TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

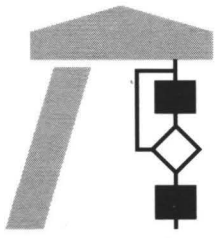
Interner Bericht

Post-processing Polygonal Voxel Data from Numerical Simulation

T. Bobach, G. Umlauf

Oktober 2004

332/04



FACHBEREICH
INFORMATIK

Postfach 3049 · D-67653 Kaiserslautern

Post-processing Polygonal Voxel Data from Numerical Simulation

T. Bobach, G. Umlauf

332/04

Technische Universität Kaiserslautern
Geometrische Algorithmen
Postfach 3049
67653 Kaiserslautern
Germany

Oktober 2004

Herausgeber: Geometrische Algorithmen
Leiter: Professor Dr. Georg Umlauf

Post-processing Polygonal Voxel Data from Numerical Simulation

T. Bobach[1] and G. Umlauf[2]

[1]DaimlerChrysler Research and Technology, Ulm, Germany,
[2]Geometric Algorithms Group, Computer Science Department, University of Kaiserslautern, Germany

Abstract

Many applications dealing with geometry acquisition and processing produce polygonal meshes that carry artifacts like discretization noise. While there are many approaches to remove the artifacts by smoothing or filtering the mesh, they are not tailored to any specific application subject to certain restrictive objectives. We show how to incorporate smoothing schemes based on the general Laplacian approximation to satsify all those objectives at the same time for the results of flow simulation in the application field of car manufacturing.

In the presented application setting the major restrictions come from the bounding volume of the flow simulation, the so-called installation space. In particular, clean mesh regions (without noise) should not be smoothed while at the same time the installation space must not be violated by the smoothing of the noisy mesh regions. Additionally, aliasing effects at the boundary between clean and noisy mesh regions must be prevented. To address the fact that the meshes come from flow simulation, the presented method is versatile enough to preserve their exact volume and to apply anisotropic filters using the flow information.

Although the paper focuses on the results of a specific application, most of its findings can be transferred to different settings as well.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Geometric algorithms, languages, and systems

Keywords: voxel data, smoothing, anisotropic smoothing, Laplacian smoothing.

1. Introduction

Polygonal surfaces are a comfortable and wide-spread way to represent geometric models. One particular problem usually found with surfaces reconstructed from physical scanning is the inherent noise due to the limited hardware precision and the discrete sampling process. Since this makes further processing of the geometry like the derivation of differential geometry properties more difficult, these artifacts must be removed. For this purpose a large pool of mesh smoothing and filtering methods addressing a wide range of aspects has been developed.

Artifacts similar to those from physical scanning can be found in meshes extracted from computation grids as used in numerical simulation. One example are volume discretizations for flow simulation, where the meshes are the exterior boundary surfaces of the simulated flow volume. In this setting the noise comes from discretizing the volume into atomic cells.

Considering the underlying simulation process for the above mentioned example, certain additional constraints can be identified that demand special attention during smoothing, e.g. preserving clean (noise-free) geometry, preserving volume, respecting spatial constraints, etc.

Based on concrete flow simulation data we build a framework that integrates these constraints and provides an interface to smoothing schemes that are based on the general Laplacian approximation. As these schemes define smoothness on the mesh in the same way, we can generically support them in our framework, while being able to address all constraints at the same time. The framework allows to imprint flow information into the smoothing operator. Though this paper describes the processing of a concrete application setting, most results can be transferred to similar settings.

This paper is structured as follows. We first explain the application setting and the resulting objectives for our framework in Section 2. After giving an overview on related work in Section 3, we explain how we meet the objectives from Section 2 and point out how our solution can be transferred to similar problems in Sections 4, 5, and 6. Finally, we give a summary of the presented work and propose future enhancements in Section 7.

2. Application Setting

2.1. Notation

In the following we will refer to a *mesh* as the piecewise linear surface $\mathcal{M} = (\mathcal{K}, \mathcal{P})$. The set $\mathcal{K} = \{(v_1, \ldots, v_n) | v_i \in \{1 \ldots m\}\}$ contains the faces of the polygonal surface, namely 1-faces $(i) \in \mathcal{K}$, 2-faces $(i, j) \in \mathcal{K}$ and n-faces $(i_1, \ldots, i_n) \in \mathcal{K}$, corresponding to *points, edges* and *polygons*. For triangle meshes faces become $(i, j, k) \in \mathcal{K}$. For simplicity, we will refer to a vertex $(i) \in \mathcal{K}$ by i. $\mathcal{P}$ is an ordered set of vertex positions $\{\mathbf{p}_i = \varphi(i) \mid i \in \mathcal{K}\}$ under the geometric realization $\varphi : \mathcal{K} \to \mathbb{R}^3$ with each vertex $\mathbf{p}_i$ associated with its index i.

Furthermore, the one-ring neighborhood for a vertex $i \in \mathcal{K}$ is given by

$$n(i) := \{j \mid (i, j) \in \mathcal{K} \text{ or } (j, i) \in \mathcal{K}\},$$

which is the set of vertices directly connected to it. Thus, $|n(i)|$ is the number of direct neighbors of vertex i.

We use ψ to denote a 3d vector field, which is defined by a vector valued function on $\mathbb{R}^3$. In our setting, a vector field on a mesh is the restriction of the 3d vector field to the vertices of the mesh, $\psi : \mathcal{P} \to \mathbb{R}^3$.

2.2. Data Origin

The meshes we are dealing with originate from a simulation tool that is being developed in the DaimlerChrysler REM/AC division to simulate the channel shape for the most economic air flow through a given installation space [MKR]. The setup for this approach is determined by the installation space, which defines the allowed spatial extent, and the regions for in- and outflow. In order to converge to a stationary flow, the approach uses a sequence of CFD calculations based on finite volumes. This implies initially rendering the installation space into a computation grid composed of atomic cells using volume meshing algorithms.

The result of the simulation is the flow volume, that is swept from inflow to outflow. Its exterior boundary surface must be extracted from the simulation data for later CAD-processing or manufacturing of the channel. The direct extraction of this surface by starting a stream surface from the brim of the inlet is not applicable here as the resulting flow typically contains many vortices and foldings that produce complicated geometries. Instead, the exterior boundary surface of the union of all cells touching the flow is computed. Note that the channel interior is not characterized by scalar values, making isosurface methods useless here. This surface is exported as a polygonal mesh $\mathcal{M}_{sim}$ that inherits the discrete, approximating nature from the computation grid because of cells only partly contained within the flow volume. $\mathcal{M}_{sim}$ may be an open surface due to the inflow/outflow holes.

The resulting discretization noise in $\mathcal{M}_{sim}$ makes later usage of the geometry data more difficult. Nevertheless, there are regions in $\mathcal{M}_{sim}$ that coincide with the boundary of the installation space and thus carry no noise. Therefore, $\mathcal{M}_{sim}$ must be smoothed in noisy regions while at the same time the installation space must not be violated.

For the post-processing of $\mathcal{M}_{sim}$ the boundary of the installation space is also represented as a polygonal mesh $\mathcal{M}_{inst}$. We consider $\mathcal{M}_{sim}$ a 2-manifold without topological defects such as hanging nodes or missing faces. Additionally, the flow information given by a vector field ψ can be used for the smoothing.

2.3. Smoothing Objectives

The setup provided by the simulation tool implies the following objectives for the smoothing framework:

1) **Triangle meshes** $\mathcal{M}_{sim}$ must be triangulated with vertex valences close to six due to subsequent smoothing. (Section 4)
2) **Preserve clean geometry** The smoothing should not affect parts that are not noisy (clean). (Section 4)
3) **Prevent aliasing between clean and noisy parts** The distinction between clean and noisy parts produces aliasing artifacts which must be avoided. (Section 4)
4) **Support of anisotropic (directional) smoothing** The vector field information is available and should be used to tailor the smoothing operator. (Section 5)
5) **Bound the impact of smoothing** To ensure that important geometric features are preserved, while the discretization noise is reliably removed, over-smoothing must be prevented. (Section 5)
6) **Volume preservation** Because the computed flow is sensitive to the dimensions of its bounding geometry, the volume of the given geometry must be preserved. (Section 5)
7) **Comply with installation space** The smoothed meshes must not violate the installation space. (Section 6)

In Sections 4, 5 and 6 we will show how to set up a framework based on Laplacian smoothing that satisfies the above objectives. It provides a semi-automatic procedure to post-process $\mathcal{M}_{sim}$ for further usage in CAD or manufacturing applications. The proposed framework is versatile enough to work with any Laplacian based smoothing scheme.

Objectives 1)-3) are concerned with preparing $\mathcal{M}_{sim}$ for smoothing and are discussed in detail in Section 4. Objec-

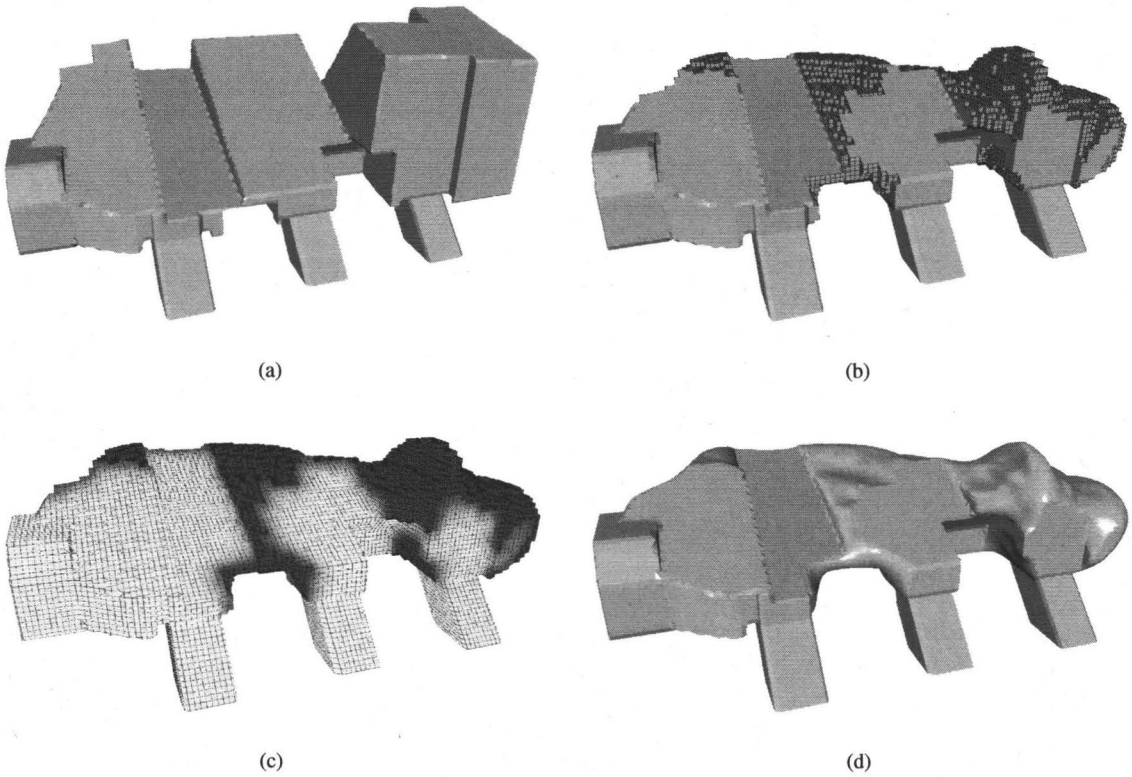


Figure 1: *The pictures illustrate the intended process: (a) shows the installation space $\mathcal{M}_{inst}$, (b) the derived simulated mesh $\mathcal{M}_{sim}$, (c) the distinction between noisy parts $\mathcal{M}_{free}$ (red) and clean parts $\mathcal{M}_{fixed}$ (yellow), (d) the mesh after applying our smoothing framework.*

tives 4)-6) affect the smoothing operation which is described in Section 5. The last objective 7) ensures the validity of the smoothed geometry, as explained in Section 6.

Due to objective 2) we will distinguish between clean parts $\mathcal{M}_{fixed} = \mathcal{M}_{sim} \cap \mathcal{M}_{inst}$, that need no smoothing and must stay fixed, and noisy parts $\mathcal{M}_{free} = \mathcal{M}_{sim} \setminus \mathcal{M}_{fixed}$, that can move freely during smoothing.

2.4. Generalization to other settings

Most of the above mentioned objectives also apply to other settings. Triangular faces (objective 1)) are required wherever existing polygonal surfaces have to be processed by methods working on the simplical representation by triangles. A distinction between noisy and clean mesh regions (objective 2), 3)) is useful for geometries emerging from heterogeneous creation processes. The prevention of oversmoothing (objective 5)) is applicable in more or less every smoothing application, while exact volume preservation (objective 6)) might be negligible in most applications. At last, respecting spatial constraints (objective 7)) becomes crucial mainly in engineering applications where geometric models have to fit in a complex assembly.

3. Related Work

Geometric models are to a growing extend generated from implicit representations as they result from automatic geometry retrieval methods such as laser range scanners and computer tomography, or as a visualization of numerical simulation data. Other sources for initial surface data include isosurface extraction from scalar fields [LC87, KBSS01] and surfaces of voxel volume representations. In order to conveniently work with the data, it has to be transformed into one of the common geometry representations. One of the most versatile representations is the simplicial mesh. The conversion process covers – not necessarily in this order – reconstructing a manifold surface from the scattered data points, de-noising and simplifying it. The objectives accompanying this process are usually the validity of the manifold surface, the preservation of important geometric features and an efficient data representation.

Our setting is partly related to this problem in that we

have an untriangulated, partly noisy manifold representation of the surface. In contrast to most other settings we have to consider also spatial constraints for the modifications we impose on the mesh.

3.1. Mesh Generation

The task of rendering the scattered data into an initial, preferably closed manifold triangulation has been pursued by a large community [AGJ00, EM94, ACK01, DG03]. They all use the available data points as vertices in a triangle mesh with main focus on the validity of the resulting surface. Using one of those approaches to construct a triangulation from the point cloud after stripping the surface off all connectivity information is not feasible, since there is no control over the resulting vertex valences (Objective 1)). This last aspect is important in our setting because smoothing regular meshes leads to a more homogeneous triangle density, while vertices with high valence produce accumulations of smaller triangles and decrease the smooting rate in those regions.

If the surface is already given, triangulations satisfying special constraints can be computed using re-meshing approaches [KVLS99, VRKS01, VRS03] These methods only apply when the initial surface is free of errors and noise.

[SG03] present a method called *connectivity regularization* to minimize the number of irregular vertices in a triangular mesh. Unfortunately, this method assumes the mesh to be a low-noise approximation of the geometry which is not the case with the data in our setting.

3.2. Mesh Smoothing

In a physical digitization process the limited precision of the scanning hardware causes the scan data to be noisy. Surfaces of voxel geometries show similar, usually systematic artifacts. Surface smoothing methods have been developed to remove the artifacts and restore the fair surface. The largest class of approaches defines noise as a function of the mesh vertices, whose position is changed during the smoothing process to minimize the noise.

In simple Laplacian smoothing noise is directly associated with the value of the Laplacian approximation known as umbrella operator for triangle surfaces. Minimizing a functional based on this operator leads to a smooth surface. Since this operator is only an approximation to an energy other functionals have been investigated based on the general Laplacian approximation using e.g. mean curvature flow [DMSB99], scale-dependent Laplacian approximation [Fuj95], or second order Laplacian approximation [KCVS98]. An interpretation of Laplacian based smoothing as a form of signal processing has provided theoretical background on the application of signal processing tools on meshes [Tau95, Ale02]. If the geometric features that must be removed are much larger than the mesh resolution, the unsatisfying performance of the common approaches can be overcome with multi-resolution approaches [GSS99, EDD*95]. On the other hand to avoid over-smoothing thresholds for the vertex displacements can be used [OBB00] (Objective 5)). Nevertheless, none of these approaches allows to incorporate directional information into the smoothing operator (Objective 4)).

Another problem that share almost all methods, is the shrinkage of the volume enclosed by the surface during smoothing [Tau00, DMSB99, GSS99, KCVS98] (Objective 6)). Most methods employ heuristics to lessen the effect of shrinkage. A generic approach to exactly preserve the volume was proposed by undoing the volume difference by uniformly scaling the geometry [DMSB99]. This is, however, not suited to process meshes where parts of the geometry must stay unchanged (Objectives 2) and 3)). The signal processing approach allows to fine tune the method to be nearly volume preserving [Tau95]. To our knowledge there is only one approach that is capable of exactly preserving the volume on an atomic level [LBSP02] (Objective 6)).

Besides the schemes depending on Laplacian approximations, there are approaches based on iso-surface representations in volumetric scalar fields, as investigated by Tasdizen et al. [TWBO03], or methods applying image processing to face normals like those from Ohtake et al. [OBS02, YOB02]. Due to the different surface representations these approaches are not applicable in out setting.

3.3. Collision Detection

Restricting geometric modifications to the interior of given spatial constraints is related to the field of collision detection. This is a topic in robotics, animation and game industry, but also in the simulation of physical systems. Common objectives are temporal integrity, real-time capability and dynamic data structures [JTT01, LG98]. Those do not apply to our setting as we only need to prevent a modification from violating a spatial constraint (Objective 7)). So it suffices to compute the signed distances between points and polygons, which we can speed up using hierarchical search structures such as the Lower-Upper-Bound Tree [JC98] in combination with conservative pruning by precomputing the closest point transform [Mau03].

4. Preparation for Smoothing

4.1. Triangulation

It is common to create mostly regular computation grids by intersecting a regular hexahedral grid with the object volume. This leads to an input mesh $\mathcal{M}_{sim}$ consisting of 3-, 4-, 5- and 6-sided polygons with the wide majority being quadrilaterals. Since smoothing schemes assume triangle meshes, we triangulate the data instead of adapting the

smoothing schemes to arbitrary polygons. As mentioned in Section 3 most triangulation methods are designed to render point clouds into valid simplicial surfaces. Their objectives are usually based on the shape of the resulting triangles.

In our setting most of the mesh connectivity is already given. As most smoothing schemes regularize the mesh, they produce a denser triangle distribution in regions of high vertex valence and vice versa. This is not desirable in many applications (c.f. Section 5). Regular meshes lead to a homogeneous triangle density. Therefore, our objective is to

$$\text{minimize} \quad \sum_{i \in \mathcal{K}} \Delta_i^2, \tag{1}$$

with

$$\Delta_i = \begin{cases} 6 - |n(i)|, & \text{if } i \text{ is an interior vertex of } \mathcal{M}_{sim}, \\ 4 - |n(i)|, & \text{if } i \text{ is a boundary vertex of } \mathcal{M}_{sim}. \end{cases}$$

We chose to implement our own method for two reasons: a combinatoric approach is both intricate to set up and computationally expensive, and stochastic methods lack control. Thus, we developed a greedy algorithm to triangulate an existing polygonal mesh with the goal of keeping the vertex valences close to six. By splitting a face the valence of the affected vertices can only grow. Therefore, the iterative face splits are guided to least increase the overall vertex valences in the grid per step. While this does not prevent valences greater than 6, it lessens the amount of peak valences in favor of more regular valences (c.f. Table 1). The pseudocode of the algorithm is given in Figure 2.

triangulate_polymesh(connectivity $\mathcal{K}$)
  let S the set of all potential new edges,
  S_{min} the subset of S of edges (i,j) with lowest combined valence $cv(i,j) := \max(|n(i)|, |n(j)|)$.
  with $S = \{(i,j) | (i,j) \notin \mathcal{K}, \exists f \in \mathcal{K} : i \in f \wedge j \in f\}$,
  $S_{min} = \{(i,j) \in S | \forall (a,b) \in S : cv(i,j) \leq cv(a,b)\}$.
  while $S_{min} \neq \emptyset$
    choose $s \in S_{min}$ // arbitrary
    split the face containing (i,j) into two faces sharing (i,j)
    update S and S_{min} // only edges adjacent to i or j
  end // while

Figure 2: *The greedy triangulation algorithm.*

The set S_{min} can be efficiently implemented as a priority queue, which is updated after each step for all faces adjacent to a new edge. Since each step reduces the number of vertices for a polygon until all polygons are triangles, the algorithm is guaranteed to terminate.

Afterwards we improve the result by applying edge flips to the inserted edges until no single flip further improves the regularity. A comparison to an ad-hoc triangulation using the first enumerated edge for triangulation is shown in Table 1.

after initial triangulation								
ad hoc	13	598	1196	11229	1025	580	33	32
improved	6	82	1214	12035	1372	1		
after edge flips								
ad hoc	7	328	1006	12076	945	315	25	4
improved	6	78	1144	12186	1295	1		
(valence)	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*

Table 1: *The table shows the histogram of vertex valences for the mesh from Figure 1 after performing the triangulation (two upper rows) and after the following edge flips (two lower rows).*

4.2. Restriction to Noisy Part

Retrieving $\mathcal{M}_{fixed}$ involves finding those vertices in $\mathcal{M}_{sim}$ that also lie in $\mathcal{M}_{inst}$ or a certain epsilon hull around it. This is usually done using the approach mentioned in Section 6.

In the present setting, however, $\mathcal{M}_{inst}$ and $\mathcal{M}_{sim}$ are defined on the same set of vertex positions in 3d. This geometric correspondence, which is depicted in Figures 1(a) and 1(b), simplifies the determination of $\mathcal{M}_{fixed}$ in that we only need to find the set of corresponding vertices

$$\mathcal{P}_{fixed} = \mathcal{P}_{sim} \cap \mathcal{P}_{inst}.$$

Because both meshes being given as distinct datasets, the limited precision of the floating-point representation may cause equal positions to differ if compared on a binary basis. To overcome this the points are inflated to ε-cubes and compared by checking for intersections. In a concrete implementation this is efficiently done using a binary tree based on a tolerant comparison operator.

The binary attribute introduced by weighing vertices i in $\mathcal{M}_{free}$ with $\mu_i = 1$ and those in $\mathcal{M}_{fixed}$ with $\mu_i = 0$ causes aliasing effects at the transition between fixed and free regions. This is due to vertices originating from cells only partly contained within the flow volume that touch the installation space (see Figure 3).

To realize a smooth transition and allow a certain strip of the fixed region to be affected by smoothing we let the weights μ_i of the free vertices diffuse to their neighboring fixed vertices. We modeled the diffusion process by a fixed number of smoothing steps on the weights $\mu_i \in [0,1]$,

$$\mu_i \leftarrow \begin{cases} \frac{1}{|n(i)|} \sum_{j \in n(i)} \mu_j & , \text{if } i \in \mathcal{M}_{fixed} \\ 1 & , \text{else.} \end{cases}$$

Then the weights μ_i are used to impose a local damping on the smoothing operation. An example of smoothing with and without smooth transition is shown in Figure 3(a) and 3(b).

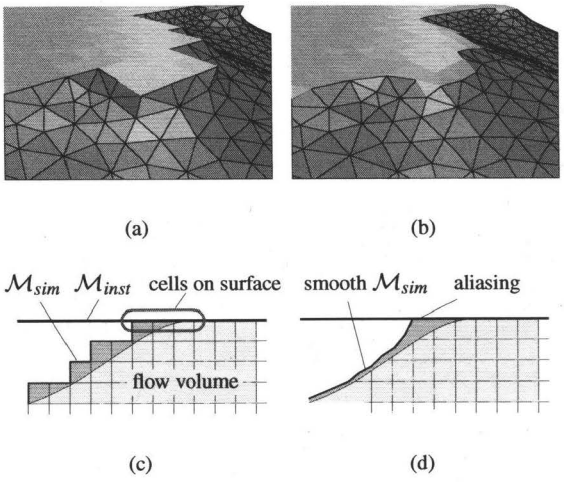


Figure 3: *This figure shows the transition between $\mathcal{M}_{fixed}$ and $\mathcal{M}_{free}$ with aliasing (a) and without (b). Note the harsh ridge in (a). The reason for the aliasing effect between $\mathcal{M}_{fixed}$ and $\mathcal{M}_{free}$ gets clear by comparing the unsmoothed (c) and smoothed (d) mesh.*

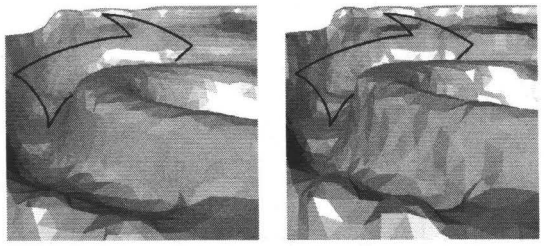


Figure 4: *This figure shows the mesh smoothed isotropically (a), and with directional control of the smoothing operation (b) with the flow direction indicated by the arrows. Geometric features perpendicular to the flow are smoothed in (a) while mainly being kept in (b).*

5. Smoothing

5.1. Laplacian Based Smoothing Schemes

Surface properties are often expressed in terms of energy functionals, whose minimization is directly related to the optimization of the property. For a manifold surface $S \subset \mathbb{R}^3$ which is parameterized by

$$f : \mathbb{R}^2 \to S,$$

such a functional is in most cases defined by the *membrane energy*

$$\mathcal{E}_M(f) := \int f_u^2 + f_v^2 \tag{2}$$

or the *thin plate energy*

$$\mathcal{E}_{TP}(f) := \int f_{uu}^2 + 2f_{uv}^2 + f_{vv}^2.$$

Here f_* denote the first and second derivatives with respect to the parameterization. While $\mathcal{E}_M$ is related to the surface area, $\mathcal{E}_{TP}$ is usually meant to be an approximation of the bending energy. As variational calculus shows, the minimum energy surfaces are characterized by the first and second order Laplace operator

$$\triangle f := f_{uu} + f_{vv} = 0$$

and

$$\triangle^2 f := f_{uuuu} + 2f_{uuvv} + f_{vvvv} = 0.$$

These operators are sensitive to short ripples in a surface S, hence they result in a direct measure for the noise magnitude in surfaces. On a piecewise linear mesh the linear approximation of the first order Laplace operator is given through the general Laplacian approximation

$$\mathcal{L}(\mathbf{p}_i) := w_{ii}\mathbf{p}_i + \sum_{j \in n(i)} w_{ij}\mathbf{p}_j \tag{3}$$

with edge-weights w_{ij} satisfying

$$w_{ii} = -\sum_{j \in n(i)} w_{ij}, \tag{4}$$

which is for a vertex i a function of its one ring neighborhood $n(i)$. A Laplacian based smoothing scheme solves $\mathcal{L}(\mathbf{p}_i) = 0$ for all $i \in \mathcal{K}$ by an explicit or implicit method. All schemes representing some modification of Laplacian smoothing still suffer from some of the following three drawbacks:

Trivial Convergence In the unconstrained setup all vertices are subject to smoothing. The minimization objective for Laplacian smoothing results in a mesh converging to a point or a tense membrane spanned between fixed boundaries in case of the first order Laplacian.

Unsatisfactory Performance on Large Features If the features to be smoothed are much larger than the local face size of the mesh, the local support of the Laplacian operator imposes a significant slowdown of the smoothing.

Parametric Distortion Some applications like texturing are sensitive to parametric distortions. The relaxation process represented by Laplacian smoothing implicitly regularizes the parameterization and thus distorts it.

Except the first, none of these drawbacks affect our setting.

5.2. Directional Smoothing

The vector field ψ defining the stationary flow resulting from the simulation determines the shape of the flow volume.

Therefore, it is natural to use the vector field information for the smoothing of $\mathcal{M}_{sim}$. We achieve this by choosing the edge-weights in the general Laplacian approximation depending on the vector value at the mesh vertices.

The 3d vector field ψ is defined on a mesh of 3d cells with vector values either on the cell vertices or the cell centers and is to be understood interpolatory in between. We incorporate the vector information in the smoothing stencil by modifying the edge-weights, such that the noise term in flow direction is attenuated stronger than perpendicular to it. Consequently, neither amount nor sign of the vector field is needed. Only its direction is important.

Smoothing means minimizing the Laplacian such that the mesh converges to $\mathcal{L}(\mathbf{p}_i) = 0, i \in \mathcal{K}$. Directional information is now incorporated into (3) by damping the influence of values crossing the flow while favoring those in flow direction. This yields the unnormalized weights

$$w'_{ij} = \cos^2 \theta_{ij} \in [0..1]$$

with θ_{ij} being the angle between flow direction $\psi(\mathbf{p}_i)$ and edge $\mathbf{p}_i\mathbf{p}_j$ measured in the plane defined by the vertex normal. The cosine is squared to keep w'_{ij} positive. The balance between isotropic and anisotropic parts of the Laplacian is controlled by a user-defined value μ_{flow}. If w_{ij} are the weights from the isotropic general Laplacian approximation (3), the choice

$$w''_{ij} = w_{ij} + \mu_{flow} w'_{ij}$$

results in isotropic smoothing for $\mu_{flow} = 0$ and completely direction-dependent smoothing for $\mu_{flow} \to \infty$. The sum constraint in (4) requires $\sum_j w_{ij} = 0$. So normalizing w''_{ij} finally leads to the new general Laplacian approximation weights

$$\overline{w}_{ij} = \frac{1}{\sum_j w''_{ij}} w''_{ij},$$
$$\overline{w}_{ii} = -1.$$

This weight definition can be applied to any smoothing scheme based on the general Laplacian approximation (3). The result of applying directional smoothing can be seen in Figure 4.

Since the vector values $\psi(\mathbf{p}_i)$ and the vertex normals depend on the vertex positions $\mathbf{p}_i$, both should be re-evaluated after each smoothing step. The vertex normals are easily recomputed after each smoothing step, while calculating the interpolated vector values requires a complex treatment of the 3d cell structure and the implementation of an interpolation scheme as described in [SML98]. For our setting we assume that face sizes are small compared to the flow features. Thus, it is not necessary to update the flow information at the displaced vertex position after each smoothing step.

5.3. Bounding the Smoothing Impact

Depending on the number of smoothing steps arbitrary large features can be removed. We avoid over-smoothing by adapting the approach of [OBB00]. Originally, they threshold the vertex displacement based on the Laplacian value, which does not allow a geometric interpretation. Instead, we use the mean curvature $\kappa(\mathbf{p})$ as proposed in [MDSB02]. For Laplacian smoothing the vertex displacement for each smoothing step $\mathbf{p}_i \leftarrow \mathbf{p}_i + \lambda\overline{\mathcal{L}}(\mathbf{p}_i)$ is thresholded by

$$\overline{\mathcal{L}}(\mathbf{p}_i) = \begin{cases} \frac{\kappa(\mathbf{p}_i)-T}{\kappa(\mathbf{p}_i)}\mathcal{L}(\mathbf{p}_i) & \text{, if } \kappa(\mathbf{p}_i) > T \\ 0 & \text{, else} \end{cases} \tag{5}$$

with a user-controlled threshold T.

Because the mean curvature κ is a geometric invariant, damping with a threshold value of T forces the smoothing scheme to flatten the surface no more than the bending of a sphere with radius $r = \frac{1}{T}$. If the radius is too small with respect to the surface resolution, the actual notion of a touching sphere defining curvature is no longer valid.

5.4. Implementing the Volume Preserving Scheme

The sixth objective is volume preservation, which practically all sophisticated smoothing schemes address. In our setting the separate treatment of noisy and clean parts of the mesh requires volume preservation at an atomic level. As pointed out in Section 3 the only approach with these capabilities is that of Liu et al. [LBSP02].

Based on the general Laplacian approximation (3), this approach uses the membrane energy (2) under the constraint of volume preservation. Both smoothing and volume preservation are kept local by defining the energy on a *smoothing stencil* for two adjacent vertices i and j and their joined one-ring neighborhoods $\overline{n}(i,j) := n(i) \cup n(j)$ and $n(i,j) := \overline{n}(i,j) \setminus \{i,j\}$ by

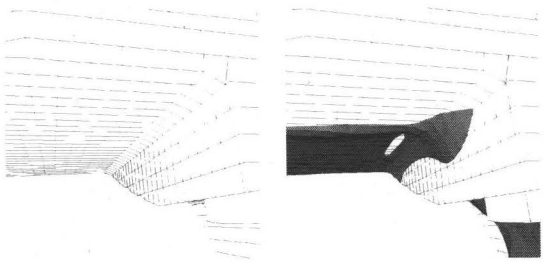
$$\begin{aligned} \mathcal{E}(i,j) &= \frac{1}{2}\left(\mathcal{L}(i)^2 + \mathcal{L}(j)^2 + \sum_{k\in n(i,j)} \mathcal{L}(k)^2\right) \\ &= \frac{1}{2}\sum_{k\in\overline{n}(i,j)}\left(w_{kk}\mathbf{p}_k + \sum_{l\in n(k)} w_{kl}\mathbf{p}_l\right)^2 \end{aligned} \tag{6}$$

with weights w_{ij} from the general Laplacian approximation (3). Liu et al. keep the volume constrained minimization of (6) with respect to the positions of $\mathbf{p}_i$ and $\mathbf{p}_j$ linear by decomposing it into two linear steps. First (6) is encoded as a least squares problem

$$\mathcal{E}(i,j) = \frac{1}{2}\|\mathbf{KP} - \mathbf{Q}\|^2,$$

which is solved with respect to $\mathbf{P} = (\mathbf{p}_i\mathbf{p}_j)^T$. This leads to a flattened stencil with new vertex positions $\mathbf{p}'_i$ and $\mathbf{p}'_j$ and a volume difference ΔV_0.

In a second step (6) is minimized again with the additional



(a) (b)

Figure 5: *This figure shows a closeup of $\mathcal{M}_{sim}$ (a), and the result of smoothing without collision detection (b).*

constraint that the volume changes by $\Delta V_1 = -\Delta V_0$. To linearize this constraint both vertices are displaced by the same vector $\mathbf{w}$, which allows to express ΔV_1 as

$$\Delta V_1(\mathbf{w}) = \mathbf{n}^T \mathbf{w}, \tag{7}$$

where $\mathbf{n}$ is a fix vector depending only on the vertices of the smoothing stencil. The additional constraint (7) is included using Lagrange multipliers.

Since the method is based on equation (6), our modification of the weights $\overline{w}_{ij}$ integrates seamlessly.

6. Collision Detection

In order to restrict the geometric modifications to the interior of the installation space, we initially assume that $\mathcal{M}_{sim}$ is completely contained within $\mathcal{M}_{inst}$ and discard each atomic operation that would lead to an intersection of $\mathcal{M}_{sim}$ with $\mathcal{M}_{inst}$. In cases where $\mathcal{M}_{inst}$ and $\mathcal{M}_{sim}$ are not closed, the convex hull can be used to reintroduce the notion of interior and exterior in the vicinity of the surface hole. As long as $\mathcal{M}_{inst}$ generously covers the affected regions of $\mathcal{M}_{sim}$, no such distinction should be necessary.

The restriction to the interior is accomplished by detecting collisions between $\mathcal{M}_{sim}$ and $\mathcal{M}_{inst}$. The necessity of applying collision detection is illustrated in Figure 5, which shows a case where the smoothed mesh violates $\mathcal{M}_{inst}$. Although this detection can be computed in the neighborhood of the local smoothing operation, we face numerical problems where both meshes touch. As a robust, though approximate alternative we use the signed point-surface distance to detect if a modified vertex position leaves the installation space. This is only possible if interior and exterior of the installation space can efficiently be determined, so we assume $\mathcal{M}_{inst}$ to be oriented with its surface normals pointing outside. If $\mathcal{M}_{inst}$ and $\mathcal{M}_{sim}$ have a similar mesh resolution, this check is sufficient in practice in most cases.

We limit the combinatoric complexity of the distance computation by organizing the faces of $\mathcal{M}_{inst}$ in a search structure. Since the triangles have spatial extent, we employ the LUB (Lower Upper Bound) tree [JC98]. It approximates objects by their bounding boxes, whose projections to the coordinate axes are intervals. These are arranged in a ternary, best balanced search tree, the so-called LUB tree. It provides a small set of triangles that contains the closest triangle to the query point. The average case of a nearest neighbor query runs in logarithmic complexity with respect to the number of mesh triangles.

To determine on which side of $\mathcal{M}_{inst}$ a query point $\mathbf{q}$ lies, the shortest distance has to be computed to the triangles returned by the nearest neighbor query to the LUB tree. Since $\mathcal{M}_{inst}$ is a triangle mesh, the closest point $\mathbf{c}$ on $\mathcal{M}_{inst}$ to $\mathbf{q}$ lies either on a triangle, on an edge or at a vertex. We distinguish between interior and exterior of $\mathcal{M}_{inst}$ by comparing the ray from $\mathbf{c}$ to $\mathbf{q}$ with the surface normal at the $\mathbf{c}$.

In general, $\mathcal{M}_{inst}$ represents an oriented surface and from either of its sides emanate the 3d Voronoi regions of its entities. By definition, a query point $\mathbf{q}$ lies in the Voronoi region of its closest mesh entity $e \in \{face, edge, vertex\}$. If e is a face, $P^+(e)$ is the volume generated by sweeping e along its positive normal direction. If e is an edge and l the line containing e, then the wedge $W^+(e)$ is the set of all points $l + \lambda\mathbf{n}_0 + \mu\mathbf{n}_1$, $\lambda, \mu > 0$, where $\mathbf{n}_0$ and $\mathbf{n}_1$ are the normals of the faces adjacent to e. Now it suffices to decide for the closest entity e if the point $\mathbf{q}$ is contained in the Voronoi region $V^-(e) \subset \mathbb{R}^3$ emanating to the inner side of $\mathcal{M}_{inst}$ or the Voronoi region $V^+(e)$ emanating to the outer side:

- e **is a face:** $\mathbf{q} \in V^+(e)$, if $q \in P^+$.
- e **is an edge:** $\mathbf{q} \in V^+(e)$, if $P^+(f_0) \cap P^+(f_1) \subseteq e$ for the faces f_0 and f_1 adjacent to e.
- e **is a vertex:** $\mathbf{q} \in V^+(e)$, if it lies in every $W^+(t)$ of an edge t emanating from e with $P^+(f_0) \cap P^+(f_1) \subseteq t$ for the faces f_0 and f_1 adjacent to t.

Otherwise, the point lies on the other side of the surface or on it. In a practical implementation, all Voronoi regions intersect by an epsilon to make the computation watertight. For meshes with boundary the above classifications must be adapted.

Note that this procedure does not detect self-intersections, which we do not expect to occur in normal cases.

The displacement of a point can only cause it to leave the installation space, if it is larger than the closest distance between the point and $\mathcal{M}_{inst}$. By initially assigning each vertex of $\mathcal{M}_{sim}$ its closest distance to $\mathcal{M}_{inst}$ in a preprocessing step, we are able to avoid most of the closest point calculations during smoothing.

7. Conclusion and Future Work

We have presented a framework for post-processing of geometry data originating from flow simulation on discrete

computation grids. The concrete application setting from a simulation tool in car manufacturing motivated to demand restricting constraints on the post-processing: generation of a triangle mesh, distinction between clean and noisy geometry including aliasing effects, consideration of the flow field, bounding the smoothing impact, volume preservation and respecting the installation space. We showed how to incorporate these objectives with the general Laplacian approximation, so that the constructed framework can be applied to any Laplacian based smoothing scheme. This results in a semi-automatic conversion of the noisy simulation mesh to a fair triangle mesh suitable for further processing.

Our future work will focus on enhancing the presented framework. Topics that we will address concern the automatic choice and optimization of the remaining free parameters in our framework. For example, the threshold for bounding the smoothing impact should be determined based on the known average error in the mesh. Also we will improve the approximate collision detection procedure to produce always exact results.

References

[ACK01] AMENTA N., CHOI S., KOLLURI R. K.: The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications* (2001), ACM Press, pp. 249–266. 4

[AGJ00] ADAMY U., GIESEN J., JOHN M.: New techniques for topologically correct surface reconstruction. In *Proceedings of the conference on Visualization '00* (2000), IEEE Computer Society Press, pp. 373–380. 4

[Ale02] ALEXA M.: Wiener filtering of meshes. In *Proceedings of International Conference on Shape Modeling and Applications* (2002), pp. 51–57. 4

[DG03] DEY T. K., GOSWAMI S.: Tight cocone: a water-tight surface reconstructor. In *Proceedings of the eighth ACM symposium on Solid modeling and applications* (2003), ACM Press, pp. 127–134. 4

[DMSB99] DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. vol. 33 of *Annual Conference Series*, pp. 317–324. 4

[EDD*95] ECK M., DEROSE T., DUCHAMP T., HOPPE H., LOUNSBERY M., STUETZLE W.: Multiresolution analysis of arbitrary meshes. vol. 29 of *Annual Conference Series*, pp. 173–182. 4

[EM94] EDELSBRUNNER H., MÜCKE E. P.: Three-dimensional alpha shapes. *ACM Trans. Graph. 13*, 1 (1994), 43–72. 4

[Fuj95] FUJIWARA K.: Eigenvalues of Laplacians on a closed Riemannian manifold and its nets. In *Proceedings of AMS 123* (1995), pp. 2585–2594. 4

[GSS99] GUSKOV I., SWELDENS W., SCHRÖDER P.: Multiresolution signal processing for meshes. *Computer Graphics Proceedings (SIGGRAPH 99)* (1999), 325–334. 4

[JC98] JOHNSON D. E., COHEN E.: A framework for efficient minimum distance computations. In *Proc. IEEE Intl. Conf. Robotics & Automation, May 16-21* (1998), pp. 3678–3684. 4, 8

[JTT01] JIMENÉZ P., THOMAS F., TORRAS C.: 3d collision detection: A survey. *Computers and Graphics 25*, 2 (2001), 269–285. 4

[KBSS01] KOBBELT L., BOTSCH M., SCHWANECKE U., SEIDEL H.-P.: Feature-sensitive surface extraction from volume data. *Computer Graphics (SIGGRAPH '01)* (2001). 3

[KCVS98] KOBBELT L., CAMPAGNA S., VORSATZ J., SEIDEL H.-P.: Interactive multi-resolution modeling on arbitrary meshes. *Computer Graphics 32*, Annual Conference Series (1998), 105–114. 4

[KVLS99] KOBBELT L., VORSATZ J., LABSIK U., SEIDEL H.-P.: A shrink wrapping approach to remeshing polygonal surfaces. In *Computer Graphics Forum (Eurographics 1999)* (1999), vol. 18, pp. 119–130. 4

[LBSP02] LIU X., BAO H., SHUM H.-Y., PENG Q.: A novel volume constrained smoothing mehtod for meshes. *Graphical Models 64(3-4)* (2002), 169–182. 4, 7

[LC87] LORENSEN W., CLINE H.: Marching cubes: a high resouition 3d surface extraction algorithm. *Computer Graphics (SIGGRAPH '87) 21*, 4 (1987), 163–169. 3

[LG98] LIN M. C., GOTTSCHALK S.: Collision detection between geometric models: a survey. In *Proceedings of IMA Conference on Mathematics of Surfaces* (1998). 4

[Mau03] MAUCH S.: *Efficient Algorithms for solving Hamilton-Jacobi Equations*. PhD thesis, California Institute of Technology, Pasadena, California, 2003. 4

[MDSB02] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *VisMath* (Berlin, Germany, 2002). 7

[MKR] MOOS O., KILIMETZEK F. R., ROSSMANN R.: Bionic optimization of air-guiding systems. to appear at SAE conference, 2004. 2

[OBB00] OHTAKE Y., BELYAEV A. G., BOGAEVSKI I. A.: Polyhedral surface smoothing with simultaneous mesh regularization. In *GMP* (2000), pp. 229–237. 4, 7

[OBS02] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: Mesh smoothing by adaptive and anisotropic gaussian filter. In *Vision, Modeling, and Visualization* (Erlangen, Germany, 2002), Suzuki H., Martin R., (Eds.), IEEE, pp. 203–210. 4

[SG03] SURAZHSKY V., GOTSMAN C.: Explicit surface remeshing. In *Proceedings of Eurographics Symposium on Geometry Processing* (June 2003), pp. 17–28. 4

[SML98] SCHROEDER W., MARTIN K., LORENSEN B.: *The Visualization Toolkit.* Prentice Hall, Upper Saddle River, NJ 07458, 1998. 7

[Tau95] TAUBIN G.: A signal processing approach to fair surface design. *Computer Graphics 29* (1995), 351–358. 4

[Tau00] TAUBIN G.: Geometric signal processing on polygonal meshes. *Eurographics State Of The Art Reports* (august 2000), 107–117. 4

[TWBO03] TASDIZEN T., WHITAKER R., BURCHARD P., OSHER S.: Geometric surface processing via normal maps. *ACM Transactions on Graphics* (2003). to appear. 4

[VRKS01] VORSATZ J., RÖSSEL C., KOBBELT L., SEIDEL H.-P.: Feature sensitive remeshing. In *Eurographics 2001* (2001), Chalmers A., Rhyne T.-M., (Eds.), vol. 20. 4

[VRS03] VORSATZ J., RÖSSL C., SEIDEL H.-P.: Dynamic remeshing and applications. In *Proceedings of the eighth ACM symposium on Solid modeling and applications* (2003), ACM Press, pp. 167–175. 4

[YOB02] YAGOU H., OHTAKE Y., BELYAEV A.: Mesh smoothing via mean and median filtering applied to face normals. In *Geometric Modeling and Processing* (RIKEN, Saitama, Japan, 2002), Suzuki H., Martin R., (Eds.), IEEE, pp. 124–131. 4