EUROGRAPHICS 2011 / M. Chen and O. Deussen
(Guest Editors)

Volume 30 (*2011*), *Number 2*

Walking On Broken Mesh: Defect-Tolerant Geodesic Distances and Parameterizations

Marcel Campen and Leif Kobbelt

Computer Graphics Group, RWTH Aachen University, Germany

Abstract

Efficient methods to compute intrinsic distances and geodesic paths have been presented for various types of surface representations, most importantly polygon meshes. These meshes are usually assumed to be well-structured and manifold. In practice, however, they often contain defects like holes, gaps, degeneracies, non-manifold configurations – or they might even be just a soup of polygons. The task of repairing these defects is computationally complex and in many cases exhibits various ambiguities demanding tedious manual efforts. We present a computational framework that enables the computation of meaningful approximate intrinsic distances and geodesic paths on raw meshes in a way which is tolerant to such defects. Holes and gaps are bridged up to a user-specified tolerance threshold such that distances can be computed plausibly even across multiple connected components of inconsistent meshes. Further, we show ways to locally parameterize a surface based on geodesic distance fields, easily facilitating the application of textures and decals on raw meshes. We do all this without explicitly repairing the input, thereby avoiding the costly additional efforts. In order to enable broad applicability we provide details on two implementation variants, one optimized for performance, the other optimized for memory efficiency. Using the presented framework many applications can readily be extended to deal with imperfect meshes. Since we abstract from the input applicability is not even limited to meshes, other representations can be handled as well.

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

1. Introduction

Intrinsic distances and geodesic paths between points on a 2-manifold in 3-space are a fundamental ingredient in numerous Computer Graphics and Geometry Processing applications like isometry-invariant shape matching, intrinsic symmetry and regularity detection, surface parameterization, texturing, and tool-path generation, to name a few. A number of efficient methods to compute these have been presented – for implicit surfaces, point set representations, and most importantly polygon meshes. These meshes are usually assumed to be well-structured and manifold with complete connectivity information. Unfortunately this requirement is not always met in practice: real-world meshes often exhibit several kinds of defects depending on their origin – holes, gaps, (near-)degenerate polygons, non-manifold configurations with singular edges and vertices, or they might even be just a soup of polygons, completely lacking any connectivity information.

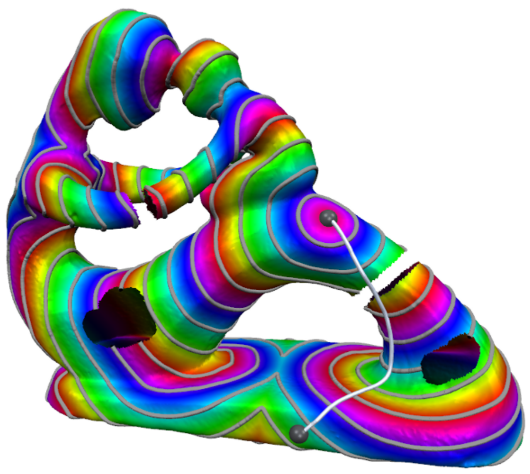


Figure 1: *An intrinsic distance field and a geodesic path computed on an imperfect mesh in a defect-tolerant way.*

A considerable number of methods that aim at repairing mesh defects have been presented (cf. the recent survey by Ju [Ju09]). They all suffer from the fact that the general repair problem is naturally ill-posed and often exhibits various geometrical and topological ambiguities if no additional prior knowledge is available. This complicates the process and leads to the fact that often user interaction and tedious manual effort is required to obtain a clean mesh in the end. Furthermore, if the application at hand does not actually require the mesh to be repaired anyway, spending these efforts solely in order to facilitate the requisite geodesic distance computations seems to be immoderate in many cases.

We present a computational framework that allows for the computation of meaningful approximate intrinsic distances and geodesic paths on meshes with all kinds of defects in a way tolerant to these defects (cf. Figure 1). Further, we show ways to locally parameterize a surface based on geodesic distance fields, easily facilitating the application of textures and decals on meshes in a defect-oblivious manner. We do all this without explicitly repairing the mesh, thereby avoiding the costly additional efforts as well as the resolution of problem-inherent ambiguities. In the case of severe mesh defects, e.g. large missing parts, the computed distance fields might of course be inconsistent with those of the object that is actually *meant* to be represented by the partial data – in particular we do not propose new disambiguation or "defect hole" – "feature hole" distinction strategies.

The basic idea is to abstract from the mesh structure (and all its potential defects) and to perform all computations discretely in a crust volume tightly restricted to the spatial regions occupied by elements of the input. It has been proven that the extrinsic distance field in such crust volumes converges uniformly to the intrinsic distance field of the surface they bound with increasing tightness [MS01]. We show ways to perform the necessary computations in a memory-efficient manner such that tightness can be achieved by using high resolutions. The discrete structure readily allows for the application of topology-sensitive morphological operations [BK05] to make computations tolerant to gaps and holes. An improved variant of the Fast Marching method [Set95] is then applied to efficiently generate distance fields. Due to the abstraction from the input, applicability is not limited to polygon meshes; other representations like point sets, implicit functions, or NURBS patches can be handled as well.

2. Related Work

2.1. Geodesic Distance Computation

Intrinsic distance fields and geodesic paths on surfaces are well-studied objects of Differential Geometry [dC76]. Their computation on polygon meshes, as well as on other surface representations like implicit functions, voxel or point sets, has been made feasible by the work of several researchers.

In 1987 Mitchell et al. [MMP87] presented the foundation for the exact computation of intrinsic distance fields on triangulated manifolds. Later practical implementations [SSK*05] and extensions [BK07] of this work have been provided. Another exact method has been presented by Chen and Han [CH90, XW09]. The Fast Marching (FM) method [Set95] has been extended to triangulated manifolds [KS98, NK02] and allows for approximate computations. Shortest geodesic paths can be obtained by gradient descent methods. Polthier and Schmies [PS06] presented another notion and show how to trace out straightest geodesics.

Kiryati and Székely [KS93] estimate intrinsic distances in structures represented by regular discrete volumetric data by graph distances on the voxel neighborhood graph. Later the FM method has been applied in this field to obtain improved, consistent results [DC00].

For point-sampled surfaces an estimation of intrinsic distances can be computed by graph distances in a point proximity graph [RDSK06, BSLT00]. For more accurate results the computation of intrinsic distances on implicit surfaces and point sets has been transformed into the discrete volumetric setting by Mémoli and Sapiro [MS01, MS05]. The FM method is then applied to the voxel representation. They provide an in-depth analysis of the quality of the approximation that is achieved. This work is probably closest in spirit to ours in that we also transform the problem into a kind of voxel setting, however, our motivation and further strategy is entirely different as our goals are defect-tolerant computations and parameterizations on meshes.

2.2. Local Surface Parameterization

Intrinsic distance computations have been used to build inverse exponential maps [dC76] to locally parameterize surfaces in numerous applications. Such approaches are constructive, do not require costly optimization procedures while still providing properties usually desired, like low stretch and angular distortion, in a local neighborhood on the surface. This has, e.g., been exploited for shape comparison [ZG04, GGGZ05], the plausible application of decals [SGW06, BKW10], and surface detail transfer [BMBZ02].

2.3. Further Applications

The fact that isometric deformations preserve geodesic distances has been exploited to perform bending invariant shape comparison of meshes [EK03] and point clouds [MS04] using Multidimensional Scaling (MDS) [CC94]. But also different approaches to shape matching and comparison make use of geodesic distances [BBK06, SCF10]. Other tasks like intrinsic symmetry detection [RBBK07], regularity detection [MBB10], and parameterization [GKK02] have shown to benefit from MDS variants that consider geodesic distances. Remeshing [PC03] and surface sampling [MD03] are further examples of applications that by exploiting geodesic distances achieve high-quality results. In the field of texturing geodesic distances have also been used to optimize mappings for casual projective texturing [TT09].

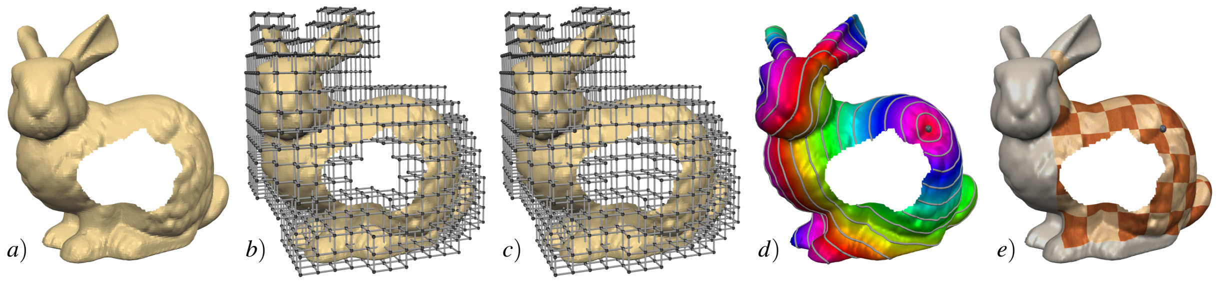


Figure 2: *Overview of our approach: a) Input mesh with defect (large artificial hole for demonstration), rendered with backface culling. b) Initial cubical complex constructed for this mesh (at a low resolution of 16^3 for illustration). c) Complex after applying topology-sensitive morphological operators; the hole is now bridged. d) Visualization of a geodesic distance field (with isolines) emanating from a point source, computed on the complex (at a resolution of 64^3), and mapped to the input mesh by interpolation. e) Application example: defect-tolerant decal textured onto the surface using a local geodesic parameterization.*

3. Overview

The framework we present takes inconsistent raw polygon meshes (cf. Figure 2 a) as input and allows for the computation of (1) intrinsic distance fields with point or polygonal sources, (2) geodesic paths between surface points, and (3) various types of local surface parameterizations. Independent of which computations are to be performed for a specific application, the first step is to abstract from the mesh representation to a cubical complex, i.e. a Cartesian grid tightly restricted to the spatial regions occupied by the mesh elements (cf. Figure 2 b). To make further computations tolerant to gaps and holes we next apply topology-sensitive morphological operations [BK05] to this complex. This closes all gaps and holes of sizes up to a user-specified tolerance threshold and yields the final complex (cf. Figure 2 c). Details on this construction are presented in Section 4.

Source points or curves for the computation of distance fields and geodesic curves are mapped into the complex (cf. Section 5.1) to obtain initial conditions for the subsequent Fast Marching (cf. Section 5.2). Finally the mapping of the results back onto the input mesh is detailed in Section 5.4 (cf. Figure 2 d). In Section 6 we present several ways to construct local surface parameterizations in defect-tolerant ways using the introduced framework and show application to texturing of arbitrary meshes (cf. Figure 2 e). Further results and analyses are provided in Section 8.

4. Mesh Abstraction

Given an input mesh $\mathcal{M} = (F, E, V)$ consisting of sets F, E, V of faces, edges, and vertices respectively. We do not want to make any assumptions about the integrity of $\mathcal{M}$, i.e. it may contain holes, gaps, singularities, degeneracies, or missing connectivity information. Hence, we first abstract from this mesh to a cubical complex representation. The idea of using some kind of discrete abstraction of polygonal geometry has been applied for the same reason in various fields, from simplification [ABA02] over vectorization [MZL*09] to model repair [BPK05], to name only a few.

4.1. Initial Complex Construction

The cubical complex we use is essentially a cut-out of a three-dimensional Cartesian grid such that all elements of F, E, and V are contained in the union of its cells. It should be minimal, i.e. restricted to the regions occupied by the mesh elements as tightly as possible. Hence, the construction of this initial minimal complex $\mathcal{C}_\mathcal{I}$ basically corresponds to the three-dimensional rasterization (or "voxelization") of the mesh elements, since the obtained voxels directly correspond to the 3-cells of the desired complex $\mathcal{C}_\mathcal{I}$. Details on efficient implementation are postponed to Section 7. We base the further description on the cubical complex notation since also the 0-cells and 1-cells of the complex are involved in the computations. In the following we will refer to the 0-cells of the complex as *nodes*, to the 1-cells as *arcs*, to the 2-cells as *walls*, and to the 3-cells simply as *cells*. Further, by $N(c)$, c being a cell of $\mathcal{C}$, we denote the set of nodes incident to c, and $N(C) = \bigcup_{c \in C} N(c)$ for a set C of cells.

Some computations can benefit from normal information at the nodes. For each node we average the normal vectors of the faces intersecting the eight incident cells to obtain an estimated normal vector. Since this is only meaningful if the set of normal vectors is coherent at least to some extent, we do only store such an estimated normal vector at nodes where the cone spanned by the set of normal vectors has a non-reflex opening angle – which can conservatively be estimated by all pairs of these vectors spanning angles smaller than $2/3\,\pi$ for simplicity. For the estimation process the normal vectors of the input have to be oriented consistently. If this is not the case it can for instance be enforced by the method of Borodin et al. [BZK04].

4.2. Morphological Operations

Bischoff et al. [BK05, BPK05] successfully applied morphological operations to voxel representations in the context of 3D model repair. By applying the discrete morphological dilation operation to our initial complex $\mathcal{C}_\mathcal{I}$ we can easily close holes and bridge gaps up to a user-specified width. In principle holes of any size can be bridged in this way, however, note that intentional feature holes, constrictions, or tunnels up to the chosen size are also closed. If these can safely be distinguished from defect holes in a specific scenario, dilation can of course be restricted to defect hole boundaries, e.g. as done in [BPK05], to alleviate this behavior.

Performing distance computations on the resulting dilated complex $\mathcal{C}_\mathcal{D}$ would, however, result in significantly lowered accuracy [MS01]. Hence we apply a morphological erosion operation on $\mathcal{C}_\mathcal{D}$ to obtain the final complex $\mathcal{C}$. To prevent this operation from tearing closed holes and gaps open again, a topology-preserving variant [BK05] is employed. This operation removes cubes only if this does not change the digital topology (defined via wall-adjacency, or *6-neighborhood*) of the complex thereby leaving a minimal sheet of cubes in holes and gaps. The entire process is illustrated in Figure 3, implementation details are given in Section 7.

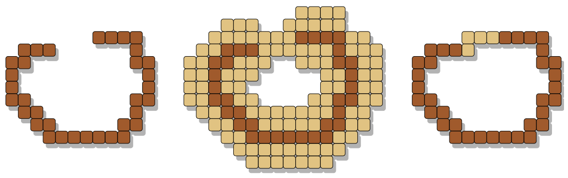


Figure 3: *2D schematic example of the employed morphological operators* dilation *(middle) and* topology-preserving erosion *(right), filling holes up to a specified size.*

5. Geodesic Computations

Having obtained the computational domain in form of a cubical complex $\mathcal{C}$ as described in the last section, we can now perform approximate geodesic distance computations by Fast Marching (FM) [Set95, DC00]. This method, applied to the complex, will compute a distance field $d : N(\mathcal{C}) \to \mathbb{R}^+$.

5.1. Initialization

A distance field d usually emanates from (a set of) source points or curves on the input surface, i.e. on $\mathcal{M}$ (but sources in free space can be handled as well). To set initial values for the front propagation of the FM method this information needs to be transferred into $\mathcal{C}$.

For the set S of all specified point and curve sources of d we determine the set C_S of cells of $\mathcal{C}$ they intersect and initialize the distance values $d(n)$ for all $n \in N(C_S)$. We set $d(n) = \min_{s \in S} dist(n,s)$. If an averaged normal vector is not available we choose $dist(\cdot,\cdot)$ to be the Euclidean distance. Otherwise we can enhance accuracy by calculating $dist(n,s)$ as the Euclidean distance between n and the orthogonal projection of s onto the tangent plane T_n at n. This suppresses the surface-orthogonal distance component merely introduced by the nodes not lying directly on the surface. The distance value of all other nodes is yet undefined, i.e. we initially set $d(n) = \infty$ for all $n \notin N(C_S)$.

5.2. Fast Marching

Starting from the initialized nodes $N(C_S)$ the FM method [Set95, DC00] can now be applied to perform a front propagation over all other nodes of $\mathcal{C}$ in order to determine distance values $d(n)$ for all nodes n that closely approximate their defect-tolerant geodesic distance to the set S of sources. The FM method keeps the nodes that are part of the current front in a priority queue, sorted by $d(\cdot)$, and always removes the node with lowest distance while updating the distance values of its adjacent nodes and adding them (back) to the queue. Since this propagation is done in upwind direction, distance values of cells removed from the queue can justly be considered final.

In the FM front propagation process a distance value update for a node is computed from the distance values at up to three adjacent nodes by a form of extrapolation. Originally, a gradient-based first-order update rule was proposed. Higher-order [Set99, HF07] rules can be employed to increase accuracy. Especially for the case of circular distance fields of point sources the first-order rules overestimate distances as pointed out by Novotni and Klein [NK02]. They show that higher accuracy is achieved by rules that take the particular circular nature of the front into account. We next extend these rules to our three-dimensional setting. Their use is to be preferred over gradient-based rules for point sources.

On a triangle mesh, given two vertices of a triangle with known intrinsic distances to the source, [NK02] determine a virtual point source in the plane of the triangle that maintains these distances. The distance value of the third vertex can then be updated to its Euclidean distance to this virtual source. In our three-dimensional domain, to update the distance value $d(n)$ of a node n from three adjacent nodes n_0, n_1, n_2 with already computed distance values, we determine a virtual point source s in space by trilateration: the points s_1 and s_2 are found as the two points satisfying the three sphere equations $d(n_i)^2 = ||x - n_i||^2$, $0 \leq i \leq 2$. Since propagation proceeds in upwind direction, the one with larger distance has to be chosen, i.e. we apply the update $d(n) = \max(||n - s_1||, ||n - s_2||)$. When less than three adjacent nodes have distances available we fall back to lower dimensional lateration. Figure 4 illustrates the effect of using these *spherical* rules instead of gradient based update rules.

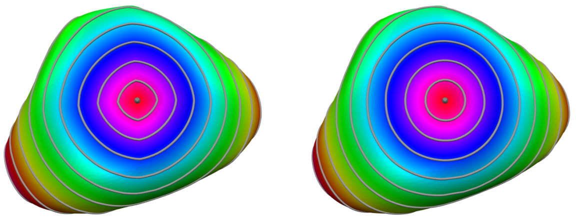


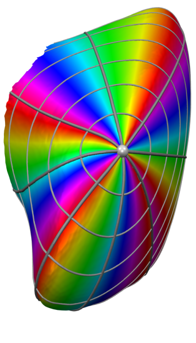
Figure 4: *Comparison of gradient-based (left) and spherical (right) update rules. As can be seen from the distance field isolines gradient-based rules overestimate distances in diagonal directions. Spherical rules show isotropic behavior.*

5.3. Polar Angle Propagation

Additionally to computing a distance field d of a point source an angular coordinate field θ can be constructed to obtain a (local) polar surface parameterization (d,θ). Schmidt et al. [SGW06] construct both fields on meshes approximately by a modified version of Dijkstra's shortest path algorithm, essentially unwrapping the surface into the tangent plane at the source. The accuracy of the radial coordinate computed in this way, however, usually cannot compete with that of the FM method. We incorporate the general idea into the FM method to simultaneously construct angular coordinates θ. For this construction normal vector information must be available (cf. Section 4.1). Nodes that lack normal information simply inherit the normals from their predecessors in the front propagation process. This proved to be sufficient in our experiments and compared to more sophisticated global normal diffusion methods does not hinder a sweeping implementation (cf. Section 7.2.2).

Let T_s be the tangent plane of the surface at the source point s and a the polar axis in that plane, defining the orientation of angle 0. Furthermore let $\mathrm{uv}(d,\theta) = (d\cos\theta, d\sin\theta)$ denote the 2D Cartesian vector defined by d and θ in T_s and $\mathrm{angle}((u,v)) = \arctan(v/u)$ the angular coordinate of the vector (u,v) in the polar system defined by T_s and a.

During an update step of the FM method, updating node n based on the distance values of one, two, or three adjacent nodes $n_0,\dots,n_m$ as described in the last section, $\theta(n)$ is set as follows: we first choose one of the updating nodes n_i and take its so-called inverse exponential map vector $\exp_s^{-1}(n_i) = \mathrm{uv}(d(n_i),\theta(n_i))$. The ideal choice is $n_{min} = \mathrm{argmin}_{i\in\{0,\dots,m\}} d(n_i)$ since this can be expected to have lowest accumulated error in its d and θ values. Then we "unwrap" the vector $n - n_{min}$ into the tangent plane T_s by first projecting it orthogonally onto $T_{n_{min}}$ (to get rid of the orthogonal component merely introduced by n not lying directly on the surface), then rotating it into T_s around the axis orthogonal to the normals at s and n_{min} (*hinge map*), and finally transforming it into a 2D vector in the coordinate system of T_s. By adding these two vectors we obtain an approximation for the angular component of $\exp_s^{-1}(n)$, i.e. we set $\theta(n) = \mathrm{angle}(\exp_s^{-1}(n_{min}) + TRP(n - n_{min}))$, where T, R, and P are the transformation, rotation, and projection operations. $d(n)$ is computed by the FM update rules as before for accuracy. The adjacent figure shows a visualization of such an angular field on a curved surface, including isolines in radial and axial directions.



5.4. Interpolation

After distance values and possibly angular values have been computed at the nodes of $\mathcal{C}$ we want to transfer this information back onto the input mesh $\mathcal{M}$. Let P be the set of points on $\mathcal{M}$ at which these values shall be made available (in most applications this is simply the set V of vertices). While it can be expected that most accurate results are achieved by integrating the points of P into $\mathcal{C}$ as virtual nodes to compute the values by the described FM update rules, this proved to result in slight discontinuities between points that are nearby but fall into different cells. By contrast, the application of trilinear interpolation leads to smooth results due to its very nature. Hence, we interpolate the values at a point p of P from the eight nodes incident to the cell of $\mathcal{C}$ that includes p.

In case a Cartesian (u,v) parameterization is to be constructed from a computed polar parameterization, it is beneficial to perform the transformation already at the nodes and then interpolate the (u,v) coordinates. This avoids special case handling near the singularity at the field's pole where trilinear interpolation is unsuited for angles and distances.

5.5. Geodesic Paths

Shortest geodesic paths between two points p and q on the surface can be constructed using a gradient descent procedure. First the distance field for source p is computed. Then, starting from q, a piecewise linear path through $\mathcal{C}$ can be constructed by proceeding stepwise in direction of the negative gradient of this field. The constructed path lies in $\mathcal{C}$, i.e. in the surrounding space of $\mathcal{M}$; if a path *on* $\mathcal{M}$ is desired, we perform a projection where possible – in hole regions the path simply remains in the cell sheet that survived erosion.

6. Parameterization

As pointed out in Section 2.2 geodesic computations have proven to be a valuable tool to constructively generate local surface parameterizations. We now present various methods to construct such parameterizations – even on inconsistent raw meshes. Figure 5 shows examples for comparison.

Center Point Parameterization A parameterization around a point can be constructed by computing a radial and angular coordinate field for this source. The resulting polar parameterization can be transformed into a Cartesian (u,v) parameterization with a user-specified orientation and scale, e.g. for applying decals onto curved surfaces in an intuitive way [SGW06]. Due to the defect-tolerance of our approach such decal application can be performed across non-connected mesh components, bridging gaps and holes, thus on a much broader range of meshes. The distortion – necessarily introduced on surfaces with non-zero Gaussian curvature – is minimal in the center and typically grows with increasing distance depending on the curvature distribution.

Boundary Curve Parameterization More flexibility and a less center-biased distortion distribution is achieved by specifying four curves u_0, v_0, u_1, v_1 forming a quadrilateral on the surface and for each computing the distance field emanating from it. The distance fields d_{u_0} and d_{u_1}

of opposite curves u_0, u_1 can then be blended into d_u by $d_u = d_{u_0}/(d_{u_0}+d_{u_1})$, analogously for d_v, and these two resulting fields d_u and d_v be taken as u and v coordinates of a parameterization that is aligned with the surface quadrilateral, mapping this region to the domain $[0,1]^2 \subset \mathbb{R}^2$.

Corner Point Parameterization Specifying desired parameterization boundary curves on a surface can be tedious. Furthermore, in order to reduce distortions in the parameterization generated as described in the last section these curves should be geodesic paths between their endpoints. We can simplify the specification task to choosing the four corner points of the desired quadrilateral and then automatically determine geodesic paths between the points (cf. Section 5.5) to construct a geodesic quadrilateral as basis for the boundary curve parameterization.

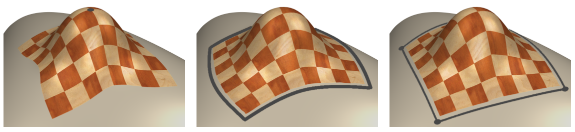


Figure 5: *Parameterization examples: center point parameterization (left), boundary curve parameterization from user-specified quadrilateral (middle), and corner point parameterization from automatically determined geodesics (right).*

7. Implementation Details

We now provide some details on the efficient implementation of the computations described in the last sections. In order to enable broad applicability an implementation variant optimized for performance as well as one optimized for memory efficiency is presented.

7.1. Cubical Complex Construction

To allow for high resolutions without excessive memory requirements we employ an octree $\mathcal{O}$ that is adaptive in multiple ways. This further allows for the efficient establishment of correspondences between elements of $\mathcal{M}$ and $\mathcal{C}$, required to transfer information between the two representations.

We start by defining the cubical root cell of $\mathcal{O}$ to include the bounding box of $\mathcal{M}$. The elements of $\mathcal{M}$ are then "inserted" into $\mathcal{O}$, intersected cells are refined up to a user-specified maximum level l and marked as *solid*. The set of these solid cells then forms the voxel representation of $\mathcal{M}$. The vertices of V (or the set P of points for which distance field values shall be computed, cf. Section 5.4) can optionally be recorded in the containing leaf cells, allowing for direct access during interpolation. Methods for the efficient traversal of octrees have been presented [FP02, Sam89], and by installing so-called *ropes* to explicitly link neighboring cells [MB90], the computational cost of cell navigation can be reduced, trading memory requirements for efficiency.

Efficient dilation operators for octrees with on-demand refinement of cells to level l are presented by Bischoff et al. [BPK05]. Let λ denote the width of a cell on level l. Given that holes and gaps up to a width of ρ shall be considered unwanted, we need to determine how many layers of cells need to be added to close these. Due to the discrete setting the dilation process has a directional bias. Slowest growth happens in space-diagonal direction where γ dilation steps bridge gaps of widths up to $2\gamma\lambda/\sqrt{3}$. Hence, to ensure closing of all holes of widths up to ρ we choose $\gamma = \rho/(2\lambda/\sqrt{3})$.

To now remove all dilated cells except for thin hole and gap bridging sheets we perform a topology-preserving erosion [BK05]. In contrast to the original description we do not only apply γ erosion steps but keep eroding until no more dilated cells can be removed without changing the digital topology of the voxel set specified by solid and dilated cells. This eliminates the "*closing*" character of the operations. Afterwards the set of solid and remaining dilated cells corresponds to the desired complex $\mathcal{C}$.

Instead of extracting the desired cubical complex $\mathcal{C}$ from $\mathcal{O}$ to represent it by a separate data structure we directly represent it by the octree. Unfortunately the FM front propagation operates on the graph of nodes and arcs of $\mathcal{C}$ – which are not explicitly represented in the octree data structure. However, we can establish a graph isomorphism between the nodes and a certain set of octree cells. This isomorphism identifies an octree cell with the node in its upper-right-back corner. The set of octree cells that is required for this purpose contains the solid cells, the dilated cells, and all cells which are incident to the lower-left-front corner (any other pair of opposite corners could have been chosen as well) of a solid or dilated cell (refined to level l if not yet the case). The 6-neighborhood graph of these cells, directly represented by the ropes, can then be used instead of the nodes-arcs graph.

7.2. Memory Efficiency

The memory requirements of the entire procedure are mainly determined by the number of cells that are constructed and hence heavily depends on the resolution chosen for processing. High resolutions can be desirable since then the cell set bounds the mesh elements tighter and resolves finer features. The number of cells of the final octree of course depends on the geometry of the input, but in our experiments we observed that on average roughly 50 million cells are generated at a resolution of 4096^3. Since about 20-50 bytes need to be stored per cell (for index pointers to the parent cell and one child, location codes [FP02], state codes, a distance value, optionally a normal vector, an angle value, and ropes) this allows us to rasterize models at resolutions up to 4096^3 without exceeding today's PCs' main memories. However, the application of morphological operations results in a higher peak cell count. For instance, performing a dilation at the abovementioned resolution of 4096^3 to close holes and gaps up to a size of 3% of the bounding box diagonal increases the number of cells from 50 million to over 1000 million, clearly constraining applicability to lower resolutions in such cases.

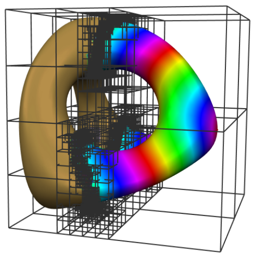
7.2.1. Tiling

To avoid this high memory peak we can perform the dilation and erosion process in a *tiled* fashion. The dilation and erosion operators are local in the sense that they only affect a local neighborhood of cells they are applied to, i.e. when applying γ dilation steps followed by γ erosion steps, the final state of a cell does not depend on cells farther away than 2γ in the 6-neighborhood graph of the cell set. Hence, when we apply these operations to one tile (the cell set clipped to an axis-aligned rectangular box) the resulting state of all cells except those in the outer 2γ cell layers of the tile is not affected by this clipping. By covering the bounding box with tiles overlapping by 4γ cells layers, the correct cell states can be obtained for all cells by applying the morphological operations to each tile separately. As pointed out in the last section restricting the erosion to γ steps results in a morphological closing of cavities in addition to gaps and holes, lowering the accuracy of distance computations. By applying $\gamma+\delta$ erosion steps (and choosing an overlap of $4\gamma+2\delta$) this can be alleviated to any desired degree. In our experiments $\delta = 3\gamma$ was the maximum encountered that was necessary to achieve the same results as with unlimited non-tiled erosion.

Since in each tile cells can be collapsed again after erosion, the peak cell count can be reduced significantly. Due to the required overlap this reduction is bounded depending on γ (and δ), but since the presence of large holes (requiring large γ) introduces significant uncertainties, the appropriateness of using very high resolutions to achieve high accuracy seems to be questionable in such cases anyway.

7.2.2. Sweeping

Despite the tiling approach the final cell set still has to fit into memory entirely. To allow for even higher resolutions we introduce a *sweeping* variant of our method. In this implementation variant cells are dynamically created by octree refinement when they are reached by the FM front propagation process and deleted by collapsing when the front has passed them. For this purpose we separate the morphological hole filing from the distance propagation process. We perform the morphological operations in a tiled fashion as described above but discard the cells of each tile after its construction – we only record the center point of each non-solid cell that survived the erosion. This yields a set of points which effectively “fill” gaps and holes of $\mathcal{M}$ up to voxel resolution.



The sweeping can then be performed without complex inline morphological operations by considering the union of $\mathcal{M}$ and this point set as input. The adjacent figure shows this sweeping in action: the octree is visualized at an intermediate state – only at the current front of propagation cells are at the finest level, away from it they are as coarse as possible.

In the following description of this sweeping, by the term “(octree) cell” we also refer to the node that is identified with the respective cell by the underlying graph isomorphism. In order to directly obtain the set of octree cells that are required to establish the isomorphism we do not consider cells whose volume is intersected by mesh elements (or hole-filling points) as solid, but cells whose volume extended by cell size λ in upper, right, and back direction is intersected. This avoids the subsequent additional refinement of cells incident to the lower-left-front corner – which would be cumbersome to manage in this sweeping variant. Cells record contained vertices as described in the previous section.

Initially only cells containing sources are refined to level l, initialized, and inserted into the front propagation queue. During the propagation, whenever a value is to be propagated into a solid neighboring cell (resp. node) this cell is refined to level l. When a cell c is removed from the queue (i.e. its distance and angle values are final) interpolation has to be performed (cf. Section 5.4). Due to the extended virtual cell size exactly those vertices whose interpolated values depend on the node corresponding to c are recorded at c. Hence we can easily add the values computed for c, multiplied by the trilinear interpolation factors, to the (zero-initialized) values of these vertices (“transposed interpolation”).

Afterwards, since it is not needed for interpolation anymore, c can be marked *collapsible* – unless there is a neighboring solid cell that is not yet finalized and hence might need the values of c for an FM update triggered by another cell. To cover this case we also call the collapsibility check for neighbor cells of c that are finalized but could not get marked collapsible so far. Whenever a cell gets marked collapsible its parent cell checks whether all children are collapsible and performs the collapse by deleting them. The parent cell is then marked collapsible and this process is invoked recursively to always obtain a maximally sparse octree.

8. Results

We now present some results generated with our implementation including runtime measurements. Experiments have been performed on a PC with 2.8 GHz Intel Core i7 CPU.

Figure 6 shows distance fields, geodesic paths, and parameterizations computed on a scanned model (358K polygons) which contains holes (partly non-simple, with *islands*). By choosing the dilation distance such that these holes are bridged the computed intrinsic distance approximations tolerate these defects. Computation times are presented in Table 1.

Figure 7 depicts another mesh as it commonly appears in practice. This mesh was created by a commercial CAD tool, by inconsistent tessellation of NURBS patches. It consists of nearly 1 million polygons in more than 11,000 non-aligned connected components (see the close-up). Converting such models into manifold one-component meshes usu-

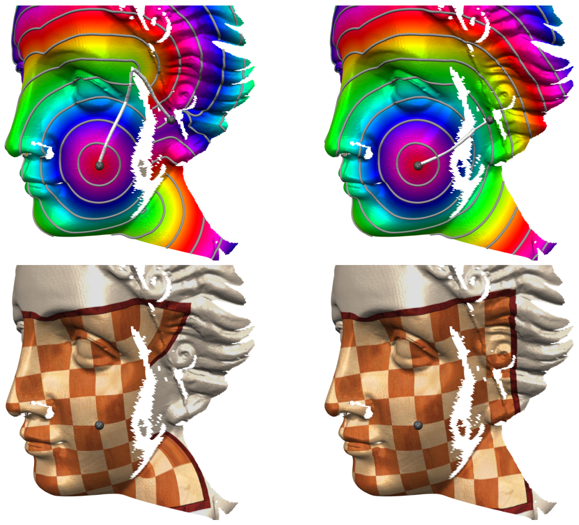


Figure 6: *Defect-tolerant computations on a raw scanned mesh* FACE *containing holes due to occlusion effects. Left: without morphological operations. Right: with morphological operations for hole bridging. The results of computations at resolution* 256^3 *are depicted.*

Resolution	64^3	128^3	256^3	512^3	1024^3
γ	2	4	8	16	32
Voxelize	1.12	1.40	2.0	3.3	6.1
Dilate	0.01	0.07	0.5	4.0	31.4*
Erode	0.02	0.09	1.0	9.4	80.2*
FM	0.02	0.09	0.3	1.4	5.6

Table 1: *Timings (in seconds) for distance field computation on model* FACE *(cf. Figure 6). *) Here* tiling *has been used (with* $\delta = 0$*; for* $\delta = 1.2\gamma$*, which sufficed to remove every unnecessary dilated cell, morphology took about 20% longer).*

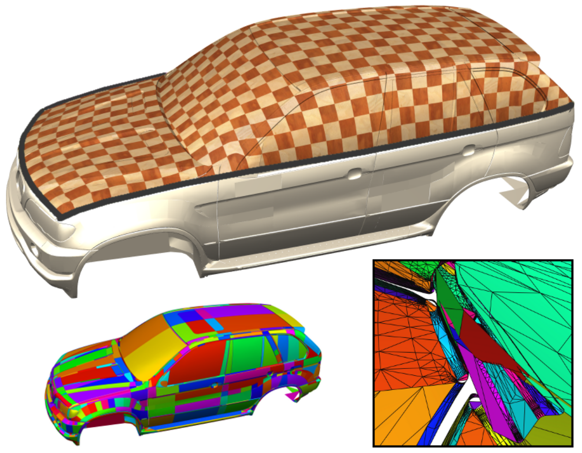


Figure 7: *Defect-tolerant texturing of an inconsistently tessellated NURBS model* CAR *by a boundary curve parameterization. The insets depict the color-coded individual connected components. As exemplified in the close-up, lots of non-trivial gaps, double-walls, and complex self-intersections are contained, essentially disqualifying boundary-snapping based algorithms for easy repair. The result of computation with resolution* 128^3 *is depicted.*

Resolution	256^3	512^3	1024^3	2048^3	4096^3	8192^3
Voxelize	4.3	5.8	8.8	18.4	-	-
FM	0.2	1.0	4.7	22.3	-	-
Sweeping	8.6	12.9	27.1	82.2	314.1	1370

Table 2: *Timings (in seconds) for one distance field computation on model* CAR *(cf. Figure 7). With the sweeping implementation higher resolutions can be handled (the peak cell count at* 8192^3 *is only 305K).*

ally requires computationally intensive global repair methods. Using our framework the model can be handled directly. Timings are presented in Table 2. Since the processing is blind to gaps and holes below leaf cell size the application of morphological operators was unnecessary in this case. Note that once rasterization, dilation, and erosion have been performed the obtained cubical complex (resp. octree) can be used for the quick computation of multiple distance fields etc. – it does not have to be rebuilt each time.

Figure 8 exemplifies the behavior on a polygon soup and shows how the morphological operators handle large holes/gaps. Figure 9 illustrates that also for consistent meshes the use of our abstraction method can be advantageous due to higher computational stability on the regular complex compared to mesh-based FM on an irregular mesh structure as also noted by Mémoli and Sapiro [MS01].

The distances computed by our method of course usually deviate from actual intrinsic distances (on consistent models) to some degree due to the finite resolution and the FM approach. To quantify this exemplarily we performed a ground truth comparison (cf. Figure 10). As can be seen in the included table the error decreases with increasing resolution until the surface detail is resolved sufficiently. Then the quality basically remains unchanged – due to the approximative nature of the FM approach it does not fully converge to the actual intrinsic distances. On account of the axis-aligned discretization computed distance fields depend on the input's orientation to some extent – the variation directly being determined by this remaining FM discretization error as could be observed in our experiments with random orientations.

Despite these deviations the computed distance fields are intrinsically fairly consistent in that they live up to the symmetry property of a metric well: the following table shows the relative symmetry errors encountered at various resolutions for 100 pairs of randomly chosen points on models TOOL (cf. Figure 9) and FERTILITY (cf. Figure 1).

	TOOL			FERTILITY		
	64^3	256^3	1024^3	64^3	256^3	1024^3
max.	2.59%	0.65%	0.40%	3.48%	0.87%	0.34%
avg.	0.75%	0.26%	0.06%	0.86%	0.29%	0.07%

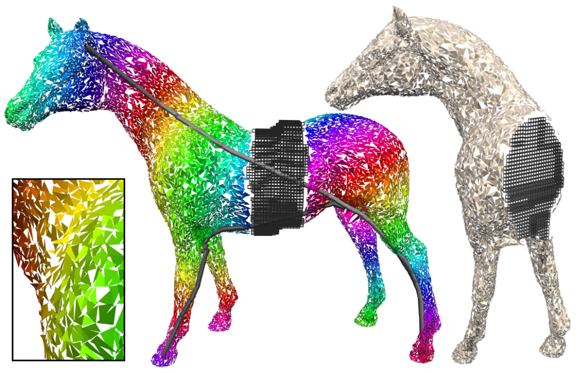


Figure 8: *Inconsistent polygon soup with a slice cut out. Dilated cells that survived erosion, a distance field, and two geodesic paths are visualized. Increasing the gap width at some point leads to the morph. operators closing the two holes instead of bridging the gap, as depicted on the right.*

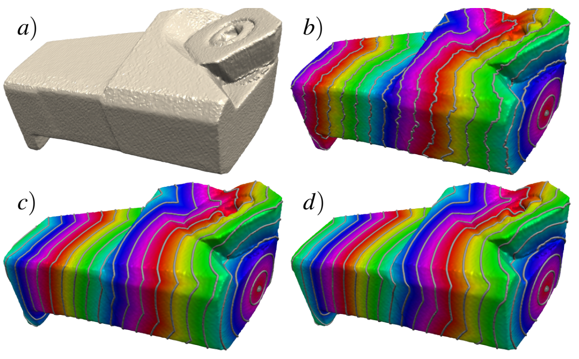
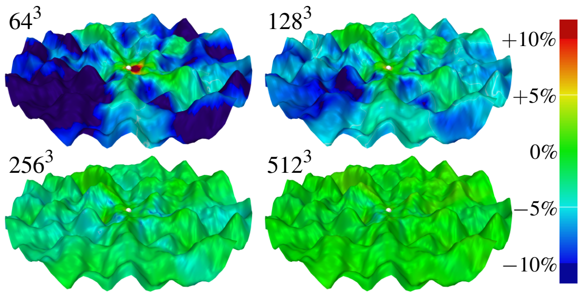


Figure 9: *a) Manifold but rough scanned model* TOOL. *b) Distance field computed using mesh-based FM [NK02]. c) Distance field computed by an implementation of exact window propagation [SSK*05] which can be considered ground truth (neglecting slight inaccuracies possibly caused by machine precision limitations). d) Distance field computed by our method (at resolution* 256^3*; >20 times faster than c).*

Almost all of the methods and applications mentioned in Sections 2.2 and 2.3 require clean manifold mesh input – most of them solely to enable geodesic distance computations. By performing these using the presented framework, the applicability of these methods in most cases trivially extends to meshes with several kinds of defects or even polygon soups. For instance can the shape analysis methods for model comparison [ZG04, GGGZ05, EK03, BBK06, SCF10] or for symmetry and regularity detection [RBBK07, MBB10] be applied to imperfect or partial meshes, and the FlexiStickers approach [TT09] for conveniently texturing models with casual photographs can as well be applied to texture *casual meshes*, making it even more opportune.

Limitations Due to the automaticity and generality of the method, it is naturally not able to resolve ambiguities that are inherent in the input due to large missing parts. Hence, the computed distance fields might be inconsistent with those of the object that is actually meant to be represented by input. Additional knowledge about the represented object or manual interaction would be required to more plausibly handle the hole bridging in such cases.



	64^3	128^3	256^3	512^3	1024^3	2048^3
max.	21.6%	13.3%	7.5%	3.8%	3.0%	3.1%
avg.	7.1%	4.6%	2.2%	0.8%	0.7%	0.7%
σ	14.6%	9.7%	4.5%	1.1%	0.6%	0.6%

Figure 10: *Visualization of the error to ground truth distances on an exemplary surface with a wide spectrum of frequencies for various resolutions. Max. and average of the abs. value of the rel. error, and standard dev. are specified.*

Future Work At high resolutions the total runtime is dominated by the morphological operations since these cause a large number of cell neighbor queries, cell splits, and collapses. We are currently investigating the use of multi-resolution morphology operators to reduce the computational cost at high resolutions. The basic idea is to (1) perform dilation followed by erosion on an inner octree level, i.e. at a lower resolution, (2) to propagate the result to the leaf level (marking descendant leafs of the cells that survived the erosion), and (3) to perform a final erosion on these leaf cells to obtain minimally thick hole-filling cell sheets again.

9. Conclusion

We presented a method that abstracts from the topological structure of a given input mesh, bridges gaps and holes up to a user-specified width, and thereby allows for the computation of plausible intrinsic distances and geodesic paths on meshes with all kinds of defects. The discrete volumetric abstraction is built in a way that high accuracy can be achieved and computationally optimal methods for the computation of distance fields can be adapted. We showed approaches to locally parameterize surfaces based on such geodesic fields in defect-tolerant ways, e.g. for the texturing of casual meshes. Many further existing methods and applications that rely on geodesic distance computations can readily be extended to deal with imperfect meshes by employing our framework.

Acknowledgements This research project was funded by the DFG Cluster of Excellence *UMIC* (DFG EXC 89).

References

[ABA02] ANDÚJAR C., BRUNET P., AYALA D.: Topology-reducing surface simplification using a discrete solid representation. *ACM Trans. Graph. 21*, 2 (2002), 88–105.

[BBK06] BRONSTEIN A. M., BRONSTEIN M. M., KIMMEL R.: Efficient computation of isometry-invariant distances between surfaces. *SIAM J. Scientific Computing 28*, 5 (2006), 1812–1836.

[BK05] BISCHOFF S., KOBBELT L.: Structure preserving cad model repair. *Comput. Graph. Forum 24*, 3 (2005), 527–536.

[BK07] BOMMES D., KOBBELT L.: Accurate computation of geodesic distance fields for polygonal curves on triangle meshes. In *Proc. VMV* (2007), pp. 151–160.

[BKW10] BÜRGER K., KRÜGER J., WESTERMANN R.: Sample-based surface coloring. *IEEE Transactions on Visualization and Computer Graphics 16*, 5 (2010), 763–776.

[BMBZ02] BIERMANN H., MARTIN I. M., BERNARDINI F., ZORIN D.: Cut-and-paste editing of multiresolution surfaces. In *SIGGRAPH '02* (2002), pp. 312–321.

[BPK05] BISCHOFF S., PAVIC D., KOBBELT L.: Automatic restoration of polygon models. *ACM Trans. Graph. 24*, 4 (2005), 1332–1352.

[BSLT00] BERNSTEIN M., SILVA V. D., LANGFORD J. C., TENENBAUM J. B.: *Graph Approximations to Geodesics on Embedded Manifolds*. Tech. rep., Stanford University, 2000.

[BZK04] BORODIN P., ZACHMANN G., KLEIN R.: Consistent normal orientation for polygonal meshes. In *CGI '04: Proc. Computer Graphics International* (2004), pp. 18–25.

[CC94] COX T., COX M.: *Multidimensional Scaling*. Chapman and Hall, 1994.

[CH90] CHEN J., HAN Y.: Shortest paths on a polyhedron. In *SCG '90: Proc. Symp. Comp. Geom.* (1990), pp. 360–369.

[dC76] DO CARMO M. P.: *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Englewood Cliffs, NJ, 1976.

[DC00] DESCHAMPS T., COHEN L. D.: Minimal paths in 3d images and application to virtual endoscopy. In *ECCV 2000, Proc. Europ. Conf. on Computer Vision* (2000), pp. 543–557.

[EK03] ELAD A., KIMMEL R.: On bending invariant signatures for surfaces. *IEEE Trans. PAMI 25* (2003), 1285–1295.

[FP02] FRISKEN S. F., PERRY R. N.: Simple and efficient traversal methods for quadtrees and octrees. *Journal of Graphics Tools 7*, 7 (2002), 2002.

[GGGZ05] GATZKE T., GRIMM C., GARLAND M., ZELINKA S.: Curvature maps for local shape comparison. In *Shape Modeling International* (2005), pp. 244–256.

[GKK02] GROSSMANN R., KIRYATI N., KIMMEL R.: Computational surface flattening: A voxel-based approach. *IEEE Trans. PAMI 24* (2002), 433–441.

[HF07] HASSOUNA M. S., FARAG A. A.: Multistencils fast marching methods: A highly accurate solution to the eikonal equation on cartesian domains. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29* (2007), 1563–1574.

[Ju09] JU T.: Fixing geometric errors on polygonal models: A survey. *J. Comput. Sci. Technol. 24*, 1 (2009), 19–29.

[KS93] KIRYATI N., SZÉKELY G.: Estimating shortest paths and minimal distances on digitized three-dimensional surfaces. *Pattern Recognition 26*, 11 (1993), 1623–1637.

[KS98] KIMMEL R., SETHIAN J. A.: Computing geodesic paths on manifolds. In *Proc. Natl. Acad. Sci.* (1998), pp. 8431–8435.

[MB90] MACDONALD D. J., BOOTH K. S.: Heuristics for ray tracing using space subdivision. *Vis. Comput. 6*, 3 (1990), 153–166.

[MBB10] MITRA N. J., BRONSTEIN A., BRONSTEIN M.: Intrinsic regularity detection in 3d geometry. In *ECCV* (2010), p. to appear.

[MD03] MOENNING C., DODGSON N. A.: *Fast Marching farthest point sampling for implicit surfaces and point clouds*. Tech. Rep. 565, University of Cambridge, 2003.

[MMP87] MITCHELL J. S. B., MOUNT D. M., PAPADIMITRIOU C. H.: The discrete geodesic problem. *SIAM J. Comput. 16*, 4 (1987), 647–668.

[MS01] MÉMOLI F., SAPIRO G.: Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces: 730. *J. Comput. Phys. 173*, 2 (2001), 764.

[MS04] MÉMOLI F., SAPIRO G.: Comparing point clouds. In *Proc. Symp. Geom. Proc.* (2004), pp. 32–40.

[MS05] MÉMOLI F., SAPIRO G.: Distance functions and geodesics on submanifolds of r^d and point clouds. *SIAM Journal of Applied Mathematics 65*, 4 (2005), 1227–1260.

[MZL*09] MEHRA R., ZHOU Q., LONG J., SHEFFER A., GOOCH A., MITRA N. J.: Abstraction of man-made shapes. *ACM Transactions on Graphics 28*, 5 (2009), #137, 1–10.

[NK02] NOVOTNI M., KLEIN R.: Computing geodesic paths on triangular meshes. In *WSCG* (2002), pp. 341–348.

[PC03] PEYRE G., COHEN L.: Geodesic remeshing using front propagation. In *INT. J. COMPUT. VISION* (2003), pp. 33–40.

[PS06] POLTHIER K., SCHMIES M.: Straightest geodesics on polyhedral surfaces. In *SIGGRAPH '06* (2006), pp. 30–38.

[RBBK07] RAVIV D., BRONSTEIN A. M., BRONSTEIN M. M., KIMMEL R.: Symmetries of non-rigid shapes. In *Proc. International Conference on Computer Vision (ICCV)* (Oct. 2007).

[RDSK06] RUGGERI M. R., DAROM T., SAUPE D., KIRYATI N.: Approximating geodesics on point set surfaces. In *Proc. Symp. on Point-Based Graphics* (2006), pp. 85–93.

[Sam89] SAMET H.: Neighbour finding in images represented by octrees. In *Vision, Graphics & Image Proc.* (1989), pp. 367–386.

[SCF10] SUN J., CHEN X., FUNKHOUSER T.: Fuzzy geodesics and consistent sparse correspondences for deformable shapes. *Computer Graphics Forum 29*, 5 (2010).

[Set95] SETHIAN J. A.: A fast marching level set method for monotonically advancing fronts. In *Proc. Nat. Acad. Sci* (1995), pp. 1591–1595.

[Set99] SETHIAN J. A.: Fast marching methods. *SIAM Review 41* (1999), 199–235.

[SGW06] SCHMIDT R., GRIMM C., WYVILL B.: Interactive decal compositing with discrete exponential maps. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (2006), pp. 605–613.

[SSK*05] SURAZHSKY V., SURAZHSKY T., KIRSANOV D., GORTLER S. J., HOPPE H.: Fast exact and approximate geodesics on meshes. *ACM Trans. Gr. 24*, 3 (2005), 553–560.

[TT09] TZUR Y., TAL A.: Flexistickers: photogrammetric texture mapping using casual images. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers* (2009), pp. 1–10.

[XW09] XIN S.-Q., WANG G.-J.: Improving Chen and Han's algorithm on the discrete geodesic problem. *ACM Trans. Graph. 28*, 4 (2009), 1–8.

[ZG04] ZELINKA S., GARLAND M.: Similarity-based surface modelling using geodesic fans. In *Proc. Symposium on Geometry Processing* (2004), pp. 204–213.