Geodesics in Heat

KEENAN CRANE
Caltech
CLARISSE WEISCHEDEL, MAX WARDETZKY
University of Göttingen

We introduce the *heat method* for computing the shortest geodesic distance to a specified subset (*e.g.*, point or curve) of a given domain. The heat method is robust, efficient, and simple to implement since it is based on solving a pair of standard linear elliptic problems. The method represents a significant breakthrough in the practical computation of distance on a wide variety of geometric domains, since the resulting linear systems can be prefactored once and subsequently solved in near-linear time. In practice, distance can be updated via the heat method an order of magnitude faster than with state-of-the-art methods while maintaining a comparable level of accuracy. We provide numerical evidence that the method converges to the exact geodesic distance in the limit of refinement; we also explore smoothed approximations of distance suitable for applications where more regularity is required.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Geometric algorithms, languages, and systems*

General Terms: digital geometry processing, discrete differential geometry, geodesic distance, distance transform, heat kernel

∎

1. INTRODUCTION

Imagine touching a scorching hot needle to a single point on a surface. Over time heat spreads out over the rest of the domain and can be described by a function $k_{t,x}(y)$ called the *heat kernel*, which measures the heat transferred from a source x to a destination y after time t. A well-known relationship between heat and distance is Varadhan's formula [1967], which says that the geodesic distance ϕ between any pair of points x, y on a Riemannian manifold can be recovered via a simple pointwise transformation of the heat kernel:

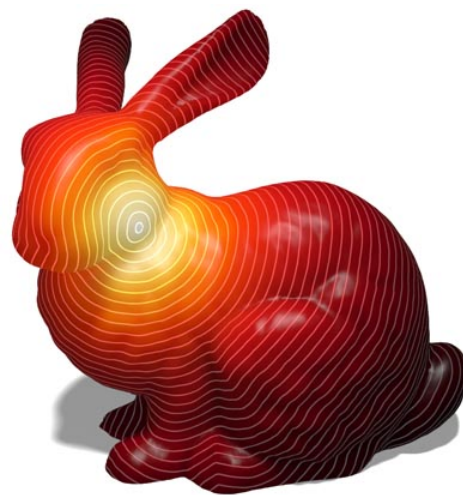$$\phi(x, y) = \lim_{t \to 0} \sqrt{-4t \log k_{t,x}(y)}. \qquad (1)$$

Fig. 1. Geodesic distance on the Stanford Bunny. The heat method allows distance to be rapidly updated for new source points or curves.

The intuition behind this behavior stems from the fact that heat diffusion can be modeled as a large collection of hot particles taking random walks starting at x: any particle that reaches a distant point y after a small time t has had little time to deviate from the shortest possible path. To date, however, this relationship has not been exploited by numerical algorithms that compute geodesic distance.

Why has Varadhan's formula been overlooked in this context? The main reason, perhaps, is that it requires a precise numerical reconstruction of the heat kernel, which is difficult to obtain for small values of t – applying the formula to a mere approximation of $k_{t,x}$ does not yield the correct result, as illustrated in Figures 2 and 6. The main idea behind the heat method is to circumvent this issue by working with a broader class of inputs, namely any function whose gradient is parallel to geodesics. We can then separate the computation of distance into two separate stages: first compute the gradient of the distance field, then recover the distance itself.

Relative to existing algorithms, the heat method offers two major advantages. First, it can be applied to virtually any type of geometric discretization, including regular and irregular grids, polygonal meshes, and even unstructured point clouds. Second, it involves only the solution of sparse linear systems, which can be prefactored once and rapidly re-solved many times. This feature makes the heat method particularly valuable for applications such as shape matching, path planning, and level set-based simulation (*e.g.*, free-surface fluid flows), which require repeated distance queries on a fixed geometric domain. Moreover, because linear elliptic equations are widespread in scientific computing, the heat method can immediately take advantage of new developments in numerical linear algebra and parallelization.
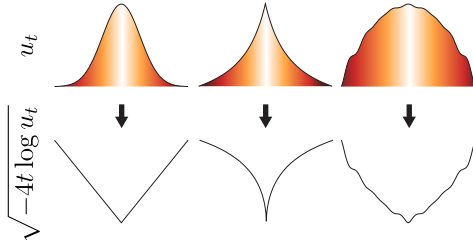
Fig. 2. Given an *exact* reconstruction of the heat kernel *(top left)* Varadhan's formula can be used to recover geodesic distance *(bottom left)* but fails in the presence of approximation or numerical error *(middle, right)*, as shown here for a point source in 1D. The robustness of the heat method stems from the fact that it depends only on the *direction* of the gradient.

2. RELATED WORK

The prevailing approach to distance computation is to solve the *eikonal equation*

$$|\nabla \phi| = 1 \qquad (2)$$

subject to boundary conditions $\phi|_\gamma = 0$ over some subset γ of the domain. This formulation is *nonlinear* and *hyperbolic*, making it difficult to solve directly. Typically one applies an iterative relaxation scheme such as Gauss-Seidel – special update orders are known as *fast marching* and *fast sweeping*, which are some of the most popular algorithms for distance computation on regular grids [Sethian 1996] and triangulated surfaces [Kimmel and Sethian 1998]. These algorithms can also be used on implicit surfaces [Memoli and Sapiro 2001], point clouds [Memoli and Sapiro 2005], and polygon soup [Campen and Kobbelt 2011], but only indirectly: distance is computed on a simplicial mesh or regular grid that approximates the original domain. Implementation of fast marching on simplicial grids is challenging due to the need for nonobtuse triangulations (which are notoriously difficult to obtain) or else a complex unfolding procedure to preserve monotonicity of the solution; moreover these issues are not well-studied in dimensions greater than two. Fast marching and fast sweeping have asymptotic complexity of $O(n \log n)$ and $O(n)$, respectively, but sweeping is often slower due to the large number of sweeps required to obtain accurate results [Hysing and Turek 2005].

The main drawback of these methods is that they do not reuse information: the distance to different subsets γ must be computed entirely from scratch each time. Also note that both sweeping and marching present challenges for parallelization: priority queues are inherently serial, and irregular meshes lack a natural sweeping order. Weber *et al.* [2008] address this issue by decomposing surfaces into regular grids, but this decomposition resamples the surface and requires a low-distortion parameterization that may be difficult to obtain (note that the heat method would also benefit from such a decomposition).

In a different development, Mitchell *et al.* [1987] give an $O(n^2 \log n)$ algorithm for computing the exact polyhedral distance from a single source to all other vertices of a triangulated surface. Surazhsky *et al.* [2005] demonstrate that this algorithm tends to run in sub-quadratic time in practice, and present an approximate $O(n \log n)$ version of the algorithm with guaranteed error bounds; Bommes and Kobbelt [2007] extend the algorithm to polygonal sources. Similar to fast marching, these algorithms propagate distance information in wavefront order using a priority queue, again making them difficult to parallelize. More importantly, the amor-
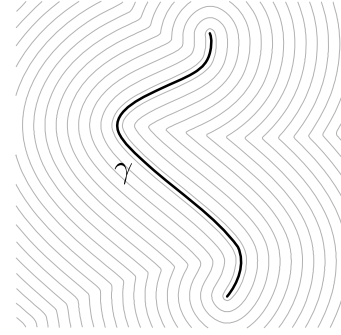

Fig. 3. The heat method computes the shortest distance to a subset γ of a given domain. Gray curves indicate isolines of the distance function.

tized cost of these algorithms (over many different source subsets γ) is substantially greater than for the heat method since they do not reuse information from one subset to the next. Finally, although [Surazhsky et al. 2005] greatly simplifies the original formulation, these algorithms remain challenging to implement and do not immediately generalize to domains other than triangle meshes.

Closest to our approach is the recent method of Rangarajan and Gurumoorthy [2011], who do not appear to be aware of Varadahn's formula – they instead derive an analogous relationship $\phi = -\sqrt{\hbar \log \psi}$ between the distance function and solutions ψ to the time-independent Schrödinger equation. We emphasize, however, that this derivation applies only in $\mathbb{R}^n$ where ψ takes a special form – in this case it may be just as easy to analytically invert the Euclidean heat kernel $u_{t,x} = (4\pi t)^{-n/2} e^{-\phi(x,y)^2/4t}$. Moreover, they compute solutions using the fast Fourier transform, which limits its computation to regular grids. To obtain accurate results their method requires either the use of arbitrary-precision arithmetic or a combination of multiple solutions for various values of $\hbar$; no general guidance is provided for determining appropriate values of $\hbar$.

Finally, there is a large literature on *smooth distances* [Coifman and Lafon 2006; Fouss et al. 2007; Lipman et al. 2010], which are valuable in contexts where differentiability is required. However, existing smooth distances may not be appropriate in contexts where the *geometry* of the original domain is important, since they do not attempt to approximate the original metric and therefore substantially violate the unit-speed nature of geodesics (Figure 10). Interestingly enough, these distances also have an interpretation in terms of simple discretizations of heat flow – see Section 3.3 for further discussion.

Fig. 4. Distance to the boundary on a region in the plane (left) or a surface in $\mathbb{R}^3$ is achieved by simply placing heat along the boundary curve. Note good recovery of the *cut locus*, *i.e.*, points with more than one closest point on the boundary.
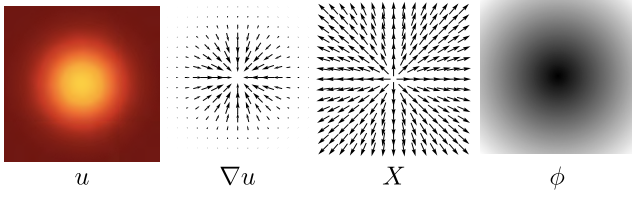
u ∇u X ϕ

Fig. 5. Outline of the heat method. (I) Heat u is allowed to diffuse for a brief period of time *(left)*. (II) The temperature gradient ∇u *(center left)* is normalized and negated to get a unit vector field X *(center right)* pointing along geodesics. (III) A function ϕ whose gradient follows X recovers the final distance *(right)*.

3. THE HEAT METHOD

Our method can be described purely in terms of operations on smooth manifolds; we explore spatial and temporal discretization in Sections 3.1 and 3.2, respectively. Let Δ be the negative-semidefinite Laplace–Beltrami operator acting on (weakly) differentiable real-valued functions over a Riemannian manifold (M, g). The heat method consists of three basic steps:

Algorithm 1 The Heat Method

  I. Integrate the heat flow $\dot{u} = \Delta u$ for some fixed time t.
 II. Evaluate the vector field $X = -\nabla u/|\nabla u|$.
III. Solve the Poisson equation $\Delta \phi = \nabla \cdot X$.

The function ϕ approximates geodesic distance, approaching the true distance as t goes to zero (Eq. (1)). Note that the solution to step III is unique only up to an additive constant – final values simply need to be shifted such that the smallest distance is zero. Initial conditions $u_0 = \delta(x)$ (*i.e.*, a Dirac delta) recover the distance to a single source point $x \in M$ as in Figure 1, but in general we can compute the distance to any piecewise submanifold γ by setting u_0 to a generalized Dirac [Villa 2006] over γ (see Figures 3 and 4).

The heat method can be motivated as follows. Consider an approximation u_t of heat flow for a fixed time t. Unless u_t exhibits *precisely* the right rate of decay, Varadhan's transformation $u_t \mapsto \sqrt{-4t \log u_t}$ will yield a poor approximation of the true geodesic distance ϕ because it is highly sensitive to errors in magnitude (see Figures 2 and 6). The heat method asks for something different: it asks only that the gradient ∇u_t points in the right direction, *i.e.*, parallel to $\nabla \phi$. Magnitude can safely be ignored since we know (from the eikonal equation) that the gradient of the true distance function has unit length. We therefore compute the normalized gradient field $X = -\nabla u/|\nabla u|$ and find the closest scalar potential ϕ by minimizing $\int_M |\nabla \phi - X|^2$, or equivalently, by solving the corresponding Euler-Lagrange equations $\Delta \phi = \nabla \cdot X$ [Schwarz 1995]. The overall procedure is depicted in Figure 5.

3.1 Time Discretization

We discretize the heat equation from step I of Algorithm 1 in time using a single backward Euler step for some fixed time t. In practice, this means we simply solve the linear equation

$$(\mathrm{id} - t\Delta)u_t = u_0 \qquad (3)$$

over the entire domain M, where id is the identity (here we still consider a smooth manifold; spatial discretization is discussed in

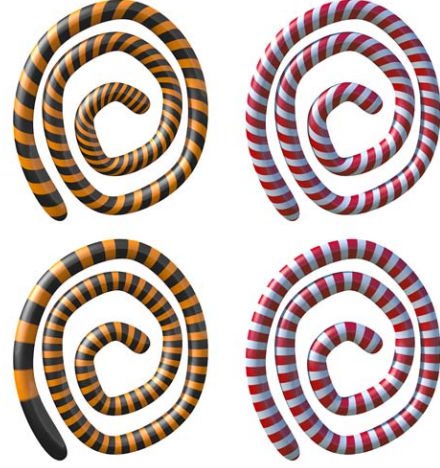Fig. 6. *Left:* Varadhan's formula. *Right:* the heat method. Even for very small values of t, simply applying Varadhan's formula does not provide an accurate approximation of geodesic distance *(top left)*; for large values of t spacing becomes even more uneven *(bottom left)*. Normalizing the gradient results in a more accurate solution, as indicated by evenly spaced isolines *(top right)*, and is also valuable when constructing a smoothed distance function *(bottom right)*.

Section 3.2). Note that backward Euler provides a maximum principle, preventing spurious oscillations in the solution [Wade et al. 2005]. We can get a better understanding of solutions to Eq. (3) by considering the elliptic boundary value problem

$$\begin{aligned} (\mathrm{id} - t\Delta)v_t = 0 &\quad \text{on} \quad M\backslash\gamma \\ v_t = 1 &\quad \text{on} \quad \gamma \,. \end{aligned} \qquad (4)$$

which for a point source yields a solution v_t equal to u_t up to a multiplicative constant. As established by Varadhan in his proof of Eq. (1), v_t also has a close relationship with distance, namely

$$\lim_{t \to 0} -\frac{\sqrt{t}}{2} \log v_t = \phi. \qquad (5)$$

This relationship ensures the validity of steps II and III since the transformation applied to v_t preserves the direction of the gradient.

3.2 Spatial Discretization

In principle the heat method can be applied to any domain with a discrete gradient (∇), divergence ($\nabla\cdot$) and Laplace operator (Δ). Here we investigate several possible discretizations.

3.2.1 *Simplicial Meshes.* Let $u \in \mathbb{R}^{|V|}$ specify a piecewise linear function on a triangulated surface. A standard discretization of the Laplacian at a vertex i is given by

$$(Lu)_i = \frac{1}{2A_i} \sum_j (\cot \alpha_{ij} + \cot \beta_{ij})(u_j - u_i),$$

where A_i is one third the area of all triangles incident on vertex i, the sum is taken over all neighboring vertices j, and α_{ij}, β_{ij} are the angles opposing the corresponding edge [Mac-Neal 1949]. We can express this operation via a matrix $L = A^{-1}L_C$, where $A \in \mathbb{R}^{|V| \times |V|}$ is a diagonal matrix containing the vertex areas and $L_C \in \mathbb{R}^{|V| \times |V|}$
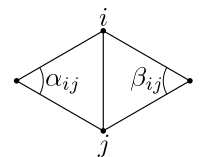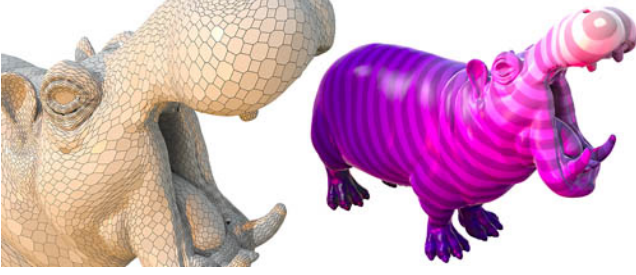
Fig. 7. Since the heat method is based on well-established discrete operators like the Laplacian, it is easy to adapt to a variety of geometric domains. Above: distance on a hippo composed of high-degree nonplanar (and sometimes nonconvex) polygonal faces.

is the *cotan operator* representing the remaining sum. Heat flow can then be computed by solving the symmetric positive-definite system

$$(A - tL_C)u = u_0$$

where u_0 is a Kronecker delta over γ (*i.e.*, one for source vertices; zero otherwise). The gradient in a given triangle can be expressed succinctly as

$$\nabla u = \frac{1}{2A_f} \sum_i u_i(N \times e_i)$$
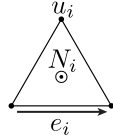
where A_f is the area of the face, N is its unit normal, e_i is the ith edge vector (oriented counterclockwise), and u_i is the value of u at the opposing vertex. The integrated divergence associated with vertex i can be written as
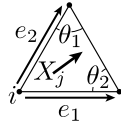
$$\nabla \cdot X = \frac{1}{2} \sum_j \cot \theta_1 (e_1 \cdot X_j) + \cot \theta_2 (e_2 \cdot X_j)$$

where the sum is taken over incident triangles j each with a vector X_j, e_1 and e_2 are the two edge vectors of triangle j containing i, and θ_1, θ_2 are the opposing angles. If we let $d \in \mathbb{R}^{|V|}$ be the vector of (integrated) divergences of the normalized vector field X, then the final distance function is computed by solving the symmetric Poisson problem

$$L_C \phi = d.$$

Conveniently, this discretization easily generalizes to higher dimensions (*e.g.*, tetrahedral meshes) using well-established discrete operators; see for instance [Desbrun et al. 2008].

3.2.2 *Polygonal Surfaces.* For a mesh with (not necessarily planar) polygonal faces, we use the polygonal Laplacian defined by Alexa and Wardetzky [2011]. The only difference in this setting is that the gradient of the heat kernel is expressed as a discrete 1-form associated with half edges, hence we cannot directly evaluate the magnitude of the gradient $|\nabla u|$ needed for the normalization step (Algorithm 1, step II). To resolve this issue we assume that ∇u is constant over each face, implying that

$$u_f^T L_f u_f = \int_M |\nabla u|^2 dA = |\nabla u|^2 A_f,$$

where u_f is the vector of heat values in face f, A_f is the magnitude of the area vector, and L_f is the local (weak) Laplacian. We can
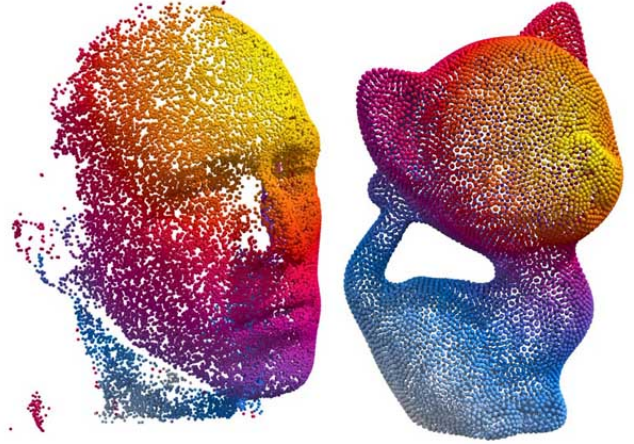

Fig. 8. The heat method can be applied directly to point clouds that lack connectivity information. *Left:* face scan with holes and noise. *Right:* kitten surface with connectivity removed. Yellow points are close to the source; disconnected clusters (in the sense of Liu *et al.*) receive a constant red value.

therefore approximate the magnitude of the gradient as

$$|\nabla u|_f = \sqrt{u_f^T L_f u_f / A_f}$$

which is used to normalize the 1-form values in the corresponding face. The integrated divergence is given by $d^T M\alpha$ where α is the normalized gradient, d is the coboundary operator and M is the mass matrix for 1-forms (see [Alexa and Wardetzky 2011] for details). These operators are applied in steps I-III as usual. Figure 7 demonstrates distance computed on an irregular polygonal mesh.

3.2.3 *Point Clouds.* For a discrete collection of point samples $P \subset \mathbb{R}^n$ of M with no connectivity information, we solve the heat equation (step I) using the symmetric *point cloud* Laplacian recently introduced by Liu et al. [2012], which extends previous work of Belkin et al. [2009a]. In this formulation, the Laplacian is represented by $A_\mathcal{V}^{-1} L_{PC}$, where $A_\mathcal{V}$ is a diagonal matrix of approximate Voronoi areas associated with each point, and L_{PC} is a symmetric positive semidefinite matrix (see [Liu et al. 2012], Section 3.4, for details).

To compute the vector field $X = -\nabla u/|\nabla u|$ (step II), we represent the function $u : P \to \mathbb{R}$ as a height function over approximate tangent planes T_p at each point $p \in P$ and evaluate the gradient of a weighted least squares (WLS) approximation of u [Nealen 2004]. To compute tangent planes, we use a moving least squares (MLS) approximation for simplicity – although other choices might be desirable (see Liu *et al.*). The WLS approximation of ∇u also provides a linear mapping $u \mapsto Du$, taking any scalar function u to its gradient. To find the best-fit scalar potential ϕ (step III), we solve the linear, positive-semidefinite Poisson equation $L_{PC}\phi = D^T A_\mathcal{V} X$. The distance resulting from this approach is depicted in Figure 8.

Other discretizations are certainly possible (see for instance [Luo et al. 2009]); we picked one that was simple to implement in any dimension. Note that the computational cost of the heat method depends primarily on the *intrinsic* dimension n of M, whereas methods based on fast marching require a grid of the same dimension m as the ambient space [Memoli and Sapiro 2001] – this distinction is especially important in contexts like machine learning where m may be significantly larger than n.

3.2.4 *Choice of Time Step.* The accuracy of the heat method relies in part on the choice of time step t. In the smooth setting, Eq. (5) suggests that smaller values of t yield better approximations of geodesic distance. In the discrete setting we instead discover a rather remarkable fact, namely that the limit solution to Eq. (4) is purely a function of the *combinatorial* distance, independent of how we discretize the Laplacian (see Appendix A). The main implication of this fact is that – *on a fixed mesh* – decreasing the value of t does not necessarily improve accuracy, even in exact arithmetic. (Note, of course, that we can always improve accuracy by refining the mesh and decreasing t accordingly.) Moreover, increasing the value of t past a certain point produces a smoothed approximation of geodesic distance (Section 3.3). We therefore seek an optimal time step t^* that is neither too large nor too small.

Determining a provably optimal expression for t^* is difficult due to the great complexity of analysis involving the cut locus [Neel and Stroock 2004]. We instead use a simple estimate that works remarkably well in practice, namely $t = mh^2$ where h is the mean spacing between adjacent nodes and $m > 0$ is a constant. This estimate is motivated by the fact that $h^2 \Delta$ is invariant with respect to scale and refinement; experiments on a regular grid (Figure 18) suggest that $m = 1$ is the smallest parameter value that recovers the ℓ_2 distance, and indeed this value yields near-optimal accuracy for a wide variety of irregularly triangulated surfaces, as demonstrated in Figure 22. In this paper the time step

$$\boxed{t = h^2}$$

is therefore used uniformly throughout all tests and examples, except where we explicitly seek a smoothed approximation of distance, as in Section 3.3.

3.3 Smoothed Distance

Geodesic distance fails to be smooth at points in the *cut locus*, *i.e.*, points at which there is no unique shortest path to the source – these points appear as sharp cusps in the level lines of the distance function. Non-smoothness can result in numerical difficulty for applications which need to take derivatives of the distance function ϕ (*e.g.*, level set methods), or may simply be undesirable aesthetically.

Several distances have been designed with smoothness in mind, including diffusion distance [Coifman and Lafon 2006], commute-time distance [Fouss et al. 2007], and biharmonic distance [Lipman et al. 2010] (see the last reference for a more detailed discussion). These distances satisfy a number of important properties (smoothness, isometry-invariance, *etc.*), but are poor approximations of true geodesic distance, as indicated by uneven spacing of isolines (see Figure 10, middle). They can also be expensive to evaluate, requiring either a large number of Laplacian eigenvectors ($\sim 150 - 200$ in practice) or the solution to a linear system at each vertex.
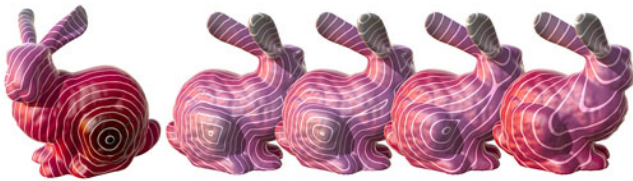

Fig. 9.   A source on the front of the bunny results in nonsmooth cusps on the opposite side. By running heat flow for progressively longer durations t, we obtain smoothed approximations of geodesic distance *(right)*.
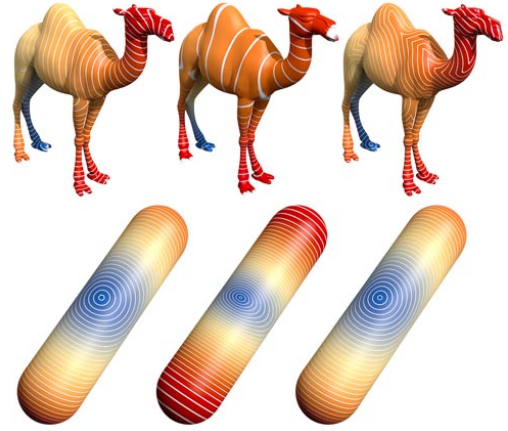

Fig. 10. *Top row:* our smooth approximation of geodesic distance (left) and biharmonic distance (middle) both mitigate sharp "cusps" found in the exact distance (right), but notice that isoline spacing of the biharmonic distance can vary dramatically. Bottom row: biharmonic distance (middle) tends to exhibit elliptical level lines near the source, while our smoothed distance (left) maintains isotropic circular profiles as seen in the exact distance (right).

In contrast, one can rapidly construct smoothed versions of geodesic distance by simply applying the heat method for large values of t (Figure 9). The computational cost remains the same, and isolines are evenly spaced for any value of t due to normalization (step II). Note that the resulting smooth distance function is isometrically (but not conformally) invariant since it depends only on the intrinsic Laplace–Beltrami operator.

Interestingly enough, existing smooth distance functions can also be understood in terms of time-discrete heat flow. In particular, the commute-time distance d_C and biharmonic distance d_B can be expressed in terms of the harmonic and biharmonic Green's functions g_C and g_B:

$$
\begin{aligned}
d_C(x,y)^2 &= g_C(x,x) - 2g_C(x,y) + g_C(y,y), \\
d_B(x,y)^2 &= g_B(x,x) - 2g_B(x,y) + g_B(y,y).
\end{aligned}
$$

On a manifold of constant sectional curvature the sum $g(x,x) + g(y,y)$ is constant, hence the commute-time and biharmonic distances are essentially a scalar multiple of the harmonic and biharmonic Green's functions (respectively), which can be expressed via one- and two-step backward Euler approximations of heat flow:

$$
\begin{aligned}
g_C &= \lim_{t\to\infty}(\mathrm{id} - t\Delta)^\dagger\delta, \\
g_B &= \lim_{t\to\infty}(\mathrm{id} - 2t\Delta + t^2\Delta^2)^\dagger\delta.
\end{aligned}
$$

(Here $\dagger$ denotes the pseudoinverse.) Note that for finite t the identity operator acts as a regularizer, preventing a logarithmic singularity. For spaces with variable curvature, the Green's functions provide only an approximation of the corresponding distance functions.

3.4 Boundary Conditions

If one is interested in the exact distance, either vanishing Neumann or Dirichlet conditions suffice since this choice does not affect the behavior of the smooth limit solution (see [von Renesse 2004], Corollary 2 and [Norris 1997], Theorem 1.1, respectively). Boundary conditions do however alter the behavior of our smoothed geodesic distance (*i.e.*, large t) – Figure 11 illustrates this behavior. Although there is no well-defined "correct" behavior for the

Fig. 11. Effect of Neumann (top-left), Dirichlet (top-right) and averaged (bottom-left) boundary conditions on smoothed distance. Note that averaged conditions mimic the behavior of the same surface without boundary.
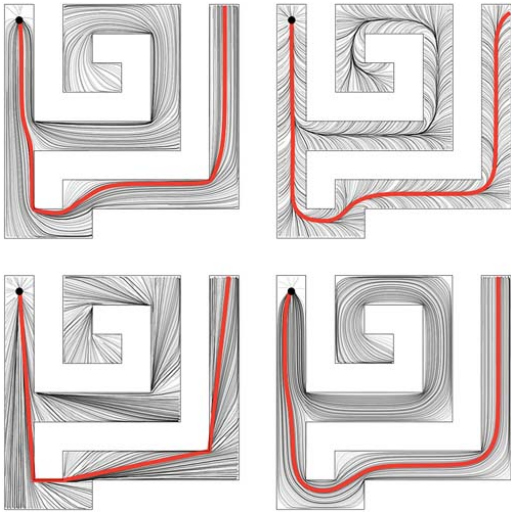

Fig. 12. For path planning, the behavior of geodesics can be controlled via boundary conditions and the integration time t. *Top-left:* Neumann conditions encourage boundary adhesion. *Top-right:* Dirichlet conditions encourage avoidance. *Bottom-left:* small values of t yield standard straight-line geodesics. *Bottom-right:* large values of t yield more natural trajectories.

smoothed solution, we advocate the use of *averaged* boundary conditions obtained as the mean of the Neumann solution u_N and the Dirichlet solution u_D, *i.e.*, $u = \frac{1}{2}(u_N + u_D)$ – these conditions tend to produce isolines that are not substantially influenced by the shape of the boundary. The intuition behind this behavior is again based on interpreting heat diffusion in terms of random walks: zero Dirichlet conditions absorb heat, causing walkers to "fall off" the edge of the domain. Neumann conditions prevent heat from flowing out of the domain, effectively "reflecting" random walkers. Averaged boundary conditions mimic the behavior of a domain without boundary: the number of walkers leaving equals the number of walkers returning. Figure 12 shows how boundary conditions affect the behavior of geodesics in a path-planning scenario.

4. COMPARISON

4.1 Performance

A key advantage of the heat method is that the linear systems in steps (I) and (III) can be prefactored. Our implementation uses sparse Cholesky factorization [Chen et al. 2008], which for Poisson-type problems has guaranteed sub-quadratic complexity but in practice scales even

better [Botsch et al. 2005]; moreover there is strong evidence to suggest that sparse systems arising from elliptic PDEs can be solved in very close to linear time [Schmitz and Ying 2012; Spielman and Teng 2004]. Independent of these issues, the amortized cost for problems with a large number of right-hand sides is roughly linear, since back substitution can be applied in essentially linear time. See inset for a breakdown of relative costs in our implementation.
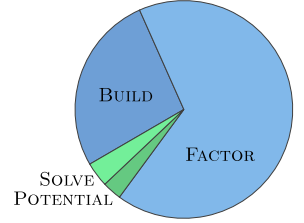
In terms of absolute performance, a number of factors affect the run time of the heat method including the spatial discretization, choice of discrete Laplacian, geometric data structures, and so forth. As a typical example, we compared our simplicial implementation (Section 3.2.1) to the first-order fast marching method of Kimmel & Sethian [1998] and the exact algorithm of Mitchell *et al.* [1987] as described by Surazhsky *et al.* [2005]. In particular we used the state-of-the-art fast marching implementation of Peyré and Cohen [2005] and the exact implementation of Kirsanov [Surazhsky et al. 2005]. The heat method was implemented in ANSI C using a simple vertex-face adjacency list. All timings were taken on a 2.4 GHz Intel Core 2 Duo machine using a single core – Table I gives timing information. Note that even for a single distance computation the heat method outperforms fast marching; more importantly, updating distance via the heat method for new subsets γ is consistently an order of magnitude faster (or more) than both fast marching and the exact algorithm.

4.2 Accuracy

We examined errors in the heat method, fast marching [Kimmel and Sethian 1998], and the exact polyhedral distance [Mitchell et al. 1987], relative to mean edge length h for a variety of triangulated surfaces. Figures 19 and 20 illustrate the rate of convergence on simple geometries where the smooth geodesic distance can be easily obtained. Both fast marching and the heat method appear to exhibit linear convergence; it is also interesting to note that the exact *polyhedral* distance provides only quadratic convergence. Keeping this fact in mind, Table I uses the polyhedral distance as a baseline for comparison on more complicated geometries – here MAX is the maximum error as a percentage of mesh diameter and MIN is the mean relative error at each vertex (a convention introduced in [Surazhsky et al. 2005]). Note that fast marching tends to achieve a smaller maximum error, whereas the heat method does better on average. Figure 14 gives a visual comparison of accuracy; the only notable discrepancy is a slight smoothing at sharp cusps; Figure 15 indicates that this phenomenon does not interfere with the extraction of the cut locus – here we simply threshold the magnitude of $\Delta\phi$. Figure 21 plots the maximum violation of metric properties – both the heat method and fast marching exhibit small approximation errors that vanish under refinement. Even for the smoothed distance ($m >> 1$) the triangle inequality is violated only for highly degenerate geodesic triangles, *i.e.*, all three vertices on a common
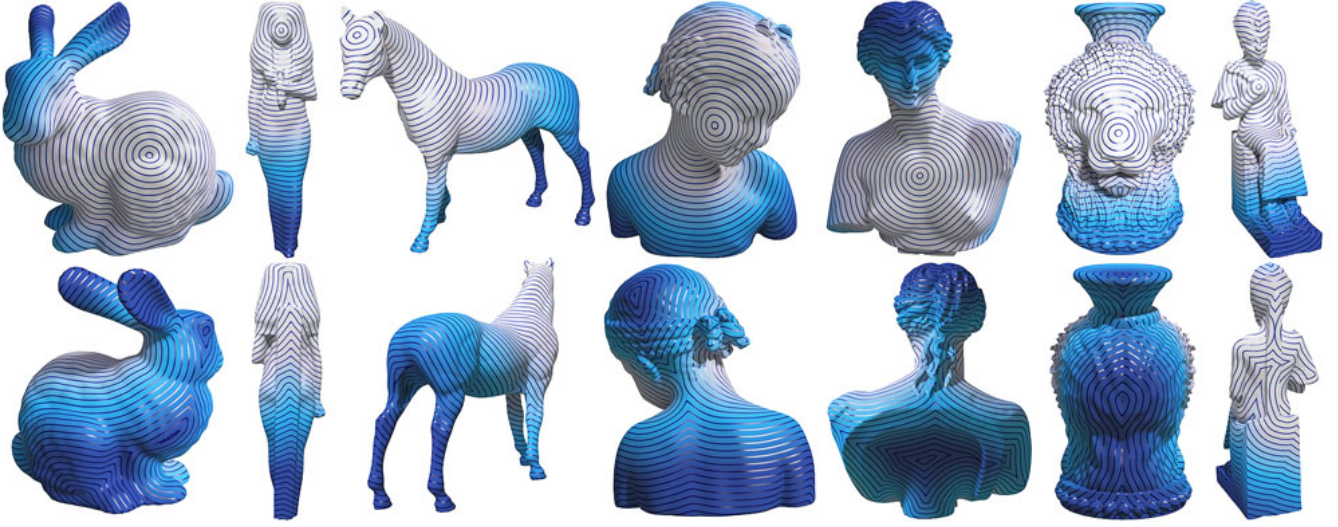
Fig. 13.   Meshes used to test performance and accuracy (see Table I). Left to right: BUNNY, ISIS, HORSE, BIMBA, APHRODITE, LION, RAMSES.

Table I. Comparison with fast marching and exact polyhedral distance. Best speed/accuracy in **bold**; speedup in orange.

| MODEL	TRIANGLES	HEAT METHOD				FAST MARCHING			EXACT
		PRECOMPUTE	SOLVE	MAX ERROR	MEAN ERROR	TIME	MAX ERROR	MEAN ERROR	TIME
BUNNY	28k	**0.21s**	**0.01s (28x)**	3.22%	**1.12%**	0.28s	**1.06%**	1.15%	0.95s
ISIS	93k	**0.73s**	**0.05s (21x)**	1.19%	**0.55%**	1.06s	**0.60%**	0.76%	5.61s
HORSE	96k	**0.74s**	**0.05s (20x)**	1.18%	**0.42%**	1.00s	**0.74%**	0.66%	6.42s
KITTEN	106k	**1.13s**	**0.06s (22x)**	0.78%	**0.43%**	1.29s	**0.47%**	0.55%	11.18s
BIMBA	149k	**1.79s**	**0.09s (29x)**	1.92%	0.73%	2.62s	**0.63%**	**0.69%**	13.55s
APHRODITE	205k	**2.66s**	**0.12s (47x)**	1.20%	**0.46%**	5.58s	**0.58%**	0.59%	25.74s
LION	353k	**5.25s**	**0.24s (24x)**	1.92%	0.84%	10.92s	**0.68%**	**0.67%**	22.33s
RAMSES	1.6M	**63.4s**	**1.45s (68x)**	0.49%	**0.24%**	98.11s	**0.29%**	0.35%	268.87s
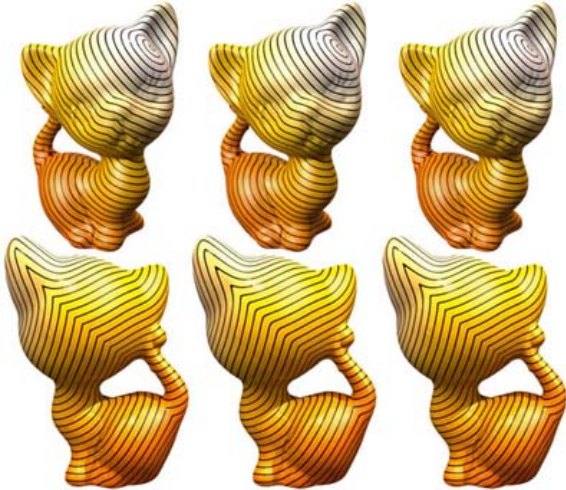
Fig. 14.   Visual comparison of accuracy. *Left:* exact geodesic distance. Using default parameters, the heat method *(middle)* and fast marching (right) both produce results of comparable accuracy, here within less than 1% of the exact distance – see Table I for a more detailed comparison.

geodesic. (In contrast, smoothed distances discussed in Section 2 satisfy metric properties exactly, but cannot be used to obtain the true geometric distance.) Overall, the heat method exhibits errors of the same magnitude and rate of convergence as fast marching (at lower computational cost) and is likely suitable for any application where fast marching is presently used.

The accuracy of the heat method might be further improved by considering alternative spatial discretizations (see for instance [Belkin et al. 2009b; Hildebrandt and Polthier 2011]), though again one should note that even the exact polyhedral distance yields only an $O(h^2)$ approximation. In the case of the fast marching method, accuracy is determined by the choice of *update rule*. A number of highly accurate update rules have been developed in the case of regular grids (*e.g.*, HJ WENO [Jiang and Peng 1997]), but fewer options are available on irregular domains such as triangle meshes, the predominant choice being the first-order update rule of Kimmel and Sethian [1998]. Finally, the approximate algorithm of Surazhsky *et al.* provides an interesting comparison since it is on par with fast marching in terms of performance and produces more accurate results (see [Surazhsky et al. 2005], Table 1). Similar to fast marching, however, it does not take advantage of precomputation and therefore exhibits a significantly higher amortized cost than the heat method; it is also limited to triangle meshes.
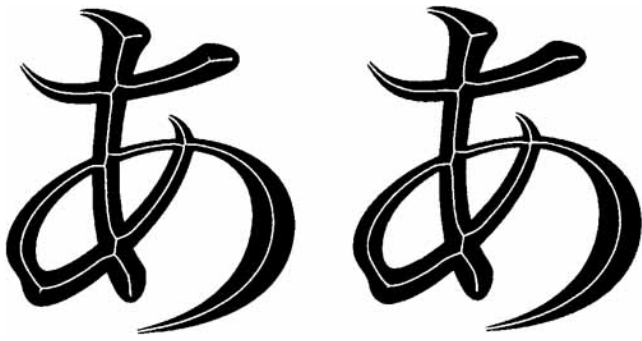
Fig. 15.   Medial axis of the hiragana letter "a" extracted by thresholding second derivatives of the distance to the boundary. Left: fast marching. Right: heat method.

4.3   Robustness

Two factors contribute to the robustness of the heat method, namely (1) the use of an unconditionally stable implicit time-integration scheme and (2) formulation in terms of purely elliptic PDEs. Figure 16 verifies that the heat method continues to work well even on meshes that are poorly discretized or corrupted by a large amount of noise (here modeled as uniform Gaussian noise applied to the vertex coordinates). In this case we use a moderately large value of t to investigate the behavior of our smoothed distance; similar behavior is observed for small t values. Figure 17 illustrates the robustness of the method on a surface with many small holes as well as long sliver triangles.

Fig. 16.   Tests of robustness. *Left:* our smoothed distance (Section 3.3) appears similar on meshes of different resolution. *Right:* even for meshes with severe noise *(top)* we recover a good approximation of the distance function on the original surface *(bottom, visualized on noise-free mesh)*.

Fig. 17.   Smoothed geodesic distance on an extremely poor triangulation with significant noise – note that small holes are essentially ignored. Also note good approximation of distance even along thin slivers in the nose.

5.   CONCLUSION

The heat method is a simple, general method that can be easily incorporated into a broad class of algorithms. However, a great deal remains to be explored, including an investigation of alternative discretizations. Further improvements on the optimal t value also provide an interesting avenue for future work, though typically the existing estimate already outperforms fast marching in terms of mean error (Table I). Another obvious question is whether a similar transformation can be applied to a larger class of Hamilton-Jacobi equations. Finally, *weighted* distance computation might be achieved by simply rescaling the source data.

REFERENCES

ALEXA, M. AND WARDETZKY, M. 2011. Discrete Laplacians on General Polygonal Meshes. *ACM Trans. Graph. 30,* 4, 102:1–102:10.

BELKIN, M., SUN, J., AND WANG, Y. 2009a. Constructing Laplace operator from point clouds in r^d. In *ACM-SIAM Symp. Disc. Alg.*

BELKIN, M., SUN, J., AND WANG, Y. 2009b. Discrete laplace operator for meshed surfaces. In *ACM-SIAM Symp. Disc. Alg.* 1031–1040.

BOMMES, D. AND KOBBELT, L. 2007. Accurate computation of geodesic distance fields for polygonal curves on triangle meshes. In *Proc. Workshop on Vision, Modeling, and Visualization (VMV)*. 151–160.

BOTSCH, M., BOMMES, D., AND KOBBELT, L. 2005. Efficient linear system solvers for mesh processing. In *IMA Conference on the Mathematics of Surfaces*. Springer, 62–83.

CAMPEN, M. AND KOBBELT, L. 2011. Walking on broken mesh: Defect-tolerant geodesic distances and parameterizations. *Computer Graphics Forum 30,* 2, 623–632.

CHEN, Y., DAVIS, T. A., HAGER, W. W., AND RAJAMANICKAM, S. 2008. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw. 35,* 22:1–22:14.

COIFMAN, R. R. AND LAFON, S. 2006. Diffusion maps. *Appl. Comput. Harmon. Anal. 21,* 5–30.

DESBRUN, M., KANSO, E., AND TONG, Y. 2008. Discrete Differential Forms for Computational Modeling. In *Discrete Differential Geometry*. Oberwolfach Seminars, vol. 38. Birkhäuser Verlag, 287–324.

FOUSS, F., PIROTTE, A., RENDERS, J.-M., AND SAERENS, M. 2007. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *Knowledge and Data Engineering, IEEE Transactions on 19,* 3 (march), 355–369.

HILDEBRANDT, K. AND POLTHIER, K. 2011. On approximation of the laplace-beltrami operator and the willmore energy of surfaces. *Comput. Graph. Forum 30,* 5, 1513–1520.

HYSING, S. AND TUREK, S. 2005. The eikonal equation: Numerical efficiency vs. algorithmic complexity on quadrilateral grids. In *Proc. Algoritmy.* 22–31.

JIANG, G. AND PENG, D. 1997. Weighted eno schemes for hamilton-jacobi equations. *SIAM J. Sci. Comput 21,* 2126–2143.

KIMMEL, R. AND SETHIAN, J. 1998. Fast Marching Methods on Triangulated Domains. *Proc. Nat. Acad. Sci. 95,* 8341–8435.

LIPMAN, Y., RUSTAMOV, R. M., AND FUNKHOUSER, T. A. 2010. Biharmonic distance. *ACM Trans. Graph. 29,* 27:1–27:11.

LIU, Y., PRABHAKARAN, B., AND GUO, X. 2012. Point-based manifold harmonics. *IEEE Trans. Vis. Comp. Graph. 18.*

LUO, C., SAFA, I., AND WANG, Y. 2009. Approximating gradients for meshes and point clouds via diffusion metric. *Comput. Graph. Forum 28,* 5, 1497–1508.

MACNEAL, R. 1949. The solution of partial differential equations by means of electrical networks. Ph.D. thesis, Caltech.

MEMOLI, F. AND SAPIRO, G. 2001. Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces. *Journal of Comp. Physics 173,* 730–764.

MEMOLI, F. AND SAPIRO, G. 2005. Distance functions and geodesics on submanifolds of $\mathbb{R}^d$ and point clouds. *SIAM J. Appl. Math. 65,* 4.

MITCHELL, J., MOUNT, D., AND PAPADIMITRIOU, C. 1987. The discrete geodesic problem. *SIAM J. of Computing 16,* 4, 647–668.

NEALEN, A. 2004. An as-short-as-possible intro. to the MLS method for scattered data approximation and interpolation. Tech. rep.

NEEL, R. AND STROOCK, D. 2004. Analysis of the cut locus via the heat kernel. *Surveys in Differential Geometry 9,* 337–349.

NORRIS, J. 1997. Heat Kernel Asymptotics and the Distance Function in Lipschitz Riemannian Manifolds. *Acta Math. 179,* 1, 79–103.

PEYRÉ, G. AND COHEN, L. D. 2005. *Prog. in Nonlin. Diff. Eq. and Their Applications.* Vol. 63. Springer, Chapter Geodesic Computations for Fast and Accurate Surface Remeshing and Parameterization, 157–171.

RANGARAJAN, A. AND GURUMOORTHY, K. 2011. A fast eikonal equation solver using the schrödinger wave equation. Tech. Rep. REP-2011-512, CISE, University of Florida. January.

SCHMITZ, P. G. AND YING, L. 2012. A fast direct solver for elliptic problems on general meshes in 2d. *Journal of Computational Physics 231,* 4.

SCHWARZ, G. 1995. *Hodge decomposition: a method for solving boundary value problems.* Lecture notes in mathematics. Springer.

SETHIAN, J. 1996. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision and Materials Science.* Cambridge University Press.

SPIELMAN, D. A. AND TENG, S.-H. 2004. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proc. ACM Symp. Theory Comp.* STOC '04. ACM, 81–90.

SURAZHSKY, V., SURAZHSKY, T., KIRSANOV, D., GORTLER, S. J., AND HOPPE, H. 2005. Fast exact and approximate geodesics on meshes. *ACM Trans. Graph. 24,* 553–560.

VARADHAN, S. R. S. 1967. On the behavior of the fundamental solution of the heat equation with variable coefficients. *Communications on Pure and Applied Mathematics 20,* 2, 431–455.

VILLA, T. 2006. Methods of geometric measure theory in stochastic geometry. Ph.D. thesis, Università degli Studi di Milano.

VON RENESSE, M.-K. 2004. Heat Kernel Comparison on Alexandrov Spaces with Curvature Bounded Below. *Potential Analysis 21,* 2.

WADE, B., KHALIQ, A., SIDDIQUE, M., AND YOUSUF, M. 2005. Smoothing with positivity-preserving Padé schemes for parabolic problems with nonsmooth data. *Numer. Meth. for Partial Diff. Eq. 21.*

WEBER, O., DEVIR, Y. S., BRONSTEIN, A. M., BRONSTEIN, M. M., AND KIMMEL, R. 2008. Parallel algorithms for approximation of distance maps on parametric surfaces. *ACM Trans. Graph. 27,* 4.

APPENDIX

A. A VARADHAN FORMULA FOR GRAPHS

LEMMA 1. *Let $G = (V, E)$ be the graph induced by the sparsity pattern of any real symmetric matrix A, and consider the linear system*

$$(I - tA)u_t = \delta$$

where I is the identity, δ is a Kronecker delta at a source vertex $u \in V$, and $t > 0$ is a real parameter. Then generically

$$\boxed{\phi = \lim_{t \to 0} \frac{\log u_t}{\log t}}$$

*where $\phi \in \mathbb{N}_0^{|V|}$ is the **graph distance** (i.e., number of edges) between each vertex $v \in V$ and the source vertex u.*

PROOF. Let σ be the operator norm of A. Then for $t < 1/\sigma$ the matrix $B := I - tA$ has an inverse and the solution u_t is given by the convergent Neumann series $\sum_{k=0}^{\infty} t^k A^k \delta$. Let $v \in V$ be a vertex n edges away from u, and consider the ratio $r_t := |s|/|s_0|$ where $s_0 := (t^n A^n \delta)_v$ is the first nonzero term in the sum and $s = (\sum_{k=n+1}^{\infty} t^k A^k \delta)_v$ is the sum of all remaining terms. Noting that $|s| \leq \sum_{k=n+1}^{\infty} t^k ||A^k \delta|| \leq \sum_{k=n+1}^{\infty} t^k \sigma^k$, we get

$$r_t \leq \frac{t^{n+1} \sigma^{n+1} \sum_{k=0}^{\infty} t^k \sigma^k}{t^n (A^n \delta)_v} = c \frac{t}{1 - t\sigma},$$

where the constant $c := \sigma^{n+1}/(A^n \delta)_v$ does not depend on t. We therefore have $\lim_{t \to 0} r_t = 0$, i.e., only the first term s_0 is significant as t goes to zero. But $\log s_0 = n \log t + \log(A^n \delta)_v$ is dominated by the first term as t goes to zero, hence $\log(u_t)_v / \log t$ approaches the number of edges n. $\square$

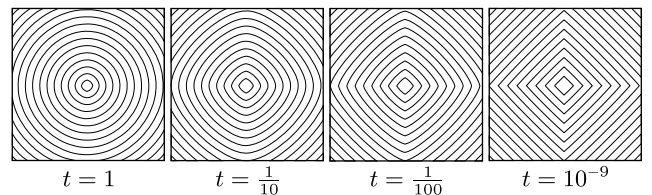Numerical experiments such as those depicted in Figure 18 agree with this analysis.

$t = 1$	$t = \frac{1}{10}$	$t = \frac{1}{100}$	$t = 10^{-9}$

Fig. 18. Isolines of $\log u_t / \log t$ computed in exact arithmetic on a regular grid with unit spacing ($h = 1$). As predicted by Lemma 1, the solution approaches the combinatorial distance as t goes to zero.
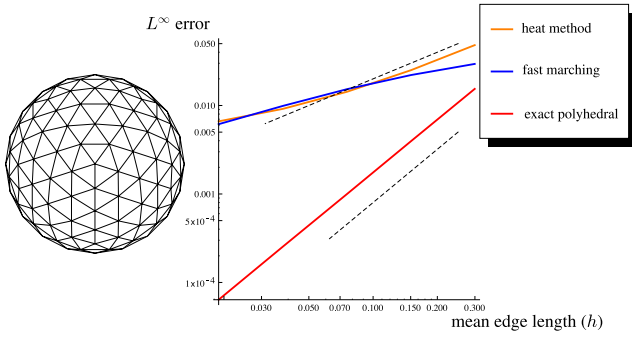
Fig. 19. L^∞ convergence of distance functions on the unit sphere with respect to mean edge length. As a baseline for comparison, we use the exact distance function $\phi(x, y) = \cos^{-1}(x \cdot y)$. Linear and quadratic convergence are plotted as dashed lines for reference; note that even the exact polyhedral distance converges only quadratically.
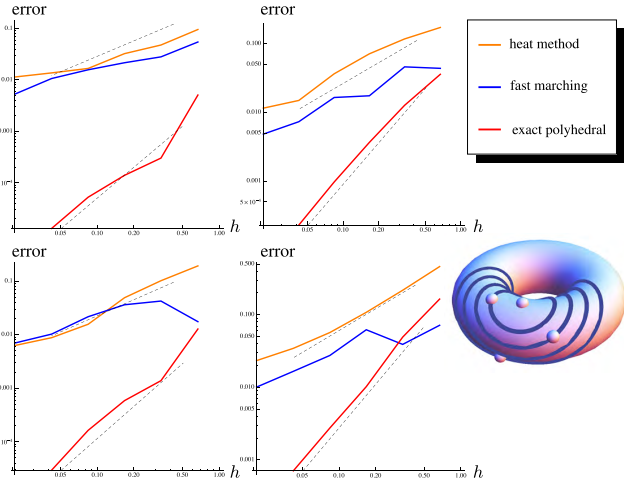

Fig. 20. Convergence of geodesic distance on the torus at four different test points. Error is the absolute value of the difference between the numerical value and the exact (smooth) distance; linear and quadratic convergence are plotted as dashed lines for reference. *Right:* test points visualized on the torus; dark blue lines are geodesic circles computed via Clairaut's relation.
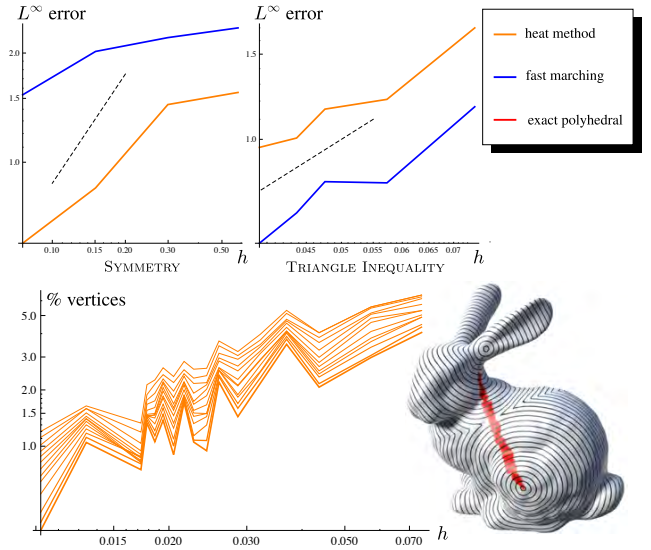
Fig. 21. Numerical approximations of geodesic distance exhibit small violations of metric properties that vanish under refinement. Here we examine errors in symmetry *(top left)* and the triangle inequality *(top right)* by checking all pairs or triples (respectively) of vertices on the Stanford bunny and plotting the worst violation as a percent of mesh diameter. Linear convergence is plotted as a dashed line for reference. *Bottom right:* violation of triangle inequality occurs only for degenerate geodesic triangles, *i.e.*, all three vertices along a common geodesic. Fixing the first two vertices, we plot those in violation in **red**. *Bottom left:* percent of vertices in violation; letting $t = mh^2$, each curve corresponds to a value of m sampled from the range $[1, 100]$.
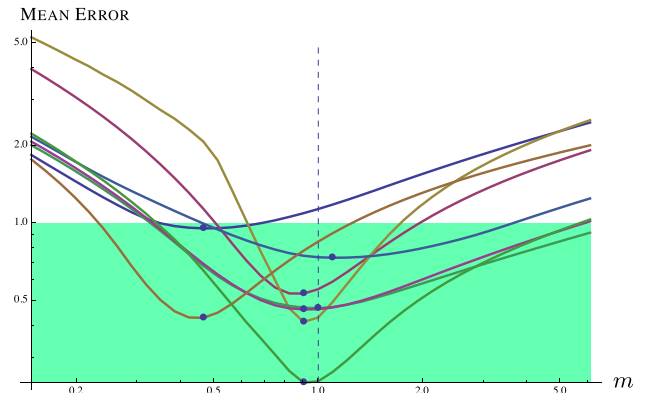

Fig. 22. Mean percent error as a function of m, where $t = mh^2$. Each curve corresponds to a data set from Table I. Notice that in most examples $m = 1$ (dashed line) is close to the optimal parameter value (blue dots) and yields mean error below 1%.