Volume xx (200y), Number z, pp. 1–9

Efficient Computation of Smoothed Exponential Maps

Philipp Herholz and Marc Alexa

Abstract

Many applications in geometry processing require the computation of local parameterizations on a surface mesh at interactive rates. A popular approach is to compute local exponential maps, i.e. parameterizations that preserve distance and angle to the origin of the map. We extend the computation of geodesic distance by heat diffusion to also determine angular information for the geodesic curves. This approach has two important benefits compared to fast approximate as well as exact forward tracing of the distance function: First, it allows generating smoother maps, avoiding discontinuities. Second, exploiting the factorization of the global Laplace-Beltrami operator of the mesh and using recent localized solution techniques, the computation is more efficient even compared to fast approximate solutions based on Dijkstra's algorithm.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1. Introduction

Many interactive geometry processing applications require local parameterizations around a point of interest, e.g. for local texture mapping (decals) or editing [TSS*11, Sch13]. Desirable properties for these maps are:

- their computation should be as fast as possible, enabling operations at interactive rates for moderately sized regions, and
- the isometric distortion of the parameterization should be small around the point of interest and then possibly degrade with distance.

Let **p** be the desired origin of the local parameterization. Fast local parameterization is commonly based on the idea of tracing geodesics emanating from **p**. Each point can then be parameterized based on its geodesic distance to **p** and the angle of this geodesic at **p** relative to a fixed reference direction in the tangent frame. In the plane, using straight lines as paths, this approach leads to polar coordinates. The concept analogous to straight lines on a surface are geodesic paths leading to geodesic polar coordinates. Geodesics connect two points along a shortest path on the surface. In contrast to the planar setting, there might be more than one shortest path. On the sphere, for example, there are infinitely many shortest path connecting a point to its antipode. Local parameterizations based on geodesics therefore only make sense for regions where there is a unique shortest geodesic for every point connecting it to the central point **p**. In this case the polar coordinates induced by the geodesics are refereed to as *log map* or, arguably more common, by its inverse, the *exponential map* (see Figure 4).

While tracing geodesics may be natural and the concept of exponential maps may be well established there are fundamental and practical problems: distinct geodesic curves emanating from the origin **p** may intersect, leading to singularities in the parameterization. Also, computing exact geodesic curves on discrete surfaces is expensive. This suggests that one rather wants to approximate the exponential map, focusing on fast computation rather than generating exact geodesic distances and angles [SGW06].

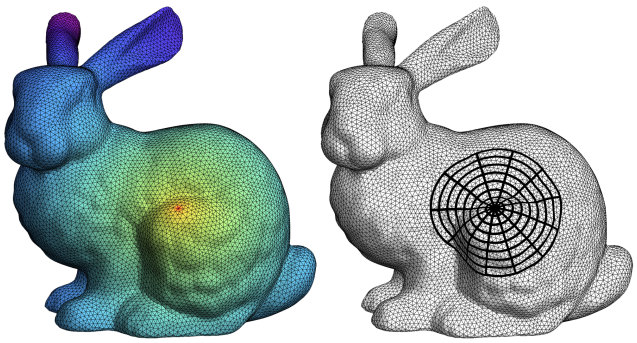

Figure 1: *Computing the heat kernel centered at a small set of vertices (left) can be used to efficiently compute exponential maps (right).*

For discrete surfaces Crane et al. [CWW13] show how distances can be computed by solving sparse linear systems. In particular they observe that gradients of the discrete heat kernel at a specific vertex closely approximate gradients of the distance field. Our technical contribution is an extension of this method to also determine the required angular information at the center vertex (see Figure 1). Since our method is based on evaluating angles of gradients formed with a fixed reference direction it inherits the robustness of the original method. As a result, we can efficiently compute both, distances as well as angles, by solving sparse linear systems.

The heat kernel is the result of simulating heat diffusion for a short time starting with a singular heat source. For larger times, however, one gets curves that resemble geodesics but are smoother. In other words, the diffusion time provides a parameter that allows to control the smoothness of the map. This property is useful in presence of surface features that would cause the exact exponential map to be discontinuous or exhibit large distortion.

Defining the linear systems on the local surface patch and then solving them using either direct or iterative methods would be significantly slower than forward traversals from the center vertex. However, Herholz et al. [HDA17] demonstrate how to reuse the factorization of global systems for local diffusion problems, resulting in a significant speedup for the type of problem we consider. Our central observation is that the computation of distances *and angles* is in fact faster than methods based on Dijkstra's algorithm. We believe this is a counter-intuitive and very useful result. It can be explained by the fact that the factorization of the global system is effectively a precomputation step, which is exploited in the solution of a particular local problem.

We demonstrate how to exploit the symmetry of the diffusion problem to obtain angular information for all vertices in a specific region using only a few local solves of the prefactored system.

This allows us to compute the angles at almost negligible extra cost compared to the distances along the paths. We demonstrate that our method is not only overall faster than existing techniques but also that the smoothness resulting from using diffusion for the definition of paths avoids fold-over and distortions occurring in exact exponential maps or Dijsktra-based approximations.

2. Related work

Our work aims at the fast generation of local parameterizations, suitable for use in interactive applications. We focus our description of related work on the subset of local parameterization techniques that are capable of generating parameterizations at interactive rates. We will describe work that we make use of or that is comparable in certain aspects in the respective later sections of this paper.

We assume the shape we consider is given as a triangle mesh. The mesh $\mathcal{M}$ is represented as a set of vertices $\mathbf{v}_i \in \mathbb{R}^3, i \in [0,\dots,n-1]$, a set of edges $\{(i,j)\} = \mathcal{E}$ and a set of triangles $\{(i,j,k)\} = \mathcal{F}$.

Schmidt et al. [SGW06] (DEM) approximate exponential maps at a vertex $\mathbf{v}_i$ by augmenting Dijkstra's algorithm. For each vertex $\mathbf{v}_j$ an arbitrary tangent frame F_j orthogonal to the vertex normal is determined. Next, local exponential maps for each vertex are computed by rotating each adjacent edge of that vertex into the tangent frame. To express vectors in a tangent frame F_j with respect to F_i a rotation aligning both frames is computed. To find the final exponential map, the mesh graph is traversed using Dijkstra's algorithm starting at vertex $\mathbf{v}_i$. When traversing an edge its local exponential map representation is rotated to the reference frame and added up. The robustness of this approach can be improved by averaging local exponential maps from all already visited neighbors in each Dijkstra step [Sch10]. We use this robust version of the algorithm in our comparisons.

The algorithm merely computes an approximation of the exponential map because it only uses linear isotropic mappings between coordinate systems and approximates the real geodesic by an edge path. Nevertheless, the mappings produced resemble exact exponential maps nicely in many practical cases.

Malvær et al. [MR12] (DGPC) also propagate information using a Dijkstra-like algorithm. They proceed by inferring information for a vertex in a triangle when distance information to both other vertices is known. To this end a virtual source point in the triangle plane is computed that has the correct distance to both vertices. The authors report increased accuracy as compared to Schmidt et al. [SGW06], assuming exact exponential maps as the reference.

Exact exponential maps can be defined using exact polyhedral geodesics, i.e. the shortest paths connecting two vertices on the mesh geometry. There are several algorithms computing this piecewise linear path. While earlier approaches where rather slow [MMP87, CH90] more sophisticated implementations could improve performance significantly [SSK*05, XW09, WFW*17]. Some of these algorithms can sacrifice accuracy for performance. Ying et al. [YWH13] present an algorithm with tunable accuracy that also propagates angular information thus producing all information needed for a exponential map. The authors report that they can outperform Crane et al. [CWW13] when the accuracy is relatively low but fail to match their performance when comparable accuracy is requested. Precomputation times are several orders of magnitude higher than for the method of Crane et al. in that case. Since our algorithm has essentially the same precomputation step as Crane et al. [CWW13], this comparison carries over to our method. Algorithms that compute polyhedral geodesics, either exact or approximate, have the advantage that they are usually not affected by the triangulation quality in contrast to heat based methods that can exhibit artifacts for anisotropic, irregular and non-Delaunay meshes. This comes usually at a higher performance cost. Moreover, we have found exponential maps based on polyhedral algorithms to lack smoothness because angles are strongly influenced by the shape of triangles adjacent to the source vertex. Furthermore, even the smallest cavity on a smooth mesh will lead to artifacts due to non-unique geodesics. We argue that smoothed versions of the exponential map, as computed by our heat based method, are preferable over exact geodesic exponential maps in many scenarios.

Sun et al. [SZZ*13] compute local stroke parameterizations enabling texture manipulation on a mesh. They employ exact polyhedral geodesics based on on Xin et al.'s work [XW09, XYH11]. Each point close to a stroke is parameterized by its distance to the stroke and the arclength at the closest point.

Exponential maps on triangular meshes have recently be used to define local charts for convolutional neural networks [MBBV15]. In these applications very fast approximations of the exponential map for small neighborhoods are required. This emphasis on performance, however, can introduce significant inaccuracies that seem to be tolerable for this specific task.

In concurrent work Sharp et al. [SSC18] introduce a heat based method to perform parallel transport of vector valued data on meshes. The basic idea of their method is similar to ours, yet they approach the development from the perspective of generalizing the

heat method. This naturally leads to stronger theoretical foundation. They also provide a wide range of different instances for specific problems, including how to quickly compute exponential maps.

In contrast, we focus on the specific problem and provide a detailed analysis of the preservation of length and angles as well as the behavior of singularities w.r.t. the time step, and contrast this behavior with a range of possibly competing methods.

3. Preliminaries

We introduce some notation and quantities associated with the triangulated surface. In particular, we need the area vector per face and the mean curvature normal derived from the discrete Laplace-Beltrami operator.

To each triangle ijk we assign the area vector

$$\mathbf{a}_{ijk} = \frac{1}{2}(\mathbf{v}_i - \mathbf{v}_j) \times (\mathbf{v}_j - \mathbf{v}_k). \tag{1}$$

The magnitude $\|\mathbf{a}_{ijk}\|$ of this vector corresponds to triangle area and its direction to the normal vector

$$\mathbf{n}_{ijk} = \mathbf{a}_{ijk} / \|\mathbf{a}_{ijk}\|. \tag{2}$$

For discretizing the heat equation we need a discrete Laplace-Beltrami operator. We suggest to use the cotan operator [PP93, MDSB03], which we denote as $\mathbf{B} \in \mathbb{R}^{n \times n}$ and a lumped mass matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$. Using this operator we define normal directions at the vertices as the direction of the area gradient (or, equivalently, the mean curvature)

$$\hat{\mathbf{n}}_i = \left(\mathbf{B}(\mathbf{v}_0, \mathbf{v}_1, \ldots)^\mathsf{T}\right)_i. \tag{3}$$

This allows us to assign a unit normal direction as

$$\mathbf{n}_i = \begin{cases} \hat{\mathbf{n}}_i / \|\hat{\mathbf{n}}_i\| & \hat{\mathbf{n}}_i \neq \mathbf{0} \\ \sum_{ijk} \mathbf{a}_{ijk} / \|\sum_{ijk} \mathbf{a}_{ijk}\| & \text{else,} \end{cases} \tag{4}$$

where we derive the vertex normal from the area vectors if the mean curvature is zero.

4. Background

4.1. Distances from Diffusion

Crane et al. [CWW13] compute geodesic distances by employing a result by Varadhan [Var67] stating that geodesic distance can be computed as a point wise transformation of the heat kernel. The heat kernel is the fundamental solution of the heat equation and describes how much heat is distributed from a point $\mathbf{p}$ to a point $\mathbf{q}$ on the surface in time t.

The heat kernel for a fixed point $\mathbf{p}$ and time t can be approximated on a surface mesh by solving a sparse linear system. For this the heat equation is discretized using the discrete Laplace-Beltrami operator $\mathbf{B}$ [CWW13]. Solving the linear system

$$(\mathbf{M} - t\mathbf{B})\mathbf{h}_i = \mathbf{A}\mathbf{h}_i = \mathbf{e}_i \tag{5}$$

where $\mathbf{e}_i$ is the $i-$th unit vector, representing a singular heat source at vertex i, yields values of the heat kernel for the fixed source point

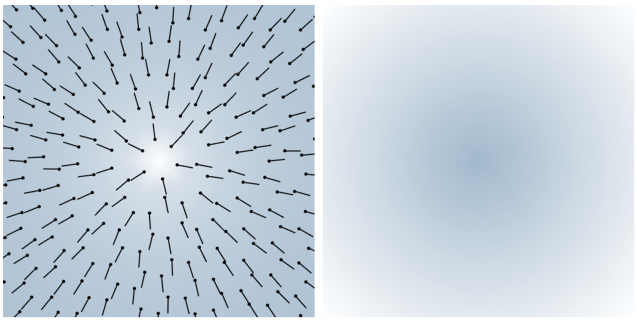

Figure 2: *Computing geodesic distance using heat diffusion. First the heat kernel is evaluated at a vertex and the piecewise linear gradient is computed (left). Integrating the normalized gradients yields geodesic distances (right).*

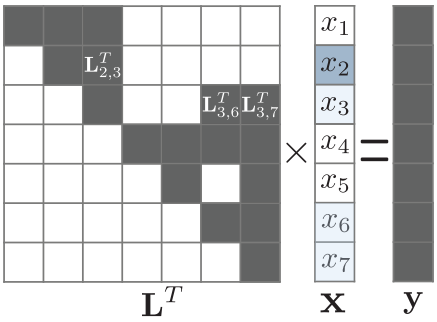

Figure 3: *By exploiting sparsity in the Cholesky factor not all variables have to be determined in order to compute x_2.*

$\mathbf{v}_i$ and time step t, see Figure 2 (left). Instead of using these values directly the authors observe that accuracy can be significantly improved by exploiting the fact that the gradient of the heat kernel is parallel to the gradient of the distance function. The gradient of the piecewise linear function defined by $\mathbf{h}_i$ can be computed for triangle $k = (k_1, k_2, k_3)$ as

$$\mathbf{g}^k_i = \frac{1}{2\|\mathbf{a}_k\|} \sum_{s=1}^{3} (\mathbf{h}_i)_{k_s} (\mathbf{n}_k \times (\mathbf{v}_{k_{s+2}} - \mathbf{v}_{k_{s+1}})) \tag{6}$$

where indices are interpreted modulo 3. Because the gradient of the distance function has unit length everywhere, the correct and desired gradient field is obtained by normalizing the gradient of the heat values on each triangle

$$\hat{\mathbf{g}}^k_i = \mathbf{g}^k_i / \|\mathbf{g}^k_i\|. \tag{7}$$

Integrating this gradient field by solving the Poisson equation

$$\mathbf{B}\mathbf{d}_i = \mathbf{D}(\hat{\mathbf{g}}^0_i, \hat{\mathbf{g}}^1_i, \ldots)^\mathsf{T}, \tag{8}$$

where $\mathbf{D}$ is the discrete divergence operator, results in the distance values $\mathbf{d}_i$.

4.2. Localized linear solves

The cost for the computation of geodesic distances using heat diffusion is dominated by the solution of two linear systems. Since both

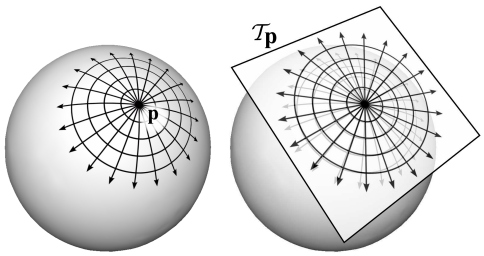

Figure 4: *Illustration of the exponential map centered at* $\mathbf{p}$*. Image courtesy of Ryan Schmidt.*

of them are sparse, symmetric and positive semi-definite the systems can be efficiently solved by constructing the sparse Cholesky factorization $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^\mathsf{T}$ [Dav11] where $\mathbf{L}$ is a sparse lower triangular matrix and $\mathbf{D}$ a diagonal matrix. Given this factorization, systems can be solved by forward- and back substitution

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{b} \\ \mathbf{L}\mathbf{D}\mathbf{L}^\mathsf{T}\mathbf{x} &= \mathbf{b} \\ \mathbf{L}\mathbf{D}\mathbf{y} &= \mathbf{b} \qquad \text{Forward substitution} \\ \mathbf{L}^\mathsf{T}\mathbf{x} &= \mathbf{y} \qquad \text{Back substantiation} \end{aligned}$$

The main computational burden is the factorization step. Herholz et al. [HDA17] introduced a method that allows to use a precomputed factorization of a *global* linear system representing heat diffusion to efficiently solve a linear system on a small surface patch. The key idea is to systematically analyze the dependencies of the values in $\mathbf{y}$ and $\mathbf{x}$ during forward- and back substitution. As it turns out only a relatively small subset of values in the solution vector has to be computed in order to yield exact values at vertices in a surface patch.

Consider the linear system illustrated in Figure 3, where colored squares stand for non-zeros. Solving for the vector $\mathbf{x}$ by back substitution is followed by determining the values x_i in reverse order. If we are only interested in a subset of the values, say $\{x_2\}$, not all values have to be computed. Because of the non-zero $L^\mathsf{T}_{2,3}$, the value x_3 has to be determined in order to evaluate x_2. x_3 in turn depends on x_6 and x_7. The values x_1, x_4 and x_5 do not have to be computed at all. All necessary values can be determined very efficiently by traversing the so called elimination tree. Interestingly also the substituted vector $\mathbf{y}$ is sparse for a sparse right-hand side $\mathbf{b}$. As a consequence exact values for a subset of all variables can be computed in a runtime that largely depends on the number of requested values. For details we refer the reader to [HDA17].

Both linear systems can be prefactored and reused to efficiently solve for geodesic distances between an arbitrary vertex and all others in a small local patch.

5. Algorithm

Given a center vertex $\mathbf{v}_i$ we wish to compute the log map for a set of vertices $\{\mathbf{v}_j\}_{j\in\mathcal{I}}$. Each vertex in this set is mapped to the geodesic distance to $\mathbf{v}_i$ and the the angle of the tangent vector defined by that

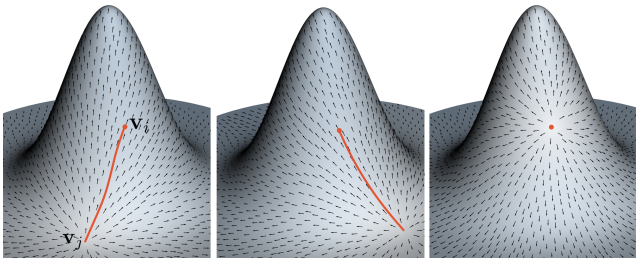

Figure 5: *Left: Simulating heat diffusion starting at vertex* $\mathbf{v}_j$ *we obtain tangent vectors to geodesics passing through* $\mathbf{v}_j$*. Evaluating these vectors at* $\mathbf{v}_i$ *defines the angular component of the log map centered at* $\mathbf{v}_i$*. Right: We show that in order to evaluate tangent vectors at* $\mathbf{v}_i$ *for all vertices in a region it is enough to solve only a few diffusion problems close to the center of the map.*

geodesic at $\mathbf{v}_i$ with a fixed reference direction. We can use the methods laid out in the previous section to compute distances based on solving prefactored sparse linear systems. The remaining problem is to compute the angular part of the log map. Our first observation is similar to Crane et al. [CWW13]: the tangents of geodesics are parallel to the gradients of the solution of heat diffusion for short times t.

An essential property of our solution is symmetry: the heat diffused in time t from vertex i to vertex j is identical to the heat diffused from j to i in the same time. This property is exactly captured in the discrete setting – at least if the discrete Laplace operator matrix $\mathbf{L}$ and mass matrix $\mathbf{M}$ are symmetric. Then the heat diffused from i to j is $(\mathbf{h}_i)_j$ where $\mathbf{h}$ is the solution to Eq. 5. This is just the ij entry of $(\mathbf{M} - t\mathbf{B})^{-1} = \mathbf{A}^{-1}$. Since the matrix $\mathbf{A}$ is symmetric, its inverse is also symmetric and we have $(\mathbf{h}_i)_j = (\mathbf{h}_j)_i$.

Computing the angle for one vertex To compute the tangent to the geodesic connecting vertices i and j we diffuse heat starting at vertex j and evaluate the gradient in vertex i. To this end we solve Equation 5 with the right hand side $\mathbf{e}_j$. Based on the heat values $\mathbf{h}_j$ we can compute the heat gradient per triangle using Equation (6). To define the gradient of $\mathbf{h}_j$ at vertex i we use gradients $\mathbf{g}_j^k$ in the triangles adjacent to it. We represent the discrete tangent space at vertex i by the vector space orthogonal to the (normalized) mean curvature normal $\mathbf{n}_i$ (4). Projecting the vertex star of i along $\mathbf{n}_i$ results in a conformal flattening, meaning the relative angles incident at vertex i are being preserved. Each gradient $\mathbf{g}_j^k$ is projected onto this discrete tangent space, as illustrated in Figure 6, leading to

$$\bar{\mathbf{g}}_j^k = \mathbf{g}_j^k - \mathbf{n}_i\mathbf{n}_i^\mathsf{T}\mathbf{g}_j^k. \tag{9}$$

The final direction is then computed by taking the area weighted average of the projected gradients:

$$\mathbf{g}_j^i = \sum_k A_k \bar{\mathbf{g}}_j^k. \tag{10}$$

The angle of this vector to a fixed reference direction in the discrete tangent space represents the angular component of the log map and forms, together with the distance information, geodesic polar coordinates for vertex $\mathbf{v}_j$ with respect to the center $\mathbf{v}_i$. Figure 5 (left)

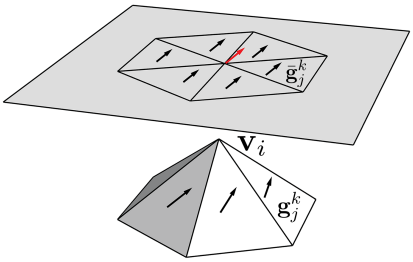

Figure 6: *Evaluating the gradient of the heat kernel centered at $\mathbf{v}_j$ in all triangles adjacent to $\mathbf{v}_i$ and averaging in the tangent plane gives the angular component of the map.*

illustrates the gradient vectors for many vertices, however, we are only interested in the vector at the center vertex of the log map.

Computing directions for all vertices in the region Suppose we want to evaluate the log map with respect to $\mathbf{v}_i$ for all vertices in a specific region. Using the technique described in the previous paragraph would require solving the heat diffusion problem

$$(\mathbf{M} - t\mathbf{B})\mathbf{h}_j = \mathbf{e}_j \tag{11}$$

for each vertex $\mathbf{v}_j$ in that region. To compute the gradient we are actually only interested in a few elements of $\mathbf{h}_j$, namely those associated with the vertex $\mathbf{v}_i$ and its immediate neighbors $\mathcal{N}(i)$. While solving for these few values is very efficient using the localization technique described before, doing so for a large number of vertices still leads to an overall inefficient approach.

Yet, we observe that symmetry allows us to reverse the roles of source and destination. This means, instead of evaluating $(\mathbf{h}_j)_k$ for every j and $k \in \mathcal{N}(i) \cup \{i\}$ we can evaluate $(\mathbf{h}_k)_j$, giving us the same values. This way we only need to perform $d = |\mathcal{N}(i) \cup \{i\}|$ forward- and back substitutions. We can easily compute these solutions in parallel exploiting vectorization which allows us to traverse the sparse matrix data structures only once during forward and back-substitution. This means that we can compute d solutions in time sub-linear in d in practice. Moreover, we need the values $\mathbf{h}_i$ anyway to compute geodesic distances.

Efficient computation Even though we are interested in the exponential map only in a small neighborhood around the source vertex the prefactored linear systems give values for all vertices of the mesh. Therefore performance depends on the size of the full mesh which is undesirable in most applications. A possible remedy would be to extract a local neighborhood of the mesh and solve the linear systems locally. This requires factoring two linear systems for each instance of the problem. Moreover, the result will differ since the heat kernel using the full operator considers heat diffusing over the complete mesh. Using a sufficiently large neighborhood, however, would mitigate this effect close to the surface vertex, see [HHA17] for a discussion of this approach.

Using the approach described by Herholz et al. [HDA17] offers a way to sidestep the repeated factorization of the local problems by reusing a global factorization. This technique computes the *exact* same numerical values as traditional forward and back substitutions. The runtime of this approach, however, depends only very mildly on the size of the full mesh.

6. Implementation

The implementation of the proposed approach is straightforward given an implementation of Cholesky factorization and local back substitution. Input for the algorithm is the index of the source vertex i and a set of vertex indizes $\mathcal{I}$ representing the neighborhood for which the exponential map shall be computed. In a preprocess Cholesky factorizations of the heat operator $\mathbf{A}$ and the mesh Laplacian $\mathbf{B}$ have been computed for the full mesh. The algorithm proceeds as follows:

1. Locally solve the heat equation (5) for i and all vertices connected to i using the precomputed Cholesky factor [HDA17] resulting in vectors $\mathbf{h}_k$.
2. Compute geodesic distances using $\mathbf{h}_i$ following Crane et al. [CWW13], integration of the gradient field is again computed by locally solving against a prefactored system.
3. For each vertex $j \in \mathcal{I}$ compute the gradient of the values $(\mathbf{h}_k)_j$ in all triangles connected to i, project them onto the tangent plane at i, build the area weighted average and compute the angle between the gradient and a fixed reference direction in the tangent plane.
4. Optionally compute coordinates by converting the polar coordinates to Cartesian ones.

7. Evaluation

We evaluated our algorithm on a set of meshes with different characteristic features as well as standard geometries. It is well known that the heat method can produce distorted results if the mesh is not Delaunay [LXFH15]. We therefore restrict our analysis to Delaunay meshes where possible. For meshes that do not possess this property a Laplacian based on the intrinsic Delaunay triangulation has to be used [FSBS06, LXFH15].

We compare our algorithm to the Dijkstra based approaches DEM and DGPC. For DEM we use our carefully optimized custom implementation including improvements regarding robustness introduced in [Sch10]. We have empirically found that a Dijkstra implementation based on Fibonacci heaps for dynamically updating the priority queue provides the best performance. For DGPC we use publicly available code provided by the authors.

7.1. Performance

To evaluate performance we conducted two basic experiments. First, we vary the patch size for different meshes and compute exponential maps using our method, DEM and DGPC (see Figure 7). Even though our algorithm exploits the sparse local evaluation method there is still a dependency of runtime on the size of the full mesh which is not the case for the alternative approaches. Across meshes of different sizes we find that our method outperforms its competitors when the patch covers at least 5% of the mesh's vertices.

In a second experiment (Figure 8) we keep the shape of the mesh

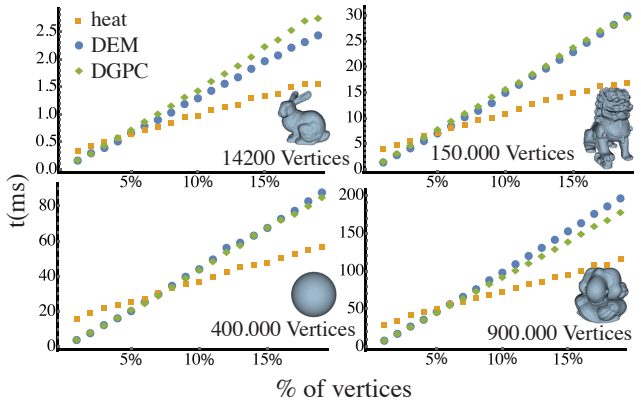

Figure 7: *Performance for exponential maps using our method (heat), discrete exponential maps (DEM) [SGW06] and discrete geodesic polar coordinates (DGPC) [MR12]. We compare timings for fixed meshes and varying patch size (measured in fraction of total vertices). Each measurement is averaged across 50 random patches.*

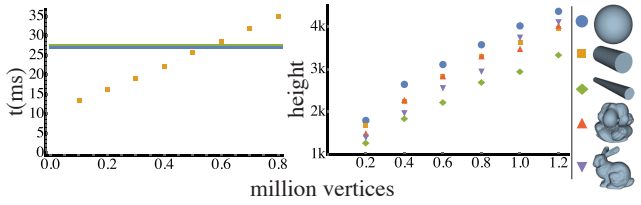

Figure 8: *Keeping the patch size fixed at 25000 vertices we vary the resolution of the full mesh. Left we see the performance of the heat method (yellow) which can beat DEM (blue) and DGPC (green) if the patch size is below 5% of the full mesh. Consequently we see our method outperform its competitors for meshes with fewer than 5×10^5 vertices. The right figure illustrates the height of the elimination tree for different mesh resolution, revealing the influence of mesh geometry on performance.*

and the patch size fixed and vary the tessellation density. The performance of DEM and DGPC is unaffected because their runtime only depends on a local traversal that is not influenced by the size of the full mesh. As seen in the first experiment our algorithm is only more efficient than the other two if the size of the exponential map is more than 5% of the full mesh in terms of vertex count. We see this result confirmed in the second experiment where we choose a patch size of 2.5×10^4 vertices and consequently see better performance only for tessellations with fewer than 5×10^5 vertices.

That our approach is faster than simple traversal algorithms can be explained by the fact that the factorization of the system matrices is an effective precomputation step. This precomputation step may also be considered the downside of our method. In order for the factorization to be reasonably sparse and allow for fast local back solves, the matrices have to be reordered using nested dissection (see [HDA17]).

Elimination tree The runtime of our algorithm is dominated by the computation of heat values by solving a sparse linear system. We employ the sparse local solve method by Herholz et al. [HDA17] which minimizes computation time by discarding values that do not have to be computed in order to evaluate heat values in a local region. The amount of values that can be discarded is directly linked to the so called elimination tree (see [HDA17] for details). If this tree is fairly balanced the performance of our method benefits the most. It is therefore natural to ask how different geometries influence the tree and what the worst case for the balancedness of this tree is. While we do not provide a formal proof we argue that spherical objects are among the worst when it comes to a balanced elimination tree. This statement is motivated by the fact that the shape of the tree is influenced by a reordering of the vertices e.g. using the nested dissection approach. An efficient reordering in terms of a balanced elimination tree is obtained by the following procedure:

1. find a small set of vertices (cut set) that divides the mesh graph in two sub meshes of roughly the same size.
2. reenumerate the vertices starting with the first sub mesh followed by the second and the vertices in the cut set.
3. recursively apply these steps on the sub meshes.

The elimination tree is balanced if both demands in 1. are fulfilled. For spherical object it is not possible to find a very small dividing set in order to split the mesh. This is in contrast to cylindrical shapes which can be cut much more efficient, especially if the ratio of radius and length is small.

In Figure 8 we indirectly measure balancedness by tree height and illustrate its dependency on the tessellation density for different shapes. As expected, the sphere yields the highest tree and therefore the worst performance of our algorithm. The Pensatore model is quite spherical and therefore close to the sphere in terms of tree height. The cylinder with the larger length/radius ratio performs best. Since we evaluated performance on different objects including the sphere we believe that our performance results extend to a large set of meshes.

7.2. Quality

The main goal of our algorithm is to generate smooth exponential maps even close to the cut locus or areas of high curvature where competing methods produce artifacts. However, we also evaluate accuracy compared to analytic results, even though this is not our main goal and might even contradict smoothness. We can show that our method still produces quite accurate results for different standard geometries.

Smoothness We compare our algorithm to the approaches by Schmidt et al. [SGW06] (DEM) and Malvær et al. [MR12] (DGPC) in terms of smoothness of the exponential map. Figure 9 shows some rather challenging cases of regions of high curvature that might also include parts of the cut locus (e.g. second row, close to the eyes). In all cases our algorithm produces smooth injective maps whereas other methods exhibit fold over and discontinuities in the map. These problems are usually more pronounced for DEM. For regions of low curvature all results are very close (last row).

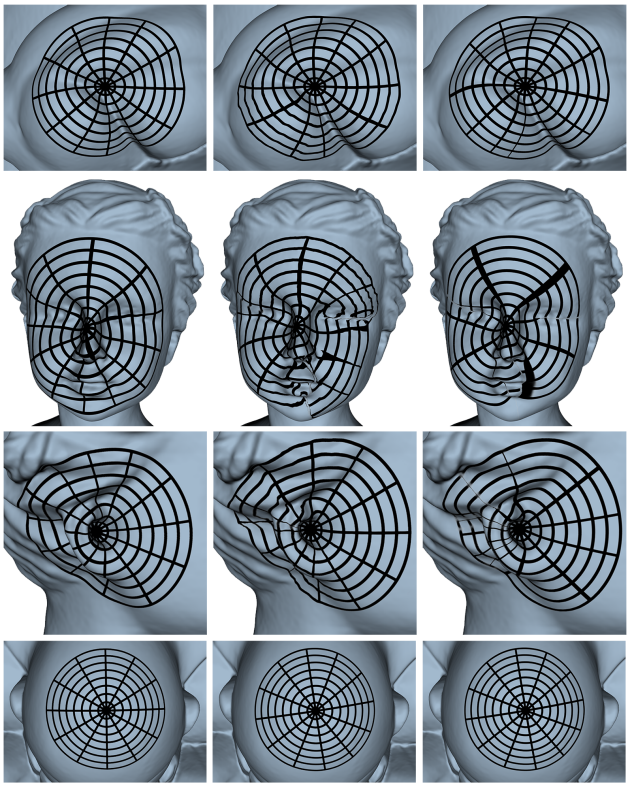

Figure 9: *Quality comparisons for our method (left), DEM (center) and DGPC (right). Our method yields smoother results for regions of high curvature. For smooth regions all methods perform similar.*

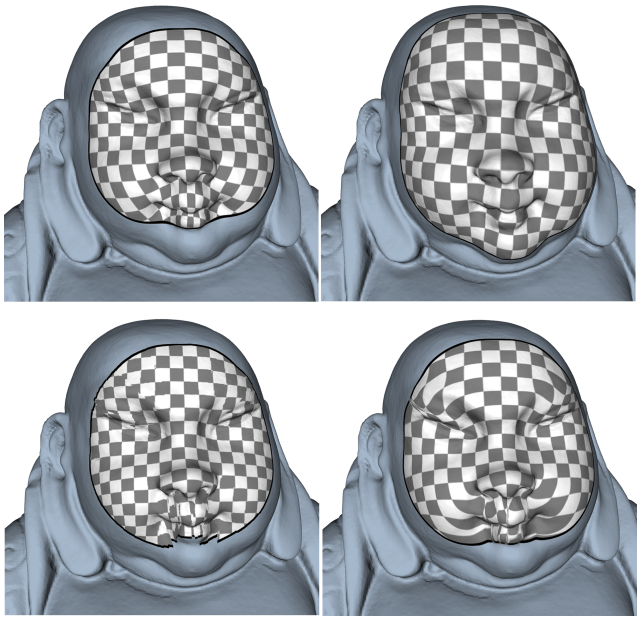

Figure 10: *Comparison of our method (top left), conformal map (top right), DEM (lower left) and DGPC (lower right).*

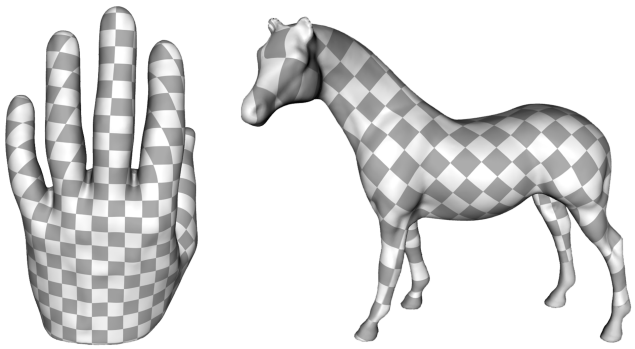

Figure 11: *Our exponential maps extend smoothly across the whole surface except for the cut locus.*

Our algorithm also performs well on very large meshes (Figure 11, 3.5M vertices) and does not only produce smooth maps but also outperforms competing methods. In this example we also compare to discrete conformal maps [DMA02] which might introduce severe area distortion and is therefore not considered in the following experiments.

Exponential maps computed using our method are smooth even far away from the central vertex (see Figure 11). However, they might exhibit distortions at elongated features, like the exact exponential map would. Unlike competing approaches our technique does not produce noise for these regions.

Our method can not guarantee injectivity of the map. Figure 14 (left) shows an extreme case. Even though our algorithm does not produce an injective map it is still smooth in contrast to other methods. Note, that even exact polyhedral geodesics as implemented by Surazhsky et al. [SSK*05] will introduce artifacts at the cut locus because the exponential map is not defined at these points. By tweaking the time step parameter t in equation (5) we can also compute injective maps for these extreme cases. A parameter of 10^2h^2 produces an injective map that is considerable smoothed and therefore not as close to the ground truth exponential map. Choosing smaller values for t yields exponential maps that are more accurate but exhibit artifacts. The parameter is therefore useful in order to tune the trade off between accuracy and smoothness. Unless noted otherwise, we use the squared mean edge length h^2 for this parameter, as suggested by Crane et al. [CWW13].

Both Dijkstra based algorithms propagate information along one path and introduce error due to the inherent assumption that mappings between tangent spaces are isotropic, which is not true in general. Moreover, this implies that in cases where geodesics are not unique the algorithm decides arbitrarily on one specific path. Heat diffusion, in contrast, can be interpreted as tracing a large number of particles moving randomly across the mesh [Law10]. Our approach therefore propagates information along all possible paths simultaneously and consequently yields smoother maps. For planar meshes the exponential maps by Schmidt et al. [SGW06] and Malvær et al. [MR12] both produce exact results by construction.

Accuracy To evaluate accuracy we compare results on three analytically defined geometries for which we can can explicitly compute values of the exponential map. We do not expect our method to outperform the other two in every case because these geometries are very smooth and therefore represent cases where the inherent assumptions of the Dijkstra based methods are very well fulfilled.

For the sphere we can easily determine the correct value at every vertex. For the two dimensional Gauss function we use the fact that it is rotationally symmetric and therefore directly gives the angular part of the exponential map when using the maximum as origin. The distance can be computed by numerical integration along a one dimensional vertical slice of the object. The saddle mesh is defined in terms of sine functions. By solving a differential equation with appropriate boundary conditions we can trace geodesics across the surface. For these geodesics the angle is constant and the distance can be easily determined by measuring arc length. We numerically solve the differential equations using Mathematica and achieve a residual error below 10^{-6}. We project points sampled from a set of geodesics, starting at the map origin, onto the mesh and compare values by linear interpolation across the mesh triangles.

To investigate the influence of mesh resolution we compare results for different mean edge lengths. For each method and mesh we measure the error in the distance and angle component (E_d and E_a, respectively) of the exponential map and report averaged values in Table 1. Figure 15 shows error plots for the sphere and Gauss meshes as well as the resulting map. For the saddle mesh only the maps are shown because the reference solution is only defined along a discrete set of geodesics.

Our method tends to yield better results if the tessellation is relatively fine. While the average error is in most cases smaller for our method compared to DEM, both in terms of distance and angle, the DGPC still produces better results in many cases. In terms of visual quality our result matches those of DGPC on the test meshes.

The influence of tessellation quality on accuracy is illustrated in Figure 13. The first mesh is a very regular triangulation of the plane and serves as reference solution. For the second mesh 50% of the edges have been randomly flipped resulting in non-regular and nearly degenerate triangles. This introduces artifacts in the exponential map but the overall quality is preserved, even at some distance from the center. The errors in distance are 0.72×10^{-2} and 0.205×10^{-2} in angle as compared to 0.91×10^{-3} and 0.9×10^{-3} respectively for the regular mesh. For an anisotropic mesh, generated by anisotropic scaling, we see significant distortion close to the center, however, at some distance accuracy can be recovered. The errors for distance and angle are 0.77×10^{-2} and 0.23×10^{-2}. Since these meshes are planar most method based on exact polyhedral geodesics or Dijkstra based propagation will yield exact result. Our algorithm is useful whenever performance and smoothness is more important than robustness against very irregular meshes.

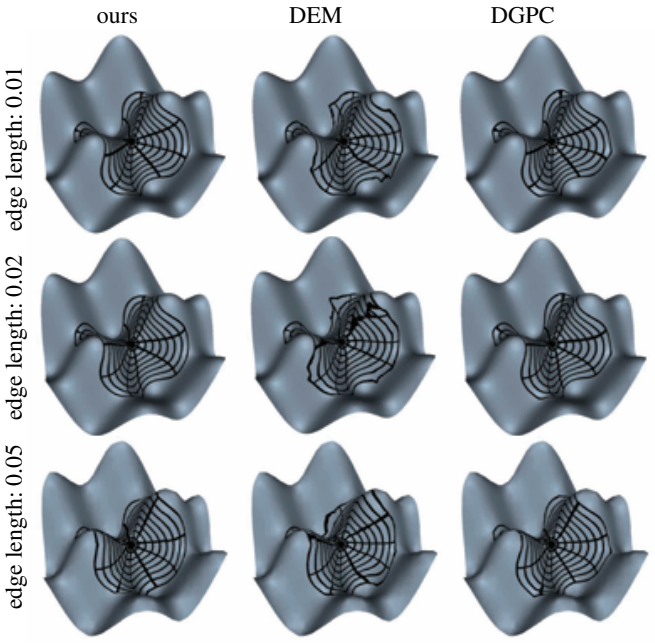

Figure 12: *Exponential maps at a saddle point for meshes of different resolution.*

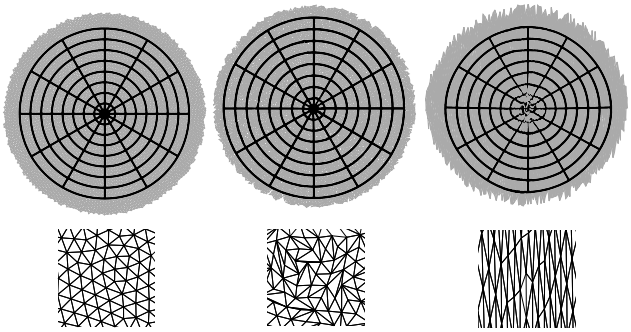

Figure 13: *Influence of mesh quality on accuracy. Our method performs well also for irregular (center) and anisotropically scaled meshes (right).*

8. Conclusion

We presented an algorithm to generate smoothed exponential maps on discrete surfaces. The method is based on discrete heat diffusion and localized solutions of linear systems. The most important observation is that exploiting linear solvers that provide localization and using system factorization as a precomputation step, it is possible to improve computation speed over simple traversal algorithms. As computing angles of the paths is similar to solving a transport problem along the path this observation could be very useful in a wide range of geometry processing algorithms.

While we believe the local parameterizations our approach provides are improving on earlier approaches also in terms of quality, it would be great to provide some guarantees in terms of the quality of the mapping. Given the complexity of current algorithms that provide such guarantees it may be doubtful that it is possible to obtain mappings with guarantees at interactive rates. The option to increase the diffusion time seems to be a good compromise in this direction.

References

[CH90] CHEN J., HAN Y.: Shortest paths on a polyhedron. In *Proceedings of the Sixth Annual Symposium on Computational Geometry* (New York, NY, USA, 1990), SCG '90, ACM, pp. 360–369. URL: http://doi.acm.org/10.1145/98524.98601, doi:10.1145/98524.98601. 2

[CWW13] CRANE K., WEISCHEDEL C., WARDETZKY M.: Geodesics in heat: A new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG) 32*, 5 (2013), 152. doi:10.1145/2516971.2516977. 1, 2, 3, 4, 5, 7, 10

[Dav11] DAVIS T. A.: *User Guide for LDL, a concise sparse Cholesky package*. Tech. rep., 2011. 4

[DMA02] DESBRUN M., MEYER M., ALLIEZ P.: Intrinsic parameterizations of surface meshes. In *Computer graphics forum* (2002), vol. 21, Wiley Online Library, pp. 209–218. doi:10.1111/1467-8659.00580. 7

[FSBS06] FISHER M., SPRINGBORN B., BOBENKO A. I., SCHRODER P.: An algorithm for the construction of intrinsic delaunay triangulations with applications to digital geometry processing. In *ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), ACM, pp. 69–74. doi:10.1145/1185657.1185668. 5

[HDA17] HERHOLZ P., DAVIS T. A., ALEXA M.: Localized solutions of sparse linear systems for geometry processing. *ACM Trans. Graph. 36*, 6 (Nov. 2017), 183:1–183:8. doi:10.1145/3130800.3130849. 2, 4, 5, 6

[HHA17] HERHOLZ P., HAASE F., ALEXA M.: Diffusion diagrams: Voronoi cells and centroids from diffusion. *Computer Graphics Forum 36*, 2 (2017), 163–175. doi:http://dx.doi.org/10.1111/cgf.13116. 5

[Law10] LAWLER G.: *Random Walk and the Heat Equation*. Student mathematical library. American Mathematical Society, 2010. doi:10.1090/stml/055. 7

[LXFH15] LIU Y.-J., XU C.-X., FAN D., HE Y.: Efficient construction and simplification of delaunay meshes. *ACM Trans. Graph. 34*, 6 (2015), 174:1–174:13. doi:10.1145/2816795.2818076. 5

[MBBV15] MASCI J., BOSCAINI D., BRONSTEIN M. M., VANDERGHEYNST P.: Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the 2015 IEEE International Conference on Computer Vision Workshop (ICCVW)* (Washington, DC, USA, 2015), ICCVW '15, IEEE Computer Society, pp. 832–840. URL: http://dx.doi.org/10.1109/ICCVW.2015.112, doi:10.1109/ICCVW.2015.112. 2

[MDSB03] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III* (Berlin, Heidelberg, 2003), Hege H.-C., Polthier K., (Eds.), Springer, pp. 35–57. doi:10.1007/978-3-662-05105-4_2. 3

[MMP87] MITCHELL J. S. B., MOUNT D. M., PAPADIMITRIOU C. H.: The discrete geodesic problem. *SIAM J. Comput. 16*, 4 (aug 1987), 647–668. URL: http://dx.doi.org/10.1137/0216045, doi:10.1137/0216045. 2

[MR12] MELVÆR E. L., REIMERS M.: Geodesic polar coordinates on polygonal meshes. *Computer Graphics Forum 31*, 8 (2012), 2423–2435. doi:10.1111/j.1467-8659.2012.03187.x. 2, 6, 7

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experim. Math. 2* (1993), 15–36. 3

[Sch10] SCHMIDT R.: *Part-Based Representation and Editing of 3D Surface Models*. PhD thesis, University of Toronto, Canada, 2010. 2, 5

[Sch13] SCHMIDT R.: Stroke parameterization. *Computer Graphics Forum 32*, 2pt2 (2013), 255–263. doi:10.1111/cgf.12045. 1

[SGW06] SCHMIDT R., GRIMM C., WYVILL B.: Interactive decal compositing with discrete exponential maps. *ACM Trans. Graph. 25*, 3 (July 2006), 605–613. doi:10.1145/1141911.1141930. 1, 2, 6, 7

[SSC18] SHARP N., SOLIMAN Y., CRANE K.: The vector heat method. *CoRR abs/1805.09170* (2018). URL: http://arxiv.org/abs/1805.09170, arXiv:1805.09170. 2

[SSK*05] SURAZHSKY V., SURAZHSKY T., KIRSANOV D., GORTLER S. J., HOPPE H.: Fast exact and approximate geodesics on meshes. In *ACM transactions on graphics (TOG)* (2005), vol. 24, ACM, pp. 553–560. doi:10.1145/1073204.1073228. 2, 7

[SZZ*13] SUN Q., ZHANG L., ZHANG M., YING X., XIN S.-Q., XIA J., HE Y.: Texture brush: An interactive surface texturing interface. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2013), I3D '13, ACM, pp. 153–160. URL: http://doi.acm.org/10.1145/2448196.2448221, doi:10.1145/2448196.2448221. 2

[TSS*11] TAKAYAMA K., SCHMIDT R., SINGH K., IGARASHI T., BOUBEKEUR T., SORKINE O.: Geobrush: Interactive mesh geometry cloning. *Computer Graphics Forum (proceedings of EUROGRAPHICS) 30*, 2 (2011), 613–622. doi:10.1111/j.1467-8659.2011.01883.x. 1

[Var67] VARADHAN S. R. S.: On the behavior of the fundamental solution of the heat equation with variable coefficients. *Communications on Pure and Applied Mathematics 20*, 2 (1967), 431–455. doi:10.1002/cpa.3160200210. 3

[WFW*17] WANG X., FANG Z., WU J., XIN S.-Q., HE Y.: Discrete geodesic graph (dgg) for computing geodesic distances on polyhedral surfaces. *Comput. Aided Geom. Des. 52*, C (2017), 262–284. URL: https://doi.org/10.1016/j.cagd.2017.03.010, doi:10.1016/j.cagd.2017.03.010. 2

[XW09] XIN S.-Q., WANG G.-J.: Improving chen and han's algorithm on the discrete geodesic problem. *ACM Trans. Graph. 28*, 4 (Sept. 2009), 104:1–104:8. URL: http://doi.acm.org/10.1145/1559755.1559761, doi:10.1145/1559755.1559761. 2

[XYH11] XIN S.-Q., YING X., HE Y.: Efficiently computing geodesic offsets on triangle meshes by the extended xin-wang algorithm. *Comput. Aided Des. 43*, 11 (Nov. 2011), 1468–1476. URL: http://dx.doi.org/10.1016/j.cad.2011.08.027, doi:10.1016/j.cad.2011.08.027. 2

[YWH13] YING X., WANG X., HE Y.: Saddle vertex graph (svg): A novel solution to the discrete geodesic problem. *ACM Trans. Graph. 32*, 6 (nov 2013), 170:1–170:12. URL: http://doi.acm.org/10.1145/2508363.2508379, doi:10.1145/2508363.2508379. 2

		Sphere			Gauss			Saddle		
		0.001	0.002	0.005	0.001	0.002	0.005	0.001	0.002	0.005
distance	heat	$0.612 \cdot 10^{-3}$	$0.211 \cdot 10^{-2}$	$0.174 \cdot 10^{-2}$	$0.593 \cdot 10^{-3}$	$0.584 \cdot 10^{-3}$	$0.142 \cdot 10^{-2}$	$0.324 \cdot 10^{-3}$	$0.928 \cdot 10^{-3}$	$0.173 \cdot 10^{-2}$
	DEM	$0.125 \cdot 10^{-2}$	$0.148 \cdot 10^{-2}$	$0.912 \cdot 10^{-3}$	$0.453 \cdot 10^{-2}$	$0.364 \cdot 10^{-2}$	$0.863 \cdot 10^{-2}$	$0.287 \cdot 10^{-3}$	$0.963 \cdot 10^{-3}$	$0.197 \cdot 10^{-2}$
	DGPC	$0.9 \cdot 10^{-4}$	$0.174 \cdot 10^{-3}$	$0.279 \cdot 10^{-3}$	$0.523 \cdot 10^{-3}$	$0.221 \cdot 10^{-3}$	$0.138 \cdot 10^{-2}$	$0.524 \cdot 10^{-4}$	$0.135 \cdot 10^{-3}$	$0.440 \cdot 10^{-3}$
angle	heat	$0.186 \cdot 10^{-3}$	$0.255 \cdot 10^{-3}$	$0.107 \cdot 10^{-2}$	$0.212 \cdot 10^{-3}$	$0.629 \cdot 10^{-3}$	$0.461 \cdot 10^{-2}$	$0.384 \cdot 10^{-2}$	$0.848 \cdot 10^{-2}$	$0.187 \cdot 10^{-1}$
	DEM	$0.325 \cdot 10^{-3}$	$0.538 \cdot 10^{-3}$	$0.138 \cdot 10^{-3}$	$0.129 \cdot 10^{-2}$	$0.168 \cdot 10^{-2}$	$0.829 \cdot 10^{-2}$	$0.323 \cdot 10^{-1}$	$0.376 \cdot 10^{-1}$	$0.402 \cdot 10^{-1}$
	DGPC	$0.409 \cdot 10^{-4}$	$0.871 \cdot 10^{-4}$	$0.21 \cdot 10^{-3}$	$0.179 \cdot 10^{-3}$	$0.408 \cdot 10^{-4}$	$0.168 \cdot 10^{-2}$	$0.111 \cdot 10^{-2}$	$0.314 \cdot 10^{-2}$	$0.106 \cdot 10^{-1}$

Table 1: *Accuracy measurements for the experiments illustrated in Figures 15 and 12. Average errors in the angle and distance component with respect to reference solutions are reported for different mesh resolutions.*

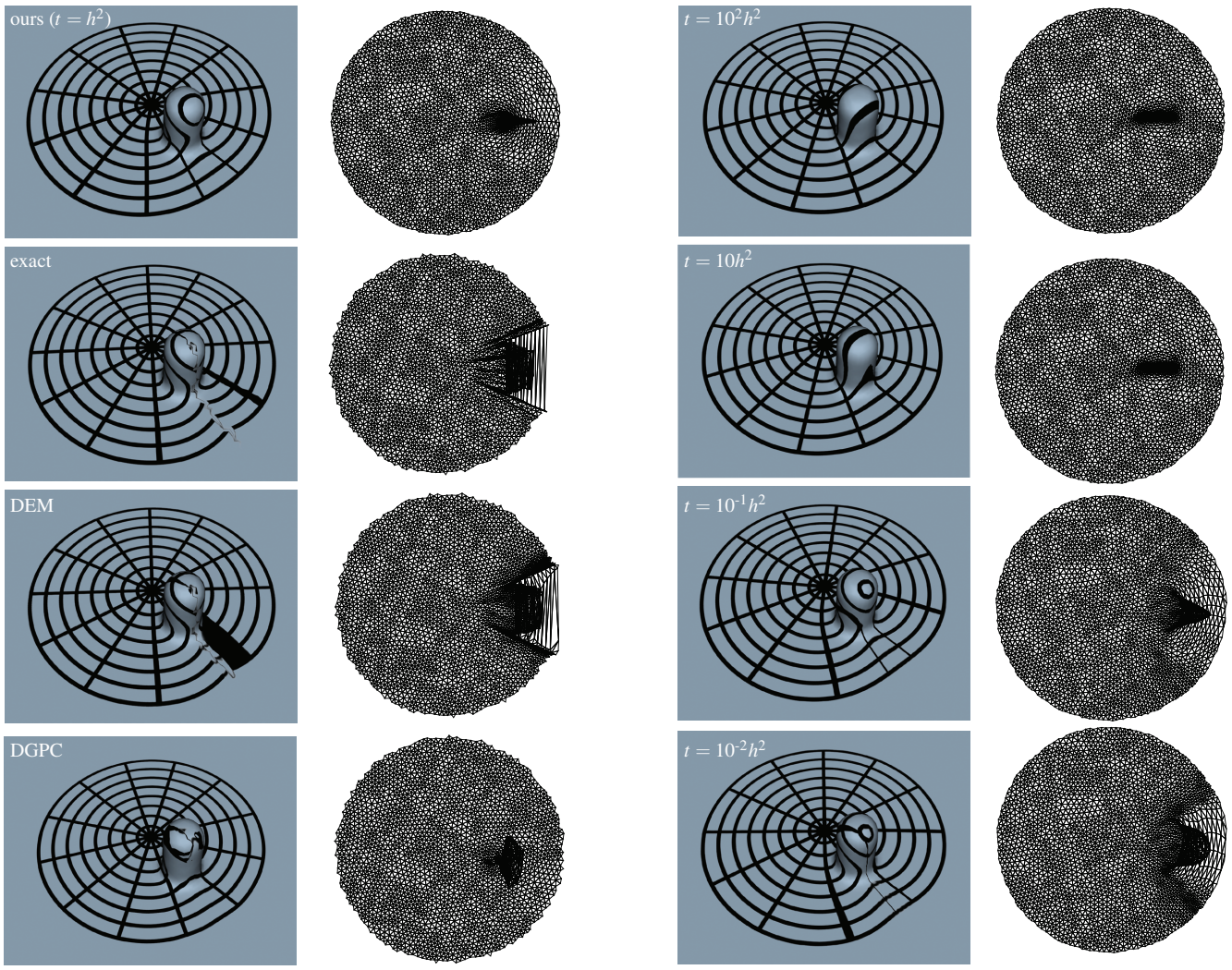

Figure 14: *Exponential maps in a challenging case including parts of the cut locus. Left: Our method compared to exact polyhedral geodesics, DEM and DGPC. Right: By varying the time step t of the diffusion problem the smoothness of the map can be influenced. As default choice we use the squared mean edge length h^2 as suggested by Crane et al. [CWW13]. For smaller time steps the exponential maps become similar to the result using exact polyhedral geodesics but are smoother.*

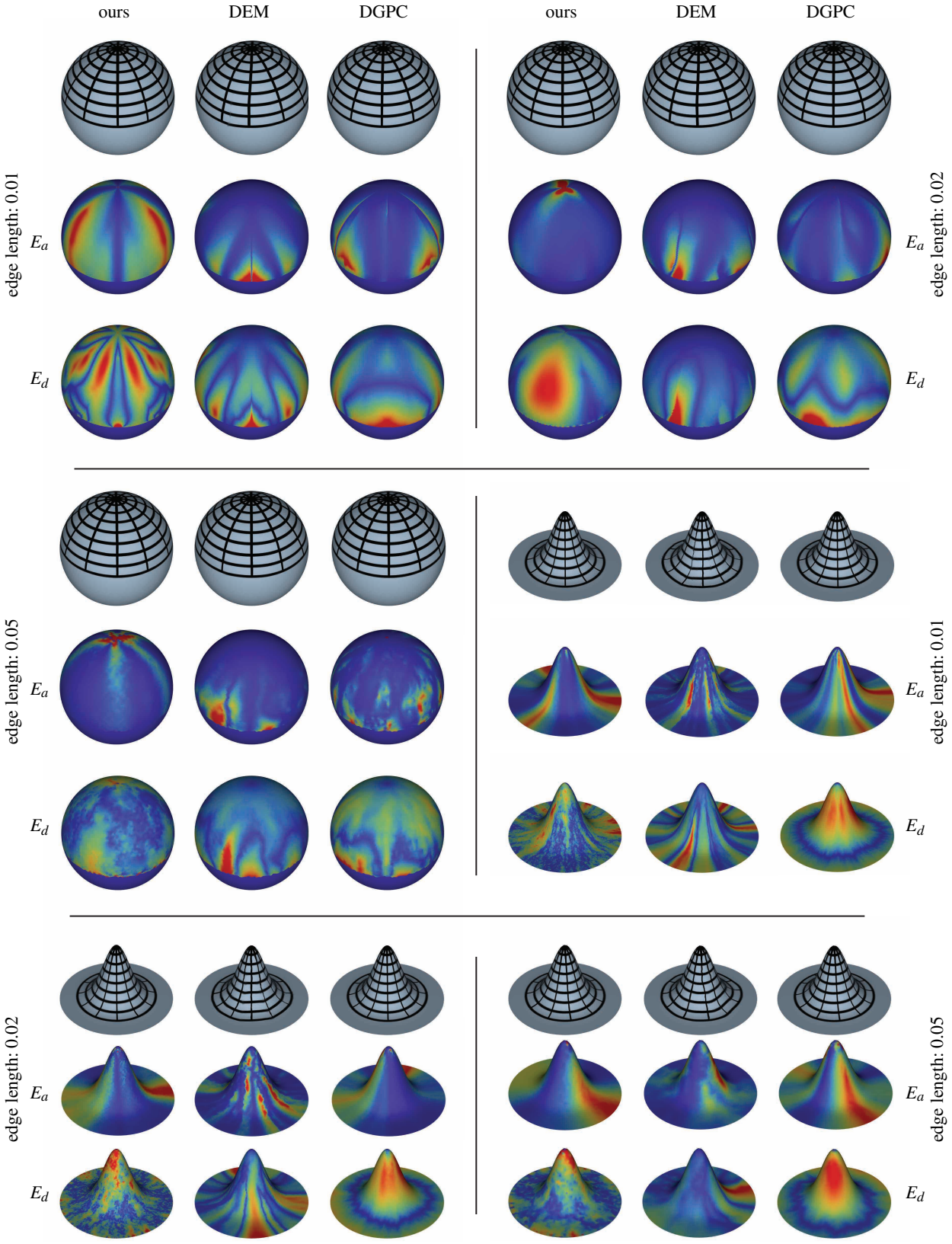

Figure 15: *Comparing our method, exact polyhedral geodesics, DGPC and discrete exponential maps on basic geometries. The map and errors in angle (E_a) and distance (E_d) are visualized for different mesh resolutions.*