SFU Computing Science Technical Report TR 2006-13 (2006), pp. 1–11

Guaranteed Nonobtuse Meshes via Constrainted Optimizations

John Y.S. Li[1] and Hao Zhang[1]

[1]GrUVi Lab, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada

Abstract

The problem of nonobtuse triangulation has been studied in the 2D domains, however, guaranteed nonobtuse remeshing of curve surfaces is still an open problem. Nonobtuse meshes are desirable in certain situations such as geodesic computations and planar mesh embedding. In this paper, we propose a solution to nonobtuse remeshing and nonobtuse decimation. Our approach utilizes a "deform-to-fit" strategy to solve the remeshing problem. We first construct an initial nonobtuse mesh, via a modified version of the Marching Cubes algorithm, that roughly approximates the input mesh. An iterative constrained optimization is then applied to obtain a high quality approximation of the input mesh. At each iteration, the constraints ensure the output mesh is guaranteed nonobtuse. By making use of quadric-based errors, we iteratively decimate the high-detail nonobtuse mesh in a similar constrained manner to obtain a hierarchy of nonobtuse meshes.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

A *nonobtuse triangle mesh* is composed of a set of triangles in which every angle is less than or equal to 90°; we call these triangles nonobtuse triangles. If each (triangle) face angle is strictly less than 90°, then the triangle mesh is said to be *acute*. Immediate benefits of having a nonobtuse or acute mesh include more efficient and more accurate geodesic computation on meshes using fast marching [KS98], guaranteed validity for planar mesh embeddings via discrete Harmonic maps [EDD*95], among others.

To the best of our knowledge, there are no known algorithms that are guaranteed to produce a nonobtuse or acute triangle mesh which either interpolates or accurately and smoothly approximates a given point cloud. The same holds for the related remeshing problem, where one seeks a nonobtuse or acute triangle mesh which approximates a given mesh. To make effective use of nonobtuse or acute meshes in interactive applications, it is also desirable to be able to construct a model hierarchy where at each level of details, we have an nonobtuse or acute mesh. We refer to this problem as that of nonobtuse or acute mesh decimation.

In this paper, we focus on nonobtuse remeshing and decimation. However, the framework we develop is quite general and flexible. It would also allow for nonobtuse mesh generation from point clouds and hold great promise for solving the acute version of the same problems. To make our presentation concise, we shall only deal with closed, 2-manifold triangle meshes. Our approach can handle meshes with boundary but require minor changes to the algorithm. A brief discussion of the boundary case is presented in Section 8.

We formulate both the nonobtuse remeshing and nonobtuse mesh decimation problems in the context of constrained optimization, where the search space is restricted to the set of nonobtuse meshes. Since a global optimization is expected to be computationally intractable, we resort to greedy schemes driven by appropriately defined costs.

For the remeshing problem, it is important to obtain an initial nonobtuse mesh which approximates the input mesh; it serves as the starting point of the greedy search. To this end, we propose to convert the input mesh into a signed distance field first and then use a modified, midpoint-based Marching Cubes algorithm to compute the initial nonobtuse mesh. An added advantage of using Marching Cubes is that we can

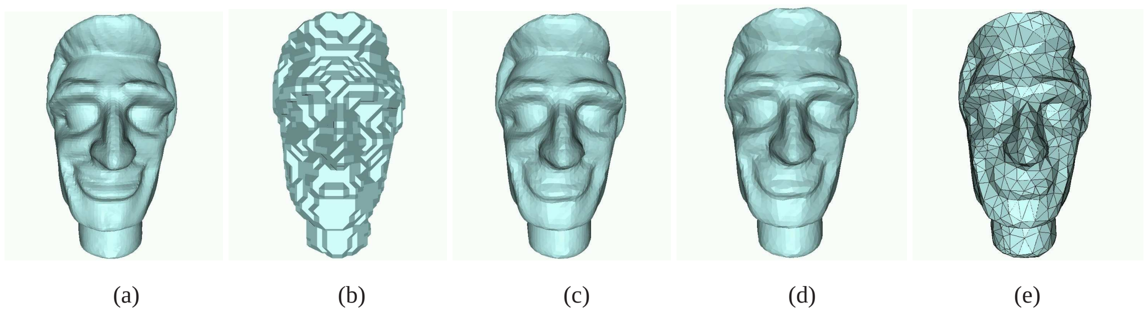

Figure 1: ***Major components of our approach.*** *(a) Original "bigsmile" model (34,712 faces). (b) Result of our modified Marching Cubes. The new mesh (15,380 faces) is nonobtuse but shows visible artifacts. (c) Result from constrained optimization, which is a high-quality approximation. The approximation error,* $\varepsilon = 0.78\%$*, is the Hausdorff distance computed from Metro [CRS98], given as a percentage of the bounding box diagonal. (d) After decimating* 50% *of the vertices (nonobtusely). Error measured against the mesh in (c):* $\varepsilon = 0.107\%$*. (e) After decimating* 90% *of the vertices. Error against (c):* $\varepsilon = 0.61\%$*.*

bound the size of the resulting mesh by properly adjusting the resolution of the cubical sampling grid.

In the subsequent search, we interleave mesh optimization steps with Laplacian smoothing, both honoring the constraint that the mesh be nonobtuse. Mesh optimization relies on a greedy scheme and highly localized constrained optimization to reduce approximation error while ensuring smoothness or regularity of the result. The smoothing step encourages tangential movement of the vertices and can help steer the algorithm toward better local minima. This can further improve the approximation quality and angle distribution (attempting to avoid small face angles) of the solution, while maintaining the nonobtuse guarantee.

Nonobtuse mesh decimation starts with an initial nonobtuse dense mesh and follows the standard quadric-based decimation procedure [GH97] augmented by appropriate constraints to ensure good approximation quality and the nonobtuse guarantee. Figure 1 illustrates the major components of our approach as we have described.

The rest of the paper is organized as follows. After briefly reviewing related previous work in Section 2, we formulate the problem we wish to solve in Section 3. We also discuss several obvious attempts at a solution and the associated difficulties. Next, we give an overview of our approach and expand its coverage in subsequent sections. Specifically, Section 4 describes scalar field construction from an input mesh and our modified, midpoint-based Marching Cubes scheme to ensure triangle nonobtusity. Next, we describe our constrained optimization scheme for remeshing in Section 5. Nonobtuse mesh decimation is covered in Section 6. In Section 7, experimental results are provided. Finally, we conclude and comment on possible future work.

2. Related work

Acute triangulation of special planar domains, e.g., a triangle or a square, has been a topic of recreational mathematics and studied fairly extensively, e.g., see [CL80]. An enlightening discussion on acute square triangulation can be found from Eppstein's Geometry Junkyard [Epp97]. Motivated by the space-time meshing algorithm of Üngör & Sheffer [US02], Eppstein et al. [ESU04] study the problem of tiling a 3D domain using tetrahedra with only acute dihedral angles. This turns out to be highly non-trivial and they propose only constructions for acute tiling of the 3D Euclidean space and slabs; this has applications in mesh generation [ESU04].

Evidence suggests that the nonobtuse triangulation problem, with a weaker angle constraint than its acute counterpart, is easier, but by no means trivial. The best known work in 2D is due to Bern et al. [BMR94], who give an $O(n\log^2 n)$ algorithm for nonobtuse triangulation of a n-vertex polygon, possibly with holes, so that the number of triangles used is only $O(n)$. An extension to d-dimensional point sets [BCER95] gives an upper bound of $O(n^{\lceil d/2 \rceil})$, but also shows that no similar result is possible for the acute case. Many other quality criteria for mesh generation exist, e.g., see the survey of Bern & Eppstein [BE92], and variants of the Delaunay triangulation (DT) are frequently applied, e.g., [ACSYD05, Che93, CS05].

Surface mesh generation and remeshing of triangle meshes are well studied problems in computer graphics, e.g., see the recent survey by Alliez et al. [AUGA05]. Two existing algorithms for high-quality meshing of curved surfaces with guaranteed angle bounds are both Delaunay-based. Chew [Che93] develops a refinement scheme based on constrained DT to ensure that all angles are in the range $[30^\circ, 120^\circ]$. Cheng & Shi [CS05] use restricted union of balls to generate an ε-sampling of a surface and extract a mesh from the DT of the sampling points. A lower bound on the minimum angle can be as high as 30° with proper choice of parameters. But to approach this bound, ε needs to be close to zero which would result in a high triangle count. Both algorithms fall short of producing guaranteed acute or nonobtuse meshes; this has been identified as an open problem by Gu & Yau [GY03], among others, and so far only some simple heuristics [GY03] have been suggested.

In this paper, we propose to solve the nonobtuse remeshing problem, while allowing for control of the size and angle quality of the mesh produced. A hierarchy of nonobtuse meshes can then be obtained via constrained mesh decimation. One of the benefits of having an nonobtuse mesh is more efficient and more accurate geodesic computations on meshes via fast marching [KS98]. Also, Floater [Flo98] has shown that the discrete Harmonic maps of Eck et al. [EDD*95] may not produce a valid planar mesh embedding due to the existence of negative weights. Nonobtusity of a mesh offers a sufficient condition for eliminating these negative weights and thus ensures a one-to-one mapping.

3. Problem formulation and possible approaches

Several interesting problems involving acute or nonobtuse meshing can be formulated, but we only discuss those related to the latter and list them below. The acute version of these problems are trivially implied.

- **Problem 1 (Nonobtuse remeshing):** Given input mesh M, construct a nonobtuse mesh $\hat{M}$, with *any connectivity*, which smoothly and accurately approximates M.
- **Problem 2 (Nonobtuse mesh generation):** Same as Problem 1, except the input is now a *point cloud*.
- **Problem 3 (Remeshing with given connectivity):** Same as Problem 1, except that the connectivity is given.
- **Problem 4 (Mesh generation with given connectivity):** Same as Problem 3, except the input is a point cloud.
- **Problem 5 (Nonobtuse triangulation):** Given a point cloud S, construct a nonobtuse mesh that *interpolates* S.

Problem 1 is what we wish to solve in this paper. As we shall describe, we use a constrained “deform-to-fit” strategy, starting with an approximating nonobtuse mesh obtained via modified Marching Cubes [LC87]. Input to this step, namely, a scalar field of signed distances from grid points to the surface to be meshed, can be derived from a given mesh or a point cloud in a variety of ways, e.g., [HDD*92]. Thus to apply our approach to solve Problem 2, a variation of Problem 1, we only require a distance measure between 3D points and the underlying surface conveyed by the point cloud that is feasible to use for our optimization.

With the mesh connectivity fixed, as for Problems 3 and 4, local smoothness of a nonobtuse solution may not be ensured if degree-3 vertices are present, since at such a vertex, the discrete Gaussian curvature has a lower bound of $\pi/2$. It is worth noting however that for a closed mesh, both degree-3 and degree-4 vertices can be easily eliminated without modifying mesh geometry. In Figure 2, we show two schemes

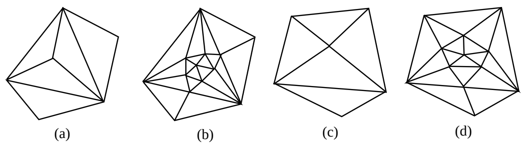

Figure 2: ***Elimination of degree-3 and degree-4 vertices.*** *(a) A degree-3 vertex and its neighborhood. (b) Seven new vertices are inserted and the resulting configuration removes the degree-3 vertex without creating any new degree-3 or degree-4 vertices. In addition, none of the existing vertices have their degrees decreased. (c) A degree-4 vertex and associated modification in (d). Note that one can easily have appropriate edges aligned so that (b), respectively, (d), does not differ from (a), respectively, (c), geometrically.*

that we have come up with. In general, higher vertex degrees appear to be more desirable as far as nonobtuse meshing is concerned. But even without degree-3 or degree-4 vertices, it is still unclear how a obtuse mesh can be deformed into a nonobtuse one, while ensuring adequate final mesh quality.

Interpolatory nonobtuse triangulation of a point cloud (Problem 5) appears to be the hardest. A related problem is to find a nonobtuse triangle meshing of a curved region whose boundary needs to be respected. A solution may not exist for any given point cloud or region boundary, unless Steiner points are allowed to be added. Even so, it is quite conceivable that a solution would be hard to find, judging from the difficulty of producing guaranteed nonobtuse meshes in the much simpler domain of 2D polygons [BMR94] or meshing curved surfaces under weaker angle conditions [Che93, CS05]. Minimizing the number of Steiner points used is likely a hard problem as well.

3.1. Some obvious attempts and difficulties

2D nonobtuse triangulation + mesh parameterization: An obvious attempt to solve the nonobtuse remeshing problem would be to take advantage of known solutions in 2D [BMR94]. One may divide a given mesh into patches and map each patch boundary into a planar polygon, apply nonobtuse polygon triangulation, and map the resulting planar triangle mesh back to 3D via an angle-preserving mesh parameterization. However, the nonobtusity of the 3D mesh can be guaranteed only if the parameterization has a guaranteed angle distortion bound, say λ, and that the maximum angle in the planar mesh is α, and that $\lambda\alpha \leq \pi/2$. We are not aware of any combination of existing algorithms that can achieve these. Furthermore, nonobtuse polygon triangulation [BMR94] necessitates the addition of Steiner points along polygon boundaries. It is unclear how consistence along shared edges between adjacent patches can be achieved so as to avoid T-vertices altogether.

Advancing fronts: One may also consider an incremental "tiling" of a given mesh surface using nonobtuse triangles, akin to the SwingWrapper approach [AFSR03]. Obviously, difficulties will arise as the advancing front is close to meeting with itself and no obtuse triangles can fill the gaps. Devising either a back-tracking or a look-ahead strategy which would provide a guarantee is quite difficult.

Local mesh manipulations: Edge flipping, mesh subdivision followed by heuristic edge collapse [GY03], and constrained iterative vertex movements are all worth considering. They all attempt to eliminate a small number of obtuse angles step by step, but none of them offer any guarantee. In the latter, the process may "get stuck" where obtuse angles still exist, but no vertex can be moved without having the resulting shape deviate significantly from the original mesh.

3.2. Our approach

None of the attempts described above guarantee the nonobtusity of the mesh produced; this appears to be the most difficult aspect of the meshing or remeshing problem. Our main idea for solving the nonobtuse remeshing problem is to first construct a mesh that is guaranteed to be nonobtuse, although it may not be a adequate approximation to the original mesh, and then iteratively improve its approximation quality while ensuring that nonobtusity is maintained.

Given a degree-k mesh vertex v, we refer to the *nonobtuse region* $\mathcal{N}(v)$ for v as the set of points which would form k nonobtuse triangles with the one-ring vertices of v. Note that $\mathcal{N}(v)$ can be empty in an arbitrary mesh, but for a nonobtuse mesh, $v \in \mathcal{N}(v) \neq \emptyset$. We constrain the movement of each vertex to be within its nonobtuse region and wish to find a final mesh that is the closest, in terms of a least-square error measure, to the original mesh. This is a constrained global optimization problem, with a complex search domain that is the set of all nonobtuse meshes having a particular connectivity. Our strategy then is to apply a heuristic search in the search space with a judicious choice of a starting point. To this end, we propose to use a modified Marching Cubes algorithm to construct an initial mesh, guaranteed to be nonobtuse, which roughly approximates the original mesh.

A computational obstacle in the subsequent optimization is that the nonobtuse constraints are nonconvex and nonlinear with respect to vertex positions. To make the optimization problem more feasible to solve, we conservatively linearize and convexify the constraints and transform the general nonlinear optimization problem into *quadratic programming* with a convex search domain. Note that our linearization and convexification step can reduce the size of the allowable search region (not to the empty set however). But by being conservative, the constraints still ensure that we always obtain a nonobtuse mesh at each step.

Once a final, high-quality nonobtuse mesh, in terms of the approximation error, angle quality, and smoothness, is arrived at, we can perform nonobtuse decimation via a constrained greedy edge collapse, based on quadric errors. This procedure resembles the quadric-based mesh decimation algorithm of Garland and Heckbert [GH97].

4. Initialization using modified Marching Cubes

Originally appeared in the seminal paper of Lorensen & Cline [LC87], the Marching Cubes algorithm produces a triangle mesh which tessellates the zero level-set (surface) of a given 3D scalar field. The scalar field is first discretized at the vertices of a regular cubical grid. Within each cube, depending on the signs of the scalar values at the cube vertices, different sets of triangles are generated. Taking into consideration of complementary and rotational symmetry, 15 distinguishable cases (see Figure 3) were given in [LC87]. Subsequently, Nielson & Hamann [NH91] improve upon the original algorithm by resolving possible ambiguities that may generate holes in the mesh produced. As a result, several new cases have been introduced.

4.1. Modified Marching Cubes

In order for the mesh produced to reflect the geometry of the zero level-set more faithfully, the original Marching Cubes algorithm [LC87] uses linear interpolation along each cube edge to determine the location of mesh vertices. Certain cases, e.g., cases 1, 3, 4, 7, and 13 (see Figure 3), only produce nonobtuse triangles; this can be easily verified via the Cosine Theorem. In general however, obtuse angles can be introduced in all the other cases.

To ensure nonobtusity of the triangles generated, we first observe that by insisting that each mesh vertex be chosen as the midpoint of its corresponding cube edge, we can eliminate obtuse angles in all but 5 cases: cases 5, 9, 11, 12, and 14. For these cases, we can eliminate the obtuse angles, all 120°, by appropriately modifying the local triangulations.

Refer to Figure 3 for our following description of the modification to the original Marching Cubes algorithm. In cases 5 and 12, a triangle with a 120° angle, with its apex along the cube edge (0,1), and its neighboring triangle through the common green edge, are replaced by 3 equilateral triangles. The newly created vertex, at the center of the cube, is the midpoint of the corresponding edge, where an original right triangle is split into two. In case 9, a new vertex at the cube center (but slightly displaced in Figure 3 for a better illustration) is created, which forms 6 equilateral triangles to replace triangles having 120° angles. In case 11, there is also a 120° angle. A new vertex at the cube center is created again and it is connected to 6 vertices around it, forming 4 equilateral triangles and 2 isosceles right triangles. Case 14 is similar, where a new vertex is also inserted at the cube center, resulting in 6 nonobtuse triangles.

To incorporate the extra cases designed to resolve possible ambiguities caused by the original Marching Cubes

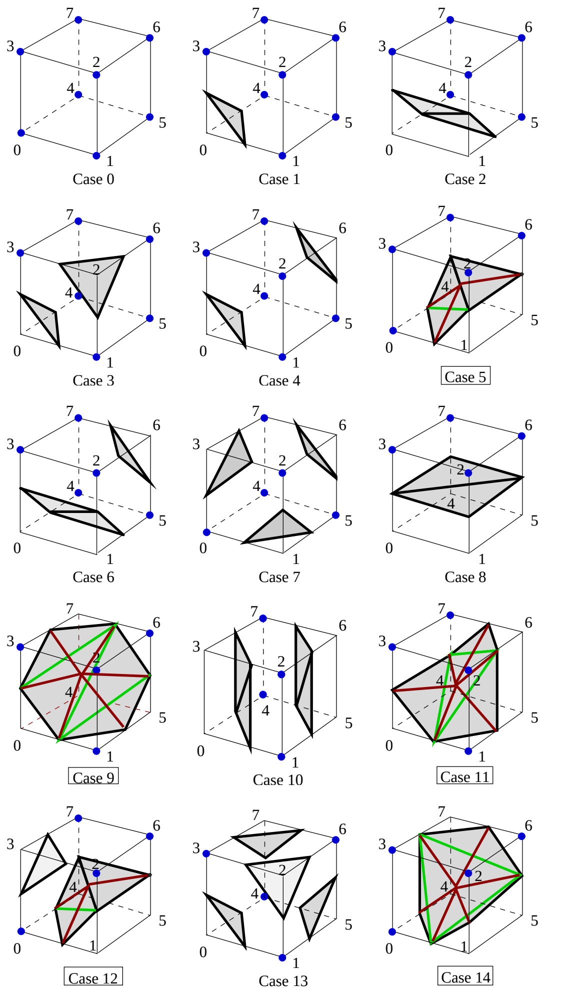

Figure 3: ***Modified Marching Cubes: regular cases.*** *We show the 15 regular cases given by the original Marching Cubes paper [LC87], where all the vertices are chosen as midpoints of the cube edges. However, in cases 5, 9, 11, 12, and 14, nonobtuse angles are still present. Simple modifications can be made to the local triangulations to rectify that. In the figures, dark green line segments represent edges from the original triangulations [LC87] and brown line segments replace them. Edges shared by the original and new triangulations are colored in dark black.*

algorithm, we follow the presentation of Lingrand et al. [LCGR02] for those extra cases. Six such cases are given, where the four shown in Figure 4 require appropriate modifications (which we describe in the next paragraph) to the local triangulations to eliminate obtuse angles. For clarity, the original triangulations are not shown. The remaining two cases [LCGR02] do not require any modification to the triangulations and are not shown either.

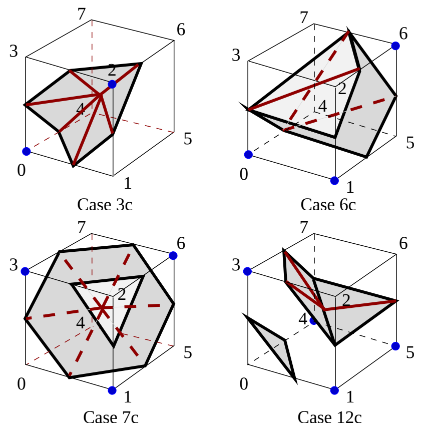

Figure 4: ***Modified Marching Cubes: extra cases.*** *New triangulations for four of the six cases from [LCGR02] that introduce obtuse angles. Brown segments are new and dark black segments come from the original triangulation.*

Refer to Figure 4, in cases 3c and 7c (the labeling of the cases follow that of Lingrand et al. [LCGR02]), the cube center is added, resulting in 6 equilaterals. In case 6c, no new vertices are added and the new triangulation has 2 right triangles and 3 isosceles triangles having side lengths of $\sqrt{6}/2$, $\sqrt{6}/2$, and $\sqrt{2}/2$, assuming unit cubes in the grid. Case 12c uses the same modification, as case 12 from Figure 3.

Note that for a closed input mesh, it is not hard to show that the triangle mesh produced using our modified Marching Cubes scheme does not contain degree-3 vertices. This is desirable for a triangle mesh as far as nonobtuse meshing is concerned. On the other hand, degree-4 vertices may exist but they are generally rare.

4.2. Construction of the scalar field

Given an input mesh M or point cloud S, one can construct a corresponding signed distance field, the 3D scalar field required by the Marching Cubes algorithm, in a variety of ways, e.g., see [HDD*92]. For the remeshing problem, given an input mesh M and an appropriately chosen cubical grid (note that the grid resolution can be used to bound the total number of triangles produced), we wish to estimate the signed distance from each grid point to the surface of M.

This is a classical and well-studied problem in geometric computing with fastest methods all utilizing some form of hierarchical spatial data structures [Gué01,BA05]. Recently, Bærentzen and Aanæs propose to rely on angle weighted pseudonormals to determine the signs of the distances computed robustly. For simplicity, we adopt the simple odd-even test for point-in-polyhedron test [Lin90], where the rays used coincide with the grid lines in our cubical grids. Note that we assume that the input mesh M is closed. If M has boundaries, in which case the notion of inside/outside is not well defined, we can first close up the mesh via dummy patches. After nonobtuse remeshing, triangles corresponding to the dummy patches may be appropriately removed.

5. Deform-to-fit via constrained optimization

An initial nonobtuse mesh $\hat{M}_0$ can be derived from the input mesh M via modified Marching Cubes, as just described. We iteratively deform $\hat{M}_0$ (let us denote by $\hat{M}$ the deformed mesh) so as to obtain progressively better approximation to M. This is accomplished through a heuristic search, within the space of nonobtuse meshes having the same connectivity as $\hat{M}_0$, for solving a global optimization problem.

The objective function is a point-wise least-square measure, combining approximation error and mesh smoothness quality. The nonobtuse constraints are linearized and convexified so that we only need to solve a series of small-scale convex quadratic programming problems. We now describe our approach in details.

5.1. Nonobtuse constraints

Nonobtuse region for a mesh edge: Given edge $e = (u, v)$, we define its *nonobtuse region* $\mathcal{N}(e)$ as the set of points w which would make the triangle $\triangle wuv$ nonobtuse. It is not hard to see that $\mathcal{N}(e)$ is the set difference between $sandw(e)$, the space "sandwiched" by the two planes U and V orthogonal to e and passing through vertices u and v, respectively, and $sphere(e)$, the sphere centered at the midpoint q of e and having a diameter of $|e|$, as shown in Figure 5(a).

Nonobtuse region for a vertex: Given vertex v and its one-ring vertices $v_0, v_1, \ldots, v_{k-1}$ in order, we define the nonobtuse region $\mathcal{N}(v)$ of v to be the set of points w that would make the triangles $\triangle wv_iv_{i+1}$ nonobtuse, $\forall i = 0, \ldots, k-1$ (subscript modulo k). Clearly, $\mathcal{N}(v) = \bigcap_{i=0}^{k-1} \mathcal{N}[(v_i, v_{i+1})]$ and in general, $\mathcal{N}(v)$ is both nonlinear and nonconvex, as are the nonobtuse regions for its corresponding edges.

Linearization, convexification, and feasible regions: To make the constrained optimization problem easier to solve, one can conservatively linearize and convexify the search space, which should then become a subset of $\mathcal{N}(v)$. A simple way to linearize the search space is to replace the sphere in Figure 5(a) by a cube that is tangent to the sphere, as shown in Figure 5(b), but the resulting search space is still nonconvex. In our implementation, we have chosen to be even more conservative. Specifically, consider a vertex v with its one-ring vertices $v_0, v_1, \ldots, v_{k-1}$ in order. Pick a one-ring edge $e = (v_i, v_{i+1})$. Clearly, v must lie on or outside $sphere(e)$, the sphere centered at the midpoint q of e and having a diameter of $|e|$. Let p be the intersection between the line segment $\overline{vq}$ and $sphere(e)$. Denote by R the plane through p and tangent to $sphere(e)$; refer to Figure 5(c). We define the *feasible region* $\mathcal{F}_v(e)$ of e with respect to v as the intersection between $sandw(e)$ and the half space defined by R that contains v. The feasible region for vertex v is given by $\mathcal{F}(v) = \bigcap_{i=0}^{k-1} \mathcal{F}_v[(v_i, v_{i+1})]$, which is linear and convex. Note that $\mathcal{F}(v) \subseteq \mathcal{N}(v)$ and in general $\mathcal{F}(v) \neq \mathcal{N}(v)$. Also, by construction, $\mathcal{F}(v)$ is always nonempty, since $v \in \mathcal{F}(v)$.

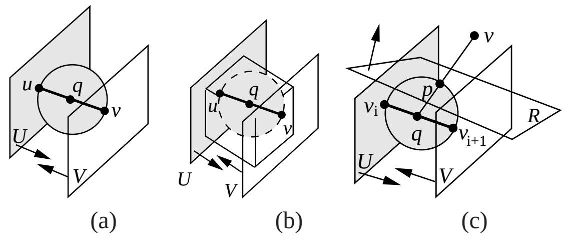

Figure 5: ***Nonobtuse and feasible regions.*** *(a) Nonobtuse region for edge (u, v). (b) Linearized nonobtuse region for edge (u, v); the sphere is replaced by a cube. (c) Feasible region for edge (v_i, v_{i+1}), with respect to the center vertex v, is a convex region delimited by planes U, V, and R.*

5.2. The global optimization problem

The global optimization problem for nonobtuse remeshing seeks to find a mesh $\hat{M}$, within the space $\mathcal{NO}(\hat{M}_0)$ of nonobtuse meshes having the same connectivity as the initial nonobtuse mesh $\hat{M}_0$, which minimizes a point-wise least-square measure. Formally, we wish to

$$\text{minimize} \sum_{v \in V(\hat{M})} \mathcal{L}(v, M), \quad \text{subject to} \quad \hat{M} \in \mathcal{NO}(\hat{M}_0), \tag{1}$$

where $V(\hat{M})$ is the set of vertices in $\hat{M}$ and

$$\mathcal{L}(v, M) = \alpha \cdot \mathcal{D}(v, M) + (1 - \alpha) \cdot \mathcal{S}(v), \tag{2}$$

where the point-to-surface distance $\mathcal{D}(v, M)$ represents a *quadric error* [GH97], $\mathcal{S}(v)$ is a smoothness (regularization) term, and $0 \le \alpha \le 1$ is a user-defined parameter that controls the trade-off between error reduction and smoothness.

- **Quadric error:** We associate a quadric Q_v with each vertex v in $\hat{M}$; Q_v is a 4×4 matrix capturing information about a set of planes associated with v and for any 3D point x in homogeneous coordinates, $Q_v(x) = x^T Q_v x$ is the sum of squared distances from x to the set of planes associated with v. In our case,

$$\mathcal{D}(v, M) = Q_v(v)/k \tag{3}$$

  gives an *average* squared distances from v to a set of k supporting planes corresponding to the triangle T of M that is closest to v, as well as to the r-ring neighbor triangles of T in M. Initially, we locate the triangle T efficiently utilizing the spatial data structure used for the construction of the scalar distance field. The parameter r is user-defined and in our experiments, we set $r = 1$.

- **Smoothness term:** At vertex v, the smoothness term
$$\mathcal{S}(v) = ||v - \mathcal{C}(v)||^2 \tag{4}$$
measures the squared distance between v and the centroid $\mathcal{C}(v)$ of its one-ring neighbors. Incorporating the smoothness term tends to produce smoother nonobtuse meshes with better angle quality, as we show in Section 7.

5.3. Optimization via heuristic deform-to-fit

Solution to the optimization problem (1) is likely intractable, we thus resort to heuristics. The first heuristic is applied to the iterative search toward a local minimum, relying on a priority queue. As mesh vertices are moved to better approximate the original mesh M, their associate quadrics are also updated heuristically, to speed up the optimization process.

Priority-queue based constrained search: During the search, each vertex v in the current mesh $\hat{M}$ may only move within its (nonempty, linear, and convex) feasible region $\mathcal{F}(v)$. Under this constraint, an optimal position v^* for the vertex v is one that minimizes $\mathcal{L}(v, M)$. To solve this convex quadratic programming problem, we rely on the OOQP solver of Gertz and Wright [GW03].

The priority $\mathcal{H}(v)$ for vertex v is given by the *improvement* made by moving v to v^*: $\mathcal{H}(v) = \mathcal{L}(v, M) - \mathcal{L}(v^*, M)$. We apply greedy search and at each iteration, a vertex having the highest priority is moved to its optimal position and all vertices influenced by the move will be re-inserted into the priority queue, since their priorities need to be updated due to change in their associated quadrics or feasible regions.

Updating quadrics: To compute the quadric for a vertex v that has just moved, we need to find a triangle T' in M that is closest to v. We can either utilize the hierarchical spatial data structure available to us from the construction of the scalar distance field or simply execute a local search for T', starting from the previous closest triangle T. The latter heuristic has worked quite well in practice, with a local search confined to within a 3-ring neighborhood of T.

Stopping: In our current implementation, we adopt a crude stopping criterion: the optimization stops when no vertex can be moved, within its feasible region, to reduce the objective function $\mathcal{L}$ further. In this case, $\mathcal{H}(v) \leq 0, \forall v \in V(\hat{M})$.

5.4. Constrained Laplacian smoothing

Experimentally, the nonobtuse meshes produced by the optimization procedure described so far are already quite satisfactory in terms of approximation error, but they can be slightly rough. This may be attributed to the fact that our optimization scheme is highly localized. To this end, we suggest to follow the mesh optimization step with a Laplacian smoothing step, which encourages *tangential* movement of the vertices to smooth out the roughness. Note that such movements may increase the approximation error. Thus we can interleave mesh optimization with constrained Laplacian smoothing, with the former being the first and the last steps of our "deform-to-fit" procedure. In a Laplacian smoothing step, we also process vertices one at a time, but without any particular order. Each vertex v processed is moved to its optimal location v^* within v's feasible region; the objective function is simply the smoothness term $\mathcal{S}(v)$.

6. Nonobtuse mesh decimation

Analogous to nonobtuse remeshing, the problem of nonobtuse mesh decimation can be formulated as a constrained local optimization problem as well and solved in an iterative greedy fashion. We iteratively perform edge collapses, starting from an initial full-resolution nonobtuse mesh, where the position of the unified vertex can be computed similarly as in the case of nonobtuse remeshing. As shown in Figure 6, the optimal position of the unified vertex w is computed within the feasible region $\mathcal{F}(u, v)$, determined similarly as in the vertex case by the neighboring vertices of the collapsed edge (u, v).

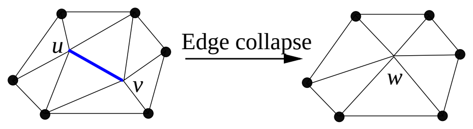

Figure 6: ***Nonobtuse edge collapse.*** *As we collapse edge (u, v) into a new vertex w, neighboring vertices of (u, v), marked by dark dots, form a one-ring which define the feasible region when computing an optimal position of w.*

We associate a quadric Q_w with each mesh vertex w during the decimation process. The set of planes associated with a vertex w, which define the quadric [GH97], in the full-resolution mesh is the set of supporting planes for the triangles incident to w. To perform greedy nonobtuse decimation, we place all candidate edges to collapse in a priority queue. For each candidate edge (u, v), we compute the optimal position w^* for the unified vertex w as follows,
$$w^* = \text{argmin}_{w \in \mathcal{F}(u,v)} [Q_u(w) + Q_v(w)]. \tag{5}$$
The cost for collapsing edge (u, v) is $Q_u(w^*) + Q_v(w^*)$ if $\mathcal{F}(u, v)$ is nonempty; otherwise, we set the cost to be $+\infty$. A candidate edge at the top of the priority queue, having the *lowest* cost, is collapsed first and the resulting new edges are re-inserted into the queue. When (u, v) is collapsed to w, we assign to the new vertex w the quadric $Q_w = Q_u + Q_v$.

Note that the greedy mesh decimation paradigm described above, based on priority queues, has been widely applied in level-of-details modeling [LRC*03]. Many techniques, e.g., lazy evaluations [CMO97], can be applicable to speed up this process, as well as our search procedure for nonobtuse remeshing. However, for simplicity, our current implementation has not utilized any such heuristics.

7. Experimental results

In this section, we use several experiments to demonstrate different characteristics of our approach and the quality of the nonobtuse meshes produced. Unless otherwise specified, in all of our experiments, we choose $\alpha = 0.5$ for mesh optimization. In most cases, only one step of constrained Laplacian smoothing is applied. The approximation error ε, which is the Hausdorff distance returned by the Metro tool [CRS98], is given as a percentage of the bounding box diagonal (BBD). The errors are measured against the original mesh for remeshing and against the full-resolution nonobtuse mesh in the case of decimation,

To start off, we show that by adjusting the resolution of the sampling grid used for modified Marching Cubes (Section 4), we can obtain high-quality nonobtuse remeshings of varying sizes; refer to Figure 7.

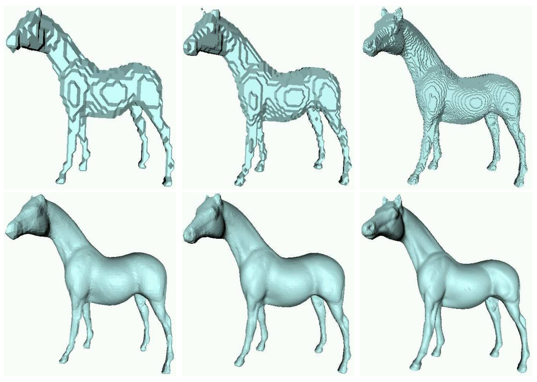

Figure 7: ***Nonobtuse remeshing of the horse model with varying sampling grids.*** *Top: results from modified Marching Cubes. Bottom: nonobtuse remeshings. (a)* $35 \times 55 \times 45$ *cubical grids; 9,966 triangles;* $\varepsilon = 2\%$. *(b)* $50 \times 80 \times 70$ *grids; 19,880 triangles;* $\varepsilon = 1.3\%$. *(c)* $110 \times 190 \times 160$ *grids; 125,046 triangles;* $\varepsilon = 0.5538\%$

In Figure 8, we break down the constrained optimization procedure we develop. The original “monster” model is shown in (a). For clearer illustration, we focus on a region, marked in (a), to show close-up shots in (b)-(f). Shown in (b) is the result obtained via mesh optimization only, with $\alpha = 1$. Without the smoothing term, poor angle quality is quite visible — $\gamma = 7.74\%$ of angles are less than 30°. In (c) and (d), we show results from mesh optimization only with $\alpha = 0.5$; angle quality is improved, shown in (c), where $\gamma = 0.12\%$. But slight geometric roughness still remains, as seen from (d). Finally, in (e) and (f), results from mesh optimization ($\alpha = 0.5$), combined with one step of Laplacian smoothing, are given. Angle quality is further improved — $\gamma = 0.03\%$, while the rough geometry has been smoothed out. These experiments show that both the smoothness term in the objective function and constrained Laplacian smoothing steps can be quite important in producing high-quality nonobtuse remeshing.

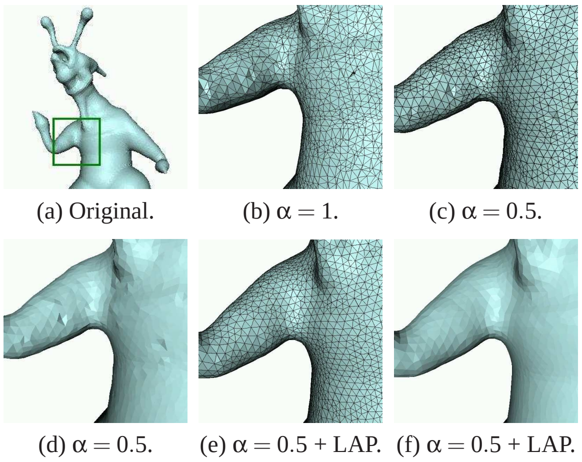

Figure 8: ***Importance of smoothness term and smoothing:*** *Without the smoothness term in the objective function, we suffer from poor angle quality, shown in (b). Percentage of small angles (angles less than* 30°*) is* $\gamma = 7.74\%$. *(c) and (d): After weighing in the smoothness term:* $\gamma = 0.12\%$. *But geometric roughness is quite visible. (e) and (f): With constrained Laplacian smoothing added:* $\gamma = 0.03\%$. *And the rough geometric features are smoothed out.*

In Table 1, we report mesh quality and algorithm performance statistics from nonobtuse remeshing of six models. Some of these six models that have not been displayed so far are shown in Figure 9. In Figure 10, some visual results for nonobtuse remeshing and mesh decimation, as well as the corresponding histogram plots of angle distributions, are given. As can be observed both via numerical results reported and visual examination, our algorithm is capable of producing high-quality nonobtuse remeshing, both in terms of approximation error and angle quality. The latter is due to the consideration for smoothing, as in general a nonobtuse mesh may have many small angles.

Besides small angles, it might also be desirable to reduce the number of angles close to 90°; many 90° angles are generated by the Marching Cubes step. Smoothing does help in reducing the number of 90° angles, as can be observed in Figure 10; see the middle column. But in our current implementation of nonobtuse decimation, no smoothing term is added to the objective function, nor are constrained Laplacian smoothing incorporated. As a result, we see spikes close to 90° bins. This can be fixed by considering smoothing for decimation in the same way as for remeshing.

In this paper, our focus has been on nonobtuse guarantees and mesh quality; speed has not been optimized. However, numerical results show that the number of vertex moves per optimization step is linear in terms of mesh size. Moreover, the number of optimization steps needed to produce a high-quality nonobtuse mesh is at most 3 in all of our experiments. We believe that, by adding speed-up techniques such as lazy evaluations or early stopping conditions, we can reduce the number of vertex moves significantly and converge more quickly to a good approximation of the initial mesh.

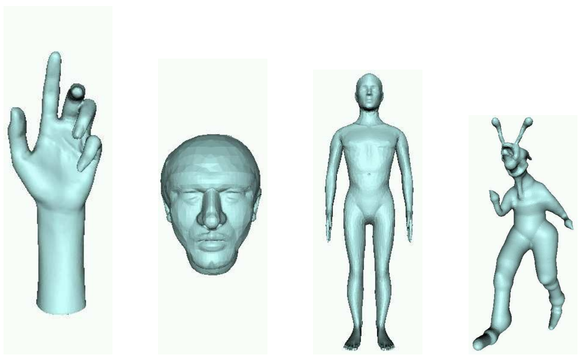

Figure 9: ***Four of the models tested in our experiments***

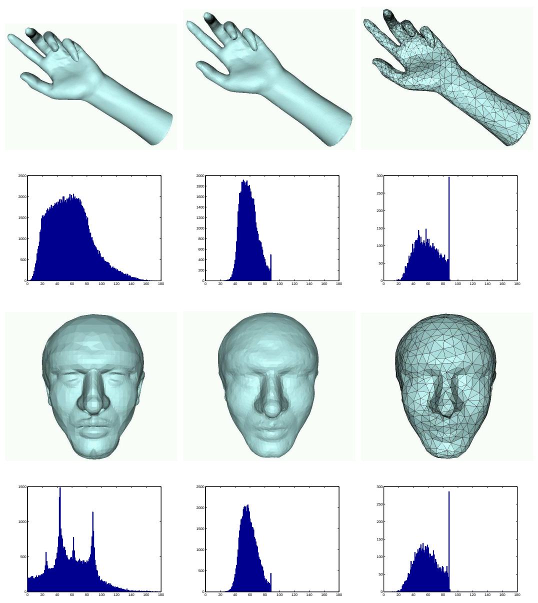

Figure 10: ***Visual results and histogram plots of angle distributions for nonobtuse remeshing and decimation.*** *From left to right: Original model; remeshed model; model after 90% of vertices decimated. The histogram plot associated with a particular model is given right below the model.*

8. Discussion and future work

In this paper, we propose a solution to an open problem — guaranteed nonobtuse remeshing. The nonobtuse mesh obtained can be further decimated while maintaining nonobtusity. The remeshing framework we develop, one starting with a modified Marching Cubes to transform an input mesh into one with guaranteed nonobtusity and then performing constrained optimization within a space of nonobtuse meshes, is quite general and flexible. We elaborate below.

- **Acute remeshing:** If Marching Cubes can produce an initial acute mesh (this is the challenging part if we insist on a guarantee), then we can constrain the search space to be a set of acute meshes and achieve acute remeshing, following the same line of approach as in the nonobtuse case. Enforcing the acute constraints is easy since we can simply move the constraining planes U, V, R, refer to Figure 5(c), of the feasible regions toward the center vertex v; this prevents the introduction of right angles. The same change also allows us to carry out acute mesh decimation.
- **Mesh generation from point cloud:** To obtain nonobtuse or acute meshes from a point cloud, the only additional requirement is a way to measure the approximation error against the underlying surface. One possible approach is to replace the planes currently used to construct the quadrics by a set of tangent planes associated with the point cloud; these tangent planes have been used in surface reconstruction by Hoppe et al. [HDD*92].
- **Handling meshes with boundaries:** Essentially, our approach can handle open meshes given a signed distance field for the input mesh is provided. However, ambiguity arises when one attempts to compute a signed distance field for any open mesh. In order to construct the initial nonobtuse mesh, we only require a scalar field that determines which points are inside the mesh. This can be easily computed by examining the intersections of the input mesh and the sample grid for the marching cubes. A point p is inside the mesh if $n \cdot (p - w) < 0$ where w is an intersection of a connected edge on the grid and a triangle of the input mesh having normal n. We can compute the scalar field by tranversing the faces of the input mesh in an advancing front manner.

In addition to acute remeshing, mesh generation, and handling boundary cases, we would also like to improve the efficiency of our constrained optimization. Lazy evaluations, early stopping, and more adaptive smoothing and error reduction procedures are all worth looking into. Tuning our approach to deal with adaptive grids for mesh generation is interesting but challenging. Quality-wise, we would like to improve the angle quality further, possibly providing a substantial lower-bound while still ensuring nonobtuse or acute guarantees; this is still an open problem. Finally, our current method is not designed to recover or enhance sharp features in the input models. It would be interesting to see whether treating the feature edges as hard or soft constraints would

Input mesh (#F)	**#F**	**#V**	**Min. ∠**	**% small ∠'s**	**Metro ε**	**1: %VM**	**2: %VM**	**3: %VM**
Armhand (50K)	18798	9401	26.14	0.05	1.00	310.6	397.9	487.3
Bigsmile (34.7K)	15380	7692	19.49	0.04	0.78	308.3	312.9	N/A
Baldhead (15.8K)	18704	9356	20.68	0.06	1.86	407.1	399.8	393.0
Horse (40K)	19880	9944	15.03	0.18	1.31	258.3	266.4	N/A
Man (29K)	68252	34122	10.52	0.14	0.55	328.7	285.1	N/A
Monster (32.5K)	48226	24115	19.79	0.03	0.64	294.8	374.3	N/A

Table 1: *This table shows quality measures and algorithm performance statistics for nonobtuse remeshing. We report: input mesh and its face count (#F); face (#F) and vertex (#V) counts of the output nonobtuse mesh; minimum angle (Min. ∠) in the nonobtuse meshes, as well as the percentage of angles less than* 30° *(% small ∠); approximation error (ε) from Metro [CRS98]. We also show the number of vertices moved in each mesh optimization step (1,2,and 3) of the constrained optimization, as a percentage ("1:%VM", "2:%VM", and "3:%VM") of the total number of vertices. These percentages give an indication of how quickly our greedy optimization procedure would converge. Note that in many cases, only two iterations are needed; in that case, the "3:%VM" column will be marked by "N/A".*

still ensure the general quality of the nonobtuse or acute meshes produced.

References

[ACSYD05] ALLIEZ P., COHEN-STEINER D., YVINEC M., DESBRUN M.: Variational tetrahedral meshing. *ACM Trans. on Graphics 24*, 3 (2005), 617–625.

[AFSR03] ATTENE M., FALCIDIENO B., SPAGNUOLO M., ROSSIGNAC J.: Swingwrapper: Retiling triangle meshes for better edgebreaker compression. *ACM Trans. on Graphics 22*, 4 (2003).

[AUGA05] ALLIEZ P., UCELLI G., GOTSMAN C., ATTENE M.: Recent advances in remeshing of surfaces, 2005. Part of the state-of-the-art report of the AIM@SHAPE EU network.

[BA05] BÆRENTZEN J. A., AANAES H.: Signed distance computation using the angle weighted pseudonormal. *IEEE Trans. on Visualization and Computer Graphics 11*, 3 (2005), 243–253.

[BCER95] BERN M., CHEW L. P., EPPSTEIN D., RUPPERT J.: Dihedral bounds for mesh generation in high dimensions. In *ACM-SIAM Symp. on Discrete Algorithms* (1995), pp. 89–196.

[BE92] BERN M., EPPSTEIN D.: Mesh generation and optimal triangulation. In *Computing in Euclidean Geometry, Lecture Notes Series on Computing* (1992), vol. 1.

[BMR94] BERN M., MITCHELL S., RUPPERT J.: Linear-size nonobtuse triangulation of polygons. In *ACM Symp. on Computational Geometry* (1994), pp. 221–230.

[Che93] CHEW P.: Guaranteed-quality mesh generation for curved surfaces. In *ACM Symp. on Computational Geometry* (1993), pp. 274–280.

[CL80] CASSIDY C., LORD G.: A square acutely triangulated. *J. Rec. Math. 13*, 4 (1980), 263–268.

[CMO97] COHEN J., MANOCHA D., OLANO M.: Simplifying polygonal models using successive mappings. In *IEEE Visualization* (1997), pp. 395–402.

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174.

[CS05] CHENG H.-L., SHI X.: Quality mesh generation for molecular skin surfaces using restricted union of balls. In *IEEE Visualization* (2005), pp. 399–405.

[EDD*95] ECK M., DEROSE T., DUCHAMP T., HOPPE H., LOUNSBERY M., STUETZLE W.: Multiresolution analysis of arbitrary meshes. In *SIGGRAPH '95* (1995), pp. 173–182.

[Epp97] EPPSTEIN D.: Acute square triangulation, 1997. The Geometry Junkyard: `http://www.ics.uci.edu/~eppstein/junkyard/acute-square/`.

[ESU04] EPPSTEIN D., SULLIVAN J. M., ÜNGÖR A.: Tiling space and slabs with acute tetrahedra. *Computational Geometry: Theory and Applications 27*, 3 (2004), 237–255.

[Flo98] FLOATER M.: Parametric tiling and scattered data approximation. *Int. J. on Shape Modeling 4* (1998), 165–182.

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *SIGGRAPH '97* (1997), pp. 209–216.

[Gué01] GUÉZIEC A.: Meshsweeper: Dynamic point-to-polygonal mesh distance and applications. *IEEE Trans. on Visualization and Computer Graphics 7*, 1 (2001), 47–60.

[GW03] GERTZ E. M., WRIGHT S. J.: Object-oriented software for quadratic programming. *ACM Trans. on Math. Software 29* (2003), 58–81. Software: `http://www.cs.wisc.edu/~swright/ooqp/`.

[GY03] GU X., YAU S.-T.: Global conformal surface parameterization. In *Proc. of Symp. on Geometry processing* (2003), pp. 127–137.

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. In *SIGGRAPH '92* (1992), pp. 71–78.

[KS98] KIMMEL R., SETHIAN J. A.: Computing geodesic paths on manifolds. *Proc. of National Academy of Science USA 95* (1998), 8431–8435.

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87* (1987), pp. 163–169.

[LCGR02] LINGRAND D., CHARNOZ A., GERVAISE R., RICHARD K.: The marching cubes, 2002. `http://www.essi.fr/~lingrand/MarchingCubes/algo.html`.

[Lin90] LINHARD J.: A quick point-in-polyhedron test. *Computer & Graphics 14*, 3 (1990), 445–448.

[LRC*03] LUEBKE D., REDDY M., COHEN J., VARSHNEY A., WATSON B., HUEBNER R.: *Level of Detail for 3D Graphics*. Morgan Kaufmann, 2003.

[NH91] NIELSON G. M., HAMANN B.: The asymptotic decider: Resolving the ambiguity in marching cubes. In *IEEE Visualization* (1991), pp. 83–91.

[US02] ÜNGÖR A., SHEFFER A.: Pitching tents in space-time: Mesh generation for discontinuous galerkin method. *Int. J. of Foundations of Computer Science 13*, 2 (2002), 201–221.