections of (growing) balls [12] was probably the first time notions from differential topology were used in CG —these constructions are essentially concerned with topological events undergone by the level sets of the (power) distance function. In a nearby vein, the precise framework of Morse theory has been applied to distance functions originating in the context of Euclidean Voronoi diagrams for points [28], and a construction termed the *flow complex* was developed in [18] based on properties of the Delaunay and Voronoi diagrams. These veins connected recently since its was shown α-shapes and flow shapes are homotopy equivalent [8, 4]. In a somewhat different realm, the distance function to a compact set has been used in mathematical morphology to investigate properties of medial axes and skeletons [27]. Further properties of this function were used in [23] to prove that any open bounded subset of $\mathbb{R}^n$ has the same homotopy type as its medial axis. From a global perspective, constructions related to distance functions bridge the gap between local geometric and global topological properties.

Application-wise, distance functions proved recently to convey important information for the study of Van der Waals models [14], in surface reconstruction [17, 7, 13, 10], and shape segmentation [9]. Apart from this Euclidean setting, Morse theoretic ideas related to Morse-Smale diagrams have been explored [5] in the realm of Forman's combinatorial Morse theory [16].

1.2 Morse Theory and the Flow Complex

Morse Theory. Following classical terminology in differential topology, a *critical* point of a differentiable function is a point where the differential of the function vanishes, and the function is called a *Morse* function if its critical points are isolated and non-degenerate. Given a Morse function defined over a manifold M, and a critical point p of that function, the stable (unstable) manifold $W^s(p)$ ($W^u(p)$) is the union of all integral curves associated to the gradient of the function, and respectively ending (originating) at p. The function is termed *Morse-Smale* provided its stable and unstable manifolds intersect transversely [26]. For such a function, the *Morse-Smale complex* is the subdivision of M formed by the connected components of the intersections $W^s(p) \cap W^u(q)$, where p and q range over all critical points.

Flow Complex and Applications. In the context of the Euclidean Voronoi diagram of points, these ideas are incarnated by the *flow complex*, which, in the following, refers to the stable and unstable manifolds associated to the distance function to a point set. The construction of stable manifolds of index two saddles in $3D$ has been presented in [8], with a generalization to any dimension in [4]. The construction of unstable manifolds has been presented in [19, 20]. In geometric modeling terms, the flow complex (i.e. selected stable and unstable manifold) has interesting approximation properties. So far, stable manifolds have been used for surface reconstruction [4], while unstable manifolds have been involved with the approximation of the medial axis [19] and the identification of special regions (cylinders and flat regions) in [20]. Aside these works which are concerned with manifolds, an example reconstruction of a self-intersecting surface in terms of stable manifolds is presented on Fig. 1 – from [6].

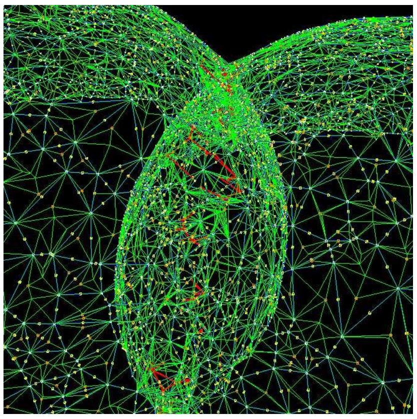
Figure 1: Reconstructing a self-intersecting surface.

1.3 Contributions and Paper Overview

The algorithms just mentioned operate under genericity hypothesis and elude a major difficulty: the presence of *cascaded constructions*. As we shall see, the *orbit* of a point i.e. its trajectory under the flow associated to the gradient of the distance function consists of line-segments. Concatenating such segments naturally yields cascaded constructions —and also predicates on such constructions, as illustrated on Fig. 2.

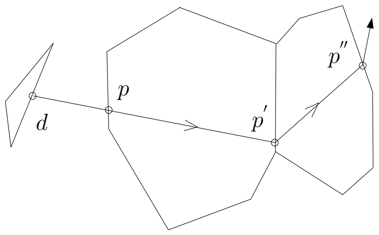

Figure 2: The flow propels p onto p', p' onto p'', etc. Rounding errors may yield inconsistencies, so that exact cascaded constructions are required.

In this context, after having recalled the fundamentals of the flow complex in section 2, we make the following contributions. In section 3, we present the complete road-map to flow across Voronoi faces. This road-map provides the necessary and sufficient operations required to build the flow operator, from which the construction of stable and unstable manifolds can be implemented. Algorithms to flow across a Voronoi facet is sketched in section 4. The question of degenerate cases handling, in conjunction with numerical issues related to cascaded constructions is addressed in section 5. Finally, section 6 presents a detailed experimental study.

1.4 Notations

Central to our constructions are the Voronoi and Delaunay diagrams. We shall assume the Voronoi diagram is represented via its dual Delaunay triangulation. Following standard usage, we assume the Delaunay triangulation is given as a collection of simplices with the proper incidences. For example, all Delaunay triangles incident upon an edge e can be accessed by rotating around this edge; two consecutive such triangles define a tetrahedron incident on e. When manipulating Voronoi/Delaunay faces of all dimensions, we shall use the following Voronoi/Delaunay terminology: Vertex/cell, Edge/facet, Facet/edge and Cell/vertex. In particular, an initial out of {V, E, F, C} or {c, f, e, v} identifies unambiguously a Voronoi or Delaunay face. When the dimension is not specified, a Voronoi/Delaunay face are denoted O/O^*. Moreover, the duality operator returning the Delaunay face associated to a Voronoi face, or vice-versa, is denoted with a super-script, e.g. $V^* = c$. The affine hull of a Delaunay simplex s (its dual s^*) is denoted Aff(s) (Aff(s^*)).

Given a Delaunay simplex of any dimension, the center of the *smallest circumscribing ball* is denoted z, while the center of the *smallest empty* ball (the ball does not contain any other sample point) is denoted y. In particular, recall z always lies on the affine hull of the simplex. If the smallest circumscribing ball is empty, the simplex is called *Gabriel*.

Finally, the inner product of two vectors x, y is denoted $< x, y >$, and the exterior product $x \wedge y$.

2. DISTANCE FUNCTION, CRITICAL POINTS, FLOW COMPLEX

2.1 The Distance Function d_K to a Compact Set

Given a compact set K of $\mathbb{R}^d$, the distance function to K is defined by $d_K(p) = \min_{q \in K} d(p, q)$, with $d(p, q)$ the Euclidean distance. Function d_K is not differentiable on the medial axis of K, but it has been shown that a generalized gradient coinciding with $\nabla d_K(p)$ where d_K is differentiable can be defined [23]. To see how, for any point p, consider the set $C(p)$ of contact points to K, i.e. the points realizing the distance $d_K(p)$. Denote $d(p)$ be the center of the smallest ball containing these contact points. This generalized gradient is defined as [23]:

$$\nabla d_K(p) = \frac{p - d(p)}{d_K(p)}. \tag{1}$$

It can be shown it defines a *flow operator*, i.e. a map $\Phi(p, t)$ defined over $\mathbb{R}^3 \times \mathbb{R}$, describing the trajectories induced by the gradient vector field $\nabla d_K(p)$. The *orbit* of point p is the set $\{\Phi(p, t)\} \mid t \in \mathbb{R}\}$.

When the compact set is a collection of sample points, function d_K is easily studied since the medial axis reduces to the $(d-1)$-skeleton of the Voronoi diagram. In particular, if O stands for the relatively open Voronoi face of smallest dimension containing point p, the set $C(p)$ corresponds to the vertices of the simplex O^*. Because point $d(p)$ actually indicates the direction of this generalized gradient, it has been called the *driver* in [18]. Equivalently, the driver is characterized as the point of O^* nearest to p. For this reason, abusing terminology, we shall speak of the driver of p, as well as the driver of O^* —meaning determined by O^*. As the driver remains the same for a relatively open Voronoi face, the orbit of a point consists of line-segments.

2.2 Critical Points of d_K

In Morse theory, when studying a Morse function on a manifold, its critical points induce a decomposition of the manifold. This decomposition is obtained by tracking the topological changes of the level set of the function on the manifold, and can be phrased in terms of homotopy or homeomorphy (or even diffeomorphy) equivalence [24]. One change occurs at each critical value, by attaching one cell of dimension k corresponding to the inflow of the critical point (for the homotopy equivalence), or the Cartesian product of such a cell with a disk of dimension $d - k$ (for the homeomorphy equivalence, with d the dimension of the manifold). The notion of critical point just recalled also applies to the generalized gradient of Eq. (1). However, if one wishes to track topological changes of the level sets of function d_K, some care is in order. As illustrated on Fig.3, the wavefront does not systematically incur a topological change when a point such that $\nabla d_K(p) = 0$ is reached. Letting *topological change* stand for the homotopy of homeomorphy type change, we define:

DEFINITION. 1. *A point p is called* topologically critical (regular) *if the level set of d_K (does not) incurs a topological change upon crossing point p.*

Following the analysis carried out in [28], we actually have:

OBSERVATION. 1. *Point p is* critical *iff it belongs to the* interior *of the convex hull of its contact points $C(p)$.*

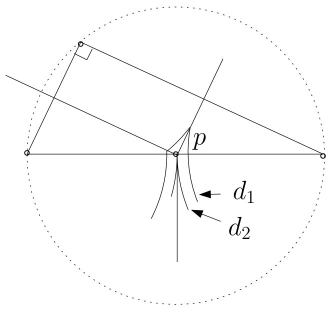

Figure 3: $2d$ example with $\nabla d_K(p) = 0$, yet, point p is topologically regular as the level set does not undergo a topological change when the distance shifts from d_1 to d_2.

Two comments are in order. First, this definition is different from that traditionally used in the context of distance functions to compact sets [23], where a point is termed critical if it belongs to the convex hull of the contact points. Second, in the context of Delaunay - Voronoi, this definition is also different from that using the intersection between the Delaunay and Voronoi faces. In fact, the characterization of critical points from Obs. 1 is more general, since the convex hull of the contact points reduces to a simplex if no degeneracy occurs in Delaunay. Following [28, 18], the *index* of a critical point p is $d - k$, with k the dimension of the open Voronoi face of lowest dimension containing p. Using critical points, we define:

DEFINITION. 2. *The* signature *of a tetrahedron consists of the number of critical points associated to its k-faces for $k = 0, \ldots, 3$, that is $(4, i_1, i_2, i_3)$ with $4 - i_1 + i_2 - i_3 = 1$.*

Definition 1 is used to disqualify selected critical points with $\nabla d_K(p) = 0$, for example an index one colliding with an index two, or an index two colliding with an index three.

2.3 Building the Flow Complex

Building the flow complex means computing the stable and unstable manifolds of the index one and index two critical points. To bridge the gap between the primitive constructions presented in this paper and the flow complex, the following comments are in order [1].

Index 1 critical point. The stable manifold of an index one saddle is the Gabriel edge. For the unstable manifold, the construction is presented in [19]. The key primitive consists of computing the image of a (portion of) Voronoi edge. Again, this is easily done with the algorithm of section 4.

Index 2 critical point. A stable manifold which does not contain a Voronoi vertex is two-dimensional, and an algorithm to compute it is presented in [18]. Under this hypothesis, which we also assume, the primitive operation consists of flowing across a Voronoi facet e^* so as to find the pre-image by the flow of a point located on a Voronoi edge bounding e^*. Running the algorithm of section 4 backward so as to decrease (and not increase) the distance provides this operation. See Fig. 4 for an illustration. The construction of unstable manifold reduces to the computation of the

[1]Due to the lack of space, the reader is referred to the original papers for the algorithms.

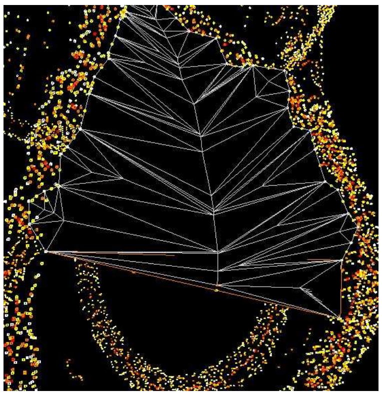

Figure 4: Schale model: critical points (index 0: grey; index 1: yellow; index 2: orange; index 3: red), and stable manifold with 264 triangles.

orbit(s) of the Voronoi vertex(ices) of the tetrahedron(a) incident on the facet defining the critical point [19]. The tricky operation is again the algorithm of section 4, used forward this time. Taking point p as a Voronoi vertex on Fig. 2 illustrates this process.

3. FLOWING ACROSS VORONOI OBJECTS: ROAD-MAP

Describing the flow operator faces several types of difficulties: first, one needs to accommodate flow to infinity across unbounded Voronoi faces and across Voronoi rays dual of Delaunay triangles found on the convex hull; second, one needs to take care of flow specific degenerate cases; third, one needs to handle Delaunay specific degenerate cases.

3.1 Flow Segment

As recalled in section 2.1, the flow is linear in a relatively open Voronoi face. To account for the restriction of an orbit to such a face, we define:

Definition. 3. *A* combinatorial flow segment *consists of three relatively open Voronoi faces: S, the face containing the start point; C, the face crossed; R, the face reached by the flow on the boundary of C, or infinity if the point flows to infinity. Given a point $p \in S$, its* flow segment *consists of its orbit through C and of its endpoint in R.*

Practically, a combinatorial flow segment is denoted by a triple $S - C - R$, each letter taken from the set {V, E, F, C} to represent a Voronoi face. Computing a flow segment thus requires identifying the face crossed and finding the trajectory itself from which the face reached is obtained. This calculation is the building block of the flow operator, as the orbit of a point consists of a sequence of flow segments.

3.2 Drivers

To discuss the possible drivers, we perform a case analysis as a function of the Voronoi object containing the start-point:
—Starting from the interior of a Voronoi cell. The driver is the sample point associated to the Voronoi cell.
—Starting from the relative interior of a Voronoi facet. The driver is the midpoint of the Delaunay edge dual of the facet.
—Starting from the relative interior of a Voronoi edge. The driver is associated to the dual Delaunay triangle or to one of its edges.
—Starting from a Voronoi vertex. If the Voronoi vertex is critical, it is its own driver. If not, the driver is associated to an edge or a triangle of the Delaunay tetrahedron dual of the Voronoi vertex, see Figs. 7 and 8. A Voronoi vertex driven by a triangle flows along a Voronoi edge. But as illustrated on Fig. 6, the converse is not strictly true, due to a degenerate geometry of the tetrahedron dual of the Voronoi vertex. Consider a Voronoi center c of a non-critical tetrahedron driven by the midpoint d of edge p_0p_1. Let S be the sphere circumscribing the tetrahedron, S_{01} be the sphere whose diameter is p_0p_1, and B_{01} the ball bounded by S_{01}. Under our hypothesis, the ball B_{01} contains p_2 and p_3. The intersection $S \cap S_{01}$ is a circle C_{01}, and since p_2 and p_3 belong to S and are located inside B_{01}, they are located on the lower spherical cap of S bounded by C_{01}. If p_2 and p_3 belong to the interior of this spherical cap, we are in the generic situation of Fig. 8 –the two triangles $p_0p_1p_2$ and $p_0p_1p_3$ are obtuse at p_2 and p_3. If p_i ($= p_2$ or p_3) belongs to the circle C_{01}, then triangle $p_0p_1p_i$ is a right angle at p_i, and the Voronoi segment $(p_0p_1p_i)^*$ is collinear with the line-segment dc.

dim(S) | dim(C)	0	1	2	3
0	V-V-V [x]	V-E-{V,∞} [x]	V-F-{V,E,∞} [x]	∅
1	∅	E-E-{V,∞} [x]	E-F-{V,E,∞} [x]	∅
2	∅	∅	F-F-{V,E,∞} [o]	∅
3	∅	∅	∅	C-C-{V,E,F,∞} [o]

Table 1: Possible flow segments, with {...} the possibilities given a start S and crossed C faces. Entries marked as [o] are relevant for the flow operator but not for the calculation of the flow complex, which relies upon entries marked [x].

3.3 Enumerating all Possible Flow Segments

Before enumerating all possible flow segments, some care is in order so as to accommodate $n > 4$ co-spherical points in the Delaunay triangulation. We define:

Definition. 4. *A point p is called a* duplicate *Voronoi vertex if it coincides with at least two circumcenters of tetrahedra featuring co-spherical points. A Voronoi vertex which is not duplicate is called* simple*. A Voronoi edge corresponding to a duplicate Voronoi vertex is called* trivial.

The possible flow segments are listed in Table 1, where S and C respectively refer to the Start and Crossed Voronoi faces. To see how this table is built, consider a start Voronoi face S. One flow segment type is obtained for each simplex type σ which is a face of S^*, and whose circumscribing ball contains the vertices of $S^* \backslash \sigma$ in its interior or boundary. Enumerating all such configurations yields the possible flow segments. For a given flow segment, by construction, the driver of the start Voronoi face S and the driver of the crossed Voronoi face C are thus identical.

Before proceeding, one can observe that there are actually three types of degeneracies:
—Generic Voronoi face, degenerate flow for selected starting points. This is the E-F-V case depicted on Fig. 5.
—Non generic Voronoi face, non-generic flow. This occurs when flowing from a Voronoi vertex, as on Fig. 6.
—Duplicate Voronoi vertices. This difficulty is independent from the previous two. As discussed in section 2, a duplicate Voronoi vertex may be topologically regular. Such a vertex may therefore yield a V-V-V flow segment.

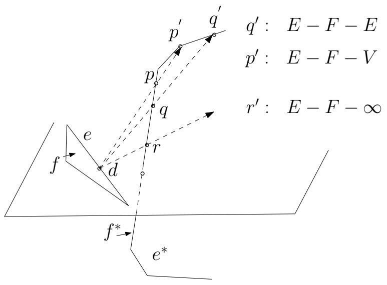

Figure 5: Flow segments E-F-{V,E,∞}: Flow from a Voronoi edge f^* dual of a Gabriel non-critical Delaunay triangle f. Driver d is associated to edge e.

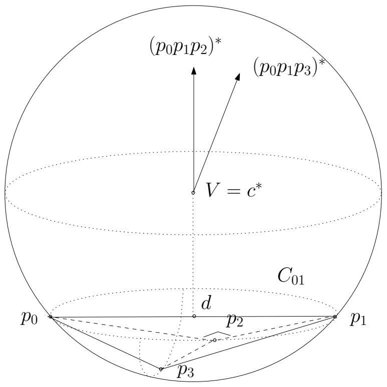

Figure 6: Point p_2 belongs to the minimum-enclosing ball defined by edge p_0p_1: Voronoi edge $(p_0p_1p_2)^*$ is collinear with segment dc.

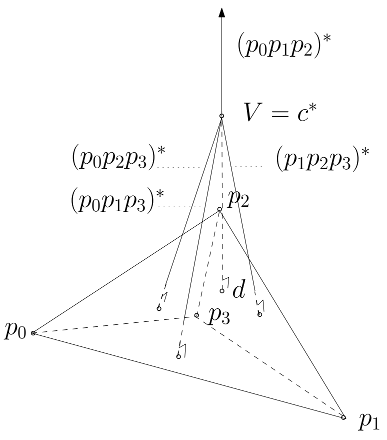

Figure 7: Flow segments V-E-{V,∞}. Signature of the tetrahedron shown is $(4,6,3,0)$.

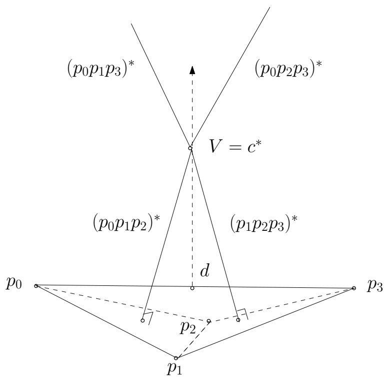

Figure 8: Flow segments V-F-{V,E,∞}. Signature of the tetrahedron shown is $(4,5,2,0)$.

4. FLOWING ACROSS A VORONOI FACET

4.1 Overview

Consider a point p on the boundary of a Voronoi face e^* –Fig. 9. We assume point p is specified from the smallest dimensional Voronoi object containing it. In particular, if point p coincides with a duplicate Voronoi vertex, we assume it is specified from a Voronoi vertex and not from a trivial Voronoi edge. Given such a point, we wish to find the open Voronoi face crossed —the face itself or one of its boundary edge, together with the point reached by the flow. The algorithm consists of examining the Voronoi edges and/or rays bounding e^*, so as to report the Voronoi edge or Voronoi vertex reached.

More precisely, let p be a point on the boundary of a Voronoi facet $F=e^*$: p is either a Voronoi vertex, or lies in the interior of a Voronoi edge. Let dp^+ be the ray emanating from d and passing through p. Assuming the facet e^* is not Gabriel and since the flow is linear in Voronoi objects, we need to compute the intersection between the ray dp^+ and the boundary of e^*. To see how to proceed, let $e=(p_0p_1)$ the non-Gabriel Delaunay edge. To compute the aforementioned intersections, we examine the intersection between the ray dp^+ and the Voronoi edges or rays bounding the facet e^*. The following numerical operations are involved.

4.2 Predicates and Construction

Predicates. Consider a Voronoi segment c_ic_{i+1} along the boundary of e^*. To check whether it is intersected by dp^+, since edge e and its dual are orthogonal, it is sufficient to compute the following signs:

$$s_0 = \text{sign}(< dp \wedge dc_i, p_0p_1 >), \tag{2}$$
$$s_1 = \text{sign}(< dp \wedge dc_{i+1}, p_0p_1 >). \tag{3}$$

The predicate for a Voronoi ray is similar. Calling c_i the finite Voronoi vertex and u the unit vector orienting the ray, we compute:

$$s_0 = \text{sign}(< dp \wedge dc_i, p_0p_1 >), \tag{4}$$
$$s_1 = \text{sign}(< dp \wedge u, p_0p_1 >). \tag{5}$$

Definition. 5. *The signature of a Voronoi edge or ray dual of a Delaunay facet f is defined by $\widetilde{f}=[s_0,s_1]$, with s_0,s_1 defined by Eqs. (2) or (4). The signature of a Voronoi face is defined as the concatenation of the signatures of its Voronoi edges/rays.*

The connexion between the flow segment and the signs is as follows:

OBSERVATION. 2. *Let s_i be the sign associated with Voronoi center c_i. One has: $s_i = 0$ iff $p = c_i$, or $p \neq c_i$ and dp is collinear to dc_i.*

Let s_i be the sign associated with the direction u of a Voronoi ray. One has $s_i = 0$ iff dp and u are collinear, and in that case, the sign of the finite Voronoi vertex may be 0 or ± 1.

Construction. The predicates just presented allow the detection of the Voronoi edges intersected at the expense of the evaluation of the sign of a polynomial. Assume we have found a Voronoi edge or ray intersected by the ray dp^+. The construction of p' is easily done by considering the intersection between the Voronoi line-segment or ray with the plane defined by the triple p_1p_2p. This intersection requires manipulating rational numbers.

Notice the construction of point p' needs to be done exactly: on Fig. 9, if point p' lies next to a Voronoi vertex, the rounding errors inherent to floating point computations may yield a constructed point located on the wrong Voronoi edge, or outside the Voronoi edge. In that case, ensuing computation may be erroneous.

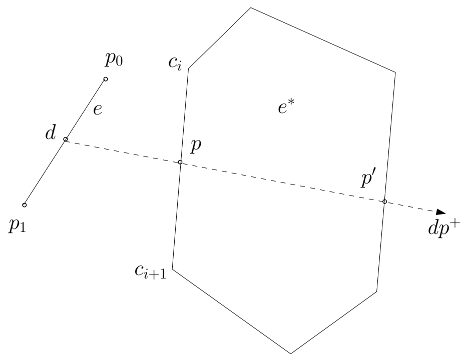

Figure 9: Flowing across a Voronoi face. The image of point p under the flow is point p'.

4.3 General Case

Following Table 1, the flow across a Voronoi facet either reaches infinity, the interior of a Voronoi edge/ray, or a Voronoi vertex. Moreover, one needs to handle the following degeneracies:

- the start point is driven by the middle of a Delaunay edge, but instead of flowing in the relative interior of the dual Voronoi facet, it flows along a Voronoi edge/ray. This situation, called *boundary degenerate flow* or bdf for short, is illustrated on Fig. 6;
- the Voronoi facet features duplicate Voronoi vertices.

To handle these cases and find point p' –if any, an elementary strategy consists of running a linear scan around the Voronoi facet, so as to consecutively probe the Voronoi edges/rays. But since the facet is convex and the flow linear, a binary search can be resorted to. Due to the lack of space, the proof of the following claim is postponed to the long version of the paper:

PROPOSITION. 1. *Consider a Voronoi facet e^* bounded by n Voronoi edges, with m degenerate Voronoi vertices. Resorting to a binary search, algorithm* `Flow_across_vor_facet` *computes the output in time $O(m + \log n)$.*

5. ROBUSTNESS AND ARITHMETIC ISSUES

5.1 About Degenerate cases and Symbolic Perturbations

Symbolic perturbations (sp) are known to alleviate the handling of degenerate cases. We opted for not using them here, as they did not prove useful enough, and this, for the following two reasons. First, symbolic perturbations, which consist of adopting a *convention* mimicking the general case although a degenerate one is faced, still require detecting degeneracies. In our setting, degeneracies are easily handled once detected . Second and more fundamentally, sp are about the combinatorics, not the geometry. That is, a sp allows one to mimic the generic setting, but the geometric embedding of the objects is not affected, and the result is the same: in our case, the degeneracies come from duplicate Voronoi vertices and boundary degenerate flow, and the flow is bound to the geometry of these Voronoi cells.

5.2 Implementation and Numerical Issues

Implementation. In standard terminology, primitives on basic geometric objects are separated between predicates, which return a boolean or enumerated result, and constructions, which produce new geometric objects or numbers. In the Computational Geometry Algorithms Library `www.cgal.org`, these objects and primitives are gathered within so called kernels. Our implementation follows this spirit, and one of its main focuses is on efficiently supporting the exact computations. As all computations are rational (in particular all constructions have rational coordinates when the input is rational as well), we do not need to resort to algebraic tools. But the main difficulty lies in the cascading of geometric constructions. To see how these are handled, we first review some fundamentals on filtering.

Filtering at the number type level. Robust geometric code requires exact predicates. As floating point operations provide the right answer in most cases, it is enough to provide an exact evaluation based on costly multi-precision arithmetic (typically that provided by the GMP library [21]) when the calculations in double fail. Filtered predicates resorting to interval arithmetic follow this line. Such predicates are based on a new number type which forwards its operations to an interval arithmetic type, and also remembers the history of operations performed by constructing a Directed Acyclic Graph (DAG). When a predicate evaluation fails using the interval type, the DAG is used to recompute the values with an exact multi-precision type [2].

Filtering at the geometric level. Filtering at the number type level requires one DAG node and one rounding mode change for each arithmetic operation, and thus incurs a significant overhead in the context of cascaded constructions. Moreover, cascading operations has the following expected effects on these tools: the width of the intervals grows so that their ability to catch cases in the computations deteriorates; the size of the rational numbers grows so that exact calculations are slowed down. These difficulties can be alleviated by transposing the filtering from the number type to the geometric level. This lazy evaluation scheme at the geometric level is implemented in CGAL's lazy kernel [15]. For constructions, the lazy kernel stores (i) pointers to the objects used for the construction (ii) an approximation of the construction, and (iii) a pointer to the exact value, the latter being computed iff a predicate on the approximation fails. Our implementation is templated by this geometric kernel [15].

6. EXPERIMENTS

This section presents results on index two stable and unstable manifolds (S2 and U2). Due to the lack of space, a report on index one unstable manifolds is skipped.

Hardware and Software used. We have used CGAL 3.3 for the experiments, together with GMP 4.2 and the GNU compiler g++ 4.1. The machine used for the experiments was a PC running Linux Fedora Core 5, with 2MB of memory and a 3GHz Pentium 4 processor.

Models. Experiments were conducted on scans of physical models, with sizes in-between 2.5k and 540k points. The following notations are used in Table 2: idx_k: number of critical points of index k; χ: Euler characteristic; t_D (t_{FC}) time (in seconds) to construct the Delaunay triangulation (selected (un)stable manifolds); M: the memory size in megabytes.

Kernel used. As discussed in section 5.2, several kernels can be used to build the flow complex. For constructing the flow complex, we found that the memory footprint required by `CGAL::Exact_predicates_exact_constructions_kernel` has been reduced by a factor between 3 and 10 between CGAL 3.2 and CGAL 3.3, as it now uses of `Lazy_kernel` internally. Thus, all experiments were conducted with this kernel.

Degeneracies and locality. To begin with, it should be mentioned *all models* tested faced degenerate cases discussed in sections 2.2 —degenerate critical points, and section 4 —duplicate Voronoi vertices. This owes to the fact that scanned data sets are (often) rounded on an integer grid. As anticipated, constructing the flow complex is a non local process due to orbits extending arbitrarily far. This is illustrated on Fig. 4, which shows the critical points of all indices and the largest index two stable manifold on the vase model. The bottom most line-segment of this manifold, in orange, is a boundary Gabriel edge. The stable manifold extends above in between the two branches of the vase, and features 264 triangles.

On S2 and U2. To get insights on S2, recall that such a manifold is bounded by Gabriel edges. An interesting information is thus the number n_{S2} of flow segments required to reach an index one critical point from an index two critical point. Notice this number matches the number of cascaded constructions along the process. By convention, a Gabriel edge of the critical triangle requires one segment.

Similarly for a U2, an obvious parameter of interest in the number n_{U2} of flow segments from the critical point to a maximum or to infinity. However, this number is an upper bound on the number of cascaded constructions, as in flowing from a Voronoi vertex, a cascade is broken whenever a flow segment along a Voronoi edge occurs —a constructed point is replaced by a Voronoi vertex. In addition to n_{U2}, we therefore investigate the distribution of the number n_{WCC} of consecutive cascaded constructions.

Consider Table 2. For a given model, the tag $S2$ ($U2$) states that apart from the detection of critical points, only the manifolds indicated have been constructed. With respect to Delaunay, the penalty factor for building S2 is at most one order of magnitude, but that for the construction of U2 is much higher. Explanations follow from the following three observations, inferred from Table 3 and Figs. 10 and 11: first, while the number of cascaded constructions n_{S2} for S2 may be large (164 for Fish, 13 for BootSki), the average is very low, showing that many of the edges of critical triangles are actually Gabriel edges; second, the statistic n_{U2} clearly dominates n_{S2}; third, the difference between n_{U2} and n_{WCC}, although significant, shows that in flowing from a Voronoi vertex, long sequences of cascaded constructions do occur.

A striking example is the BootSki model, as the cost of constructing U2 is about 450 times higher than that of computing S2. This model actually consists of half a BootSki —the boot has been divided by a sagittal plane. First observe the model features a very small number of maxima: 54. In computing U2, we need to reach these maxima or infinity. But the *concavity* of the model is such that Voronoi vertices are located far away from the model, so that the orbit of each index two critical point requires a large number of cascades before reaching a maximum or infinity.

Numerics. To assess the numerical difficulties, several parameters are of interest. First, the filter failure rate due to insufficiency in interval precision. Second, the size of each rational number involved —recall that they are only used when intervals are not good enough. Third, the depth of the DAG for each geometric construction.

Before reporting on these, let us mention the primitives involved, classified into four categories in Table 4: (1) functions involved to detect critical points: depending on the simplex dimension, these functions require checking whether a simplex is Gabriel and/or checking whether the simplex is intersected by the affine hull of its dual; (2) functions used to compute drivers: these functions consist of computing the minimum ball enclosing the vertices of a simplex, and thus reduce to squared distances comparisons; (3) functions to detect the Voronoi edge reached by the flow across a Voronoi facet —Fig. 9; (4) functions to construct a new point.

The same table reports the numbers of failures versus calls for S2 and U2, on Mecanic. Filter failure rate is low, except for `Equal` —which is used to detect whether a Voronoi edge reduces to a point, and `Orientation` —which computes the mixed product of Eqns. (2) and (4). Such failures respectively correspond to $n > 4$ (almost) co-spherical points, and to (almost) boundary degenerate flows. The cost of filter failures is varying and can be very high, as it triggers a recursive re-evaluation of all DAG nodes below those involved, using rational arithmetic.

Concerning the size of rational numbers, Fig. 12 gives the histogram of the lengths in bytes (numerator plus denominator) in logarithmic scale for the U2 computation of Mecanic and BootSki. Other models show similar distributions, and the worst case observed over all models is bounded by 2^{13} bytes. Note that we used GMP, which systematically normalizes all values. Finally, the maximum constructions depths are listed in Table 5. For all models, the distribution of these depths was decreasing very quickly.

Practical implications. As a general conclusion, cascades in the flow complex can be accommodated as the size of rational numbers stays within manageable bounds, and the interval arithmetic filter is still effective despite the cascade of operations. Yet, such cascades may prevent the construction from being practical if speed is mandatory. Let us instantiate this conclusion for reconstruction [4] and medial axis approximation [19, 20]. For the former, manifolds sought are close from the sample points, and their construction is expected to be affordable. (Even for a smooth surface, an adversary may design a ε-sample forcing a cascading linear in the number of samples.) On the other hand, unstable manifolds near the medial axis, at least on selected examples, seem much harder to obtain.

7. CONCLUSION

This paper presents the first complete and robust construction of the flow complex of the distance function to a point cloud. We show this construction stems directly from the flow operator, i.e. does not require dedicated algorithms for stable and unstable manifolds of various indices. Degenerate cases and numerical issues are discussed. In particular, we emphasize the fact that exact constructions are mandatory to ensure correctness in the general case, a constraint yielding cascaded constructions.

Application-wise, just as the Delaunay triangulation did not have a deep impact due to the lack of robust and efficient algorithms and software —until the advent of say Qhull, CGAL and Triangle, we anticipate that this paper will foster the use of flow based constructions in geometric modeling.

Yet, this paper raises a number of stimulating and difficult questions, related in particular to the cascading phenomenon. Complexity-wise, inferring the (bit-)complexity of the flow complex construction is an open problem. From a numerical perspective, one may experiment with the use of additional steps of increasing precision in the computations, rather than switching directly from machine precision intervals to full multi-precision rational. Finally, on the geometric side, the (un)stable manifolds being rather elaborate geometric objects, approximation schemes are called for.

Acknowledgments. A. Parameswaran acknowledges the support of the INRIA Internship program.

8. REFERENCES

[1] M. Berger. *A Panoramic View of Riemannian Geometry.* Springer, 2003.

[2] H. Brönnimann, C. Burnikel, and S. Pion. Interval arithmetic yields efficient dynamic filters for computational geometry. *Discrete Applied Mathematics*, 109:25–47, 2001.

[3] J. Bruce and P. Giblin. Growth, motion and 1-parameter families of symmetry sets. *Proc. R. Soc. Edinb., Sect. A*, 104:163–186, 1986.

[4] K. Buchin and J. Giesen. Flow complex: General structure and algorithm. In *CCCG*, 2005.

[5] F. Cazals, F. Chazal, and T. Lewiner. Molecular shape analysis based upon the Morse-Smale complex and the Connolly function. In *ACM Symposium on Computational Geometry*, 2003.

[6] F. Cazals and D. Cohen-Steiner. Reconstruction of compact sets in $\mathbb{R}^3$. *In preparation.*

[7] R. Chaine. A convection geometric-based approach to surface reconstruction. In *Symp. Geometry Processing*, pages 218–229, 2003.

[8] T. Dey, J. Giesen, and M. John. Alpha-shapes and flow shapes are homotopy equivalent. In *ACM Symposium on Theory of Computing*, 2003.

[9] T. K. Dey, J. Giesen, and S. Goswami. Shape segmentation and matching with flow discretization. In F. Dehne, J.-R. Sack, and M. Smid, editors, *Workshop Algorithms Data Strucutres*, pages 25–36, 2003.

[10] T.K. Dey, J. Giesen, E. Ramos, and B. Sadri. Critical points of the distance to an epsilon-sampling on a surface and flow based surface reconstruction. In *ACM SoCG*, 2005.

[11] H. Edelsbrunner. Weighted alpha shapes. Technical Report UIUCDCS-R-92-1760, Dept. Comput. Sci., Univ. Illinois, Urbana, IL, 1992.

[12] H. Edelsbrunner. The union of balls and its dual shape. *Discrete Comput. Geom.*, 13:415–440, 1995.

[13] H. Edelsbrunner. Surface reconstruction by wrapping finite point sets in space. In B. Aronov, S. Basu, J. Pach, and M. Sharir, editors, *Ricky Pollack and Eli Goodman Festschrift*, pages 379–404. Springer-Verlag, 2003.

[14] H. Edelsbrunner, M. Facello, and J. Liang. On the definition and the construction of pockets in macromolecules. *Discrete Appl. Math.*, 88:83–102, 1998.

[15] A. Fabri and S. Pion. A generic lazy evaluation scheme for exact geometric computations. In *Proc. 2nd Library-Centric Software Design*, 2006.

[16] R. Forman. Morse theory for cell complexes. *Advances in Mathematics*, 134:90–145, 1998.

[17] J. Giesen and M. John. Surface reconstruction based on a dynamical system. In *Proceedings of the 23rd Annual Conference of the European Association for Computer Graphics (Eurographics), Computer Graphics Forum 21*, pages 363–371, 2002.

[18] J. Giesen and M. John. The flow complex: A data structure for geometric modeling. In *ACM SODA*, 2003.

[19] J. Giesen, E. Ramos, and B. Sadri. Medial axis approximation and unstable manifolds. In *ACM SoCG*, 2006.

[20] S. Goswami, T. K. Dey, and C. Bajaj. Identifying flat and tubular regions of a shape by unstable manifolds. In *ACM Solid and Physical Modeling*, 2006.

[21] Torbjörn Granlund. GMP, the GNU multiple precision arithmetic library. `http://www.swox.com/gmp/`.

[22] L. Hormander. *Notions of convexity.* Birkhauser, 1994.

[23] A. Lieutier. Any open bounded subset of $\mathbb{R}^n$ has the same homotopy type than its medial axis. In *ACM Solid Modeling*, 2003.

[24] John W. Milnor. *Morse Theory.* Princeton University Press, Princeton, NJ, 1963.

[25] A. Okabe and B. Boots K. Sugihara S. Nok Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams (2nd Ed.).* Wiley, 2000.

[26] J. Palis and W. de Melo. *Geometric Theory of Dynamical Systems.* Springer, 1982.

[27] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, London, UK, 1982.

[28] D. Siersma. Voronoi diagrams and morse theory of the distance function. In O.E. Barndorff-Nielsen and E.B.V. Jensen, editors, *Geometry in Present Day Science.* World Scientific, 1999.

9. TABLES

This section features the Tables and Figures discussed in section 6.

Model	#pts	idx_0	idx_1	idx_2	idx_3	χ	t_D	t_{FC}/t_D	M
schale_S2	2714	2714	6126	4341	928	1	1.03	2.11	40.35
schale_U2	2714	2714	6126	4341	928	1	0.38	13.74	36.75
Mecanic_S2	12593	12593	31632	21886	2846	1	2.30	4.29	103.85
Mecanic_U2	12593	12593	31632	21886	2846	1	2.31	63.57	95.11
BootSki_S2	33711	33711	69097	35441	54	1	8.90	1.59	158.99
BootSki_U2	33711	33711	69097	35441	54	1	8.81	728.84	227.15
Fish_S2	54811	54811	175327	141219	20702	1	13.64	2.12	386.03
Culbuteur_S2	71150	71150	208584	138445	1010	1	15.40	2.36	386.07
blade150k_S2	150001	150001	392200	324361	82161	1	23.04	6.29	1235.67
buddha_S2	542548	542548	1192588	672270	22229	1	159.35	3.62	2520.77

Table 2: Overview. Tags S2/U2 in the model name specify the (un)stable manifolds built.

Model	tag	mean	max	stddev	mode	median
schale_S2	S2	2.20	24	2.47	1	1
schale_U2	U2	7.87	40	7.54	1	5
schale_U2	WCC	5.55	40	6.39	1	1
Mecanic_S2	S2	2.10	20	2.20	1	1
Mecanic_U2	U2	13.69	88	13.86	1	10
Mecanic_U2	WCC	8.55	88	11.35	1	1
BootSki_S2	S2	1.55	13	0.86	1	1
BootSki_U2	U2	48.22	207	26.83	42	47
BootSki_U2	WCC	35.48	207	27.92	1	17
Fish_S2	S2	1.31	164	4.11	1	1
Culbuteur_S2	S2	1.35	106	3.50	1	1
blade150k_S2	S2	2.29	76	2.71	1	1

Table 3: Assessing the *complexities* of S2 and U2. Tag column (S2/U2/WCC) respectively refer to the $n_{S2}/n_{U2}/n_{WCC}$ statistics –see text.

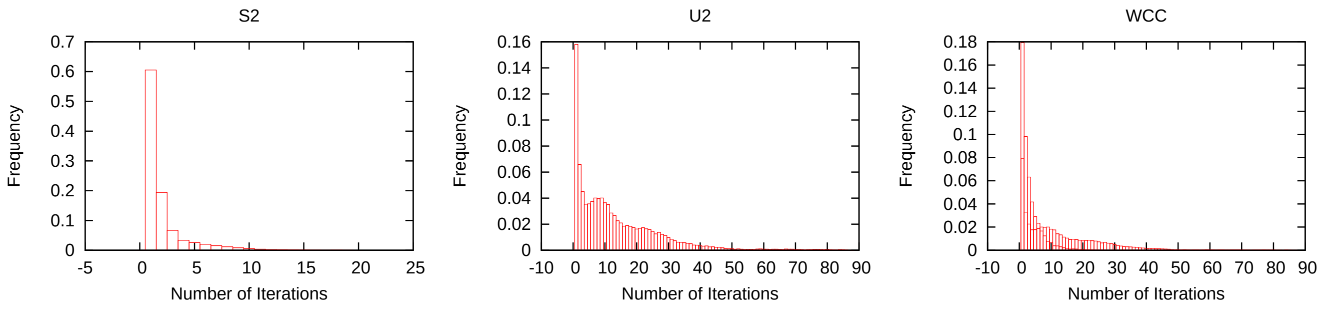

Figure 10: Mecanic: histograms for n_{S2}, n_{U2}, n_{WCC}

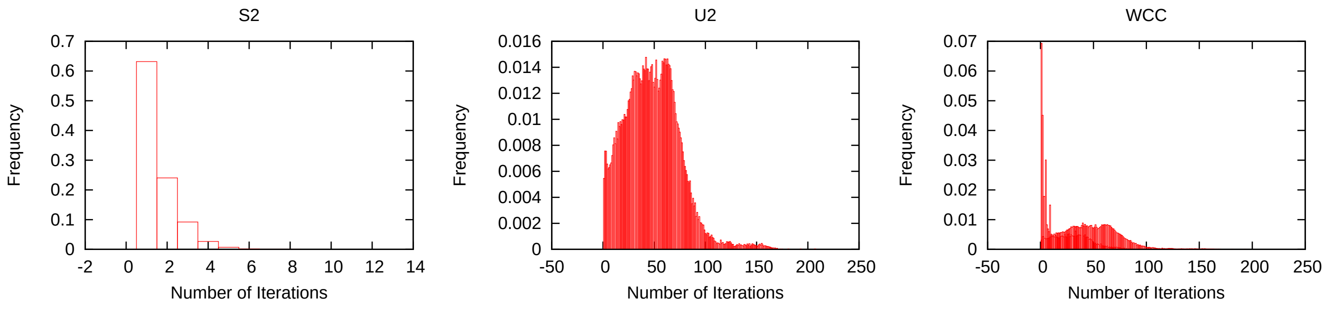

Figure 11: Bootski: histograms for n_{S2}, n_{U2}, n_{WCC}

Operation	S2	U2
(1) Does_simplex_intersect_dual_support(4 points)	21/84160	21/84160
(1) Does_simplex_intersect_dual_support(3 points)	1021/168820	1021/168820
(2) Construct_circumcenter(2 points)	0/49225	0/94352
(2) Construct_circumcenter(3 points)	0/21886	0/145409
(2) Compare_squared_distance(point, point, number)	-	0/880154
(2) Compare_distance(3 points)	-	0/38034
(2) Compute_squared_distance(2 points)	0/56364	0/694958
(3) Equal(2 points)	4225/125337	8902/1024888
(3) Orientation(3 vectors)	0/110565	22079/1385218
(4) Intersect(plane, line)	0/24770	479/292165
(4) Construct_equidistant_line(3 points)	0/24848	0/335384
(4) Construct_ray(point, line)	0/78	0/43219
(4) Construct_segment(2 points)	0/81628	0/1067714
(4) Construct_plane(3 points)	0/24770	0/292165

Table 4: Number of filter failures/calls to predicates and constructions for S2 and U2 for Mecanic

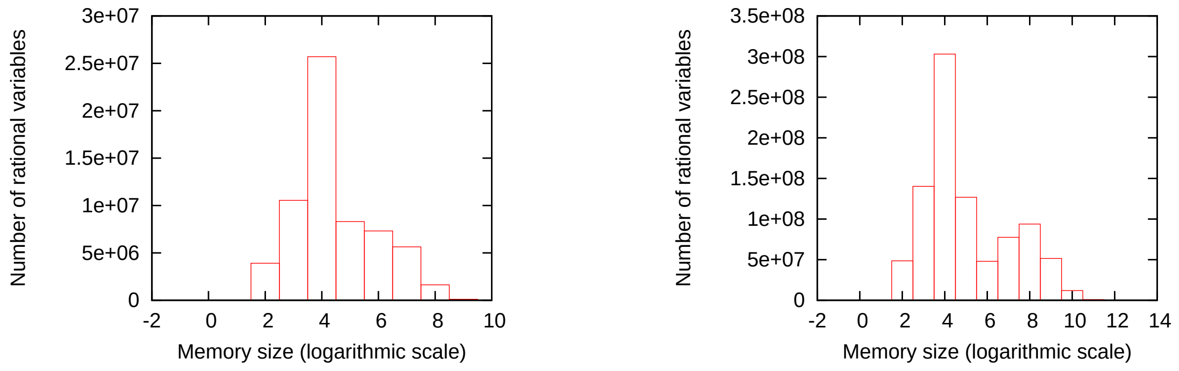

Figure 12: Histograms for the size of rational numbers for U2 for Mecanic and Bootski

Model	S2	U2
Schale	72	70
Mecanic	60	153
BootSki	39	615
Blade	228	x
Buddha	378	x
Culbuteur	318	x
Fish	492	x

Table 5: Maximum construction DAG depth observed for S2 and U2 for all models. x indicates the program does not terminate within 3 hours.