Geodesics in Heat

Keenan Crane
Caltech

Clarisse Weischedel
Universität Göttingen

Max Wardetzky
Universität Göttingen

Abstract

We introduce the *heat method* for computing the shortest geodesic distance to an arbitrary subset 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 resulting algorithm represents a significant breakthrough in the practical computation of distance on a wide variety of geometric domains, since these problems can be prefactored once and subsequently solved in 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 demonstrate 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 differentiability is required.

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

Links:

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 transfered from a source x to a destination y after time t. A well-known relationship between heat and distance is given by Varadhan's formula [1967]

$$d^2(x,y) = \lim_{t\to 0} -4t \log k_{t,x}(y),$$

which says that the geodesic distance d between any pair of points x, y on a Riemannian manifold can be recovered from the solution to a short-time heat flow. 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 manages to reach a distant point y in a small time t will have 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? One possibility is that for large values of t the function $-4t \log k_{t,x}$ is a poor approximation of geodesic distance, yet for small values of t the heat kernel is numerically inaccurate due to rapid exponential decay. A key insight of our method is that even for moderately large values of t, the *gradient* of the heat kernel continues to point very close to the correct direction, *i.e.*, along geodesic curves. We can therefore separate the computation of distance into two separate stages: first compute the gradient of the distance function via the heat kernel, then recover the distance function itself via Helmholtz-Hodge decomposition.

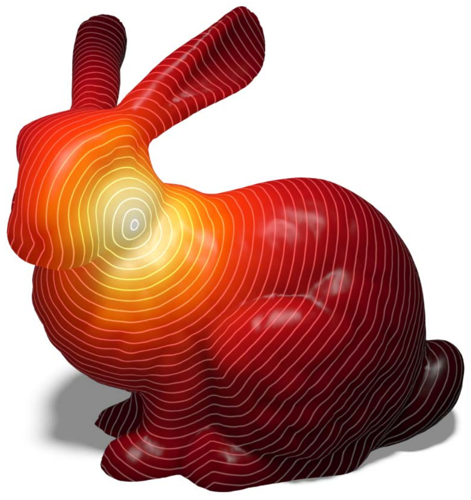


Figure 1: *Geodesic distance on the Stanford Bunny computed using the heat method.*

With respect to existing algorithms, the heat method offers two major advantages. First, it can easily be formulated for any space that admits a *Laplace–Beltrami operator*. As a consequence, our method can be applied to virtually any type of geometric discretization, including regular and irregular grids, polygonal meshes, and even unstructured point clouds. Second, our approach involves only the solution of *sparse linear systems*. As a result, problems where we want to compute the distance to a large number of different sets benefit enormously since linear systems can be prefactored once and subsequently inverted in linear time. This feature makes the heat method particularly promising for applications such as shape matching, path planning, and level set-based simulation (*e.g.*, free-surface fluid flows), all of which require repeated distance queries on a fixed geometric domain. Moreover, because Poisson-type equations are widespread in scientific computing, the heat method can immediately take advantage of new developments in numerical linear algebra and parallelization.

2 Related Work

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

$$|\nabla\phi| = 1 \tag{1}$$

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. In practice, Eq. (1) is often solved by applying an iterative relaxation scheme such as Gauss-Seidel – special update orders are known under the names *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]. Fast marching and fast sweeping have asymptotic complexity of $O(n \log n)$ and $O(n)$, respectively, but sweeping is often significantly slower due to the large

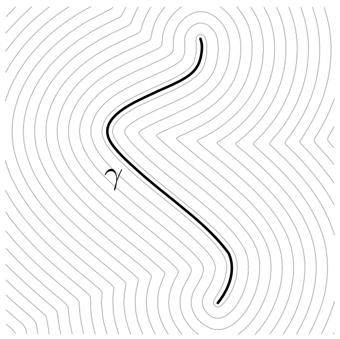


Figure 2: *The heat method computes the shortest distance to a subset γ of a given domain. (Gray curves indicate lines of equal distance.)*

number of sweeps required to obtain accurate results [Hysing and Turek 2005].

The main drawback of fast marching 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 ordering schemes 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; this same decomposition could be used to accelerate the heat method. Fast marching can also be used to approximate distance on implicit surfaces [Memoli and Sapiro 2001], point clouds [Memoli and Sapiro 2005], or polygon soup [Campen and Kobbelt 2011], but only indirectly: distance is computed on a simplicial mesh or regular grid that approximates the original domain.

In a different development, Mitchell *et al.* [1987] give an $O(n^2 \log n)$ algorithm for computing the exact piecewise linear geodesic 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 amortized cost of these algorithms (over many source vertices) will be substantially greater than for the heat method since there is no reuse of information. Finally, although the description given in [Surazhsky et al. 2005] is simpler than the original formulation, these algorithms are still 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 $d^2 = -\hbar \log \phi$ between the distance function and solutions ϕ to the time-independent Schrödinger equation. We emphasize, however, that this derivation applies only to $\mathbb{R}^n$ where ϕ takes a special form – in this case it may be just as easy to analytically invert the Euclidean heat kernel $k_{t,x} = (4\pi t)^{-n/2} e^{-d_E(x,y)^2/4t}$ to recover the Euclidean distance d_E. Moreover, they compute solutions using the fast Fourier transform, which limits computation to regular grids. For small values of $\hbar$ the issue of poor numerical accuracy is addressed by either combining multiple solutions from various values of $\hbar$, or by using arbitrary-precision arithmetic; no 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 metric of the original domain and can therefore substantially violate the unit-speed nature of geodesics (Figure 8). On spaces of constant curvature these distances can also be explained in terms of simple discretizations of heat flow – see Section 3.3 for further discussion.

Contributions We introduce the *heat method* for computing distance functions on a broad class of geometric domains and demonstrate convergence in the setting of finite elements (Appendix A). At a practical level, the heat method is ***general*** since it can be applied to any domain that admits a discrete Laplacian and gradient operator, ***simple*** to implement since it can be expressed in terms of well-established discrete differential operators and basic tools from numerical linear algebra, and highly ***efficient*** since it is based on solving sparse, positive-semidefinite linear systems (Section 3). In particular, these systems can be prefactored to yield excellent amortized performance for problems with many different sets of boundary conditions (Section 4.1). A single parameter t provides a flexible notion of distance: at one extreme ($t \to 0$), we recover exact geodesic distance; at the other extreme ($t \to \infty$) we get a smooth approximation which is invaluable in applications that require differentiability (Section 3.3). We also establish boundary conditions for our smooth approximation that successfully mimic the behavior of corresponding surfaces without boundary (Section 3.4). Together these features make the heat method a practical and powerful tool for digital geometry processing and numerical simulation.

3 The Heat Method

Our method can be described purely in terms of operations on smooth manifolds; we explore discretization in Sections 3.1 and 3.2. Let Δ be the negative-definite 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 time $t$.
 II. Evaluate the vector field $X = -\nabla u / |\nabla u|$.
III. Solve the Poisson equation $\Delta \phi = \nabla \cdot X$.
```

The final function ϕ approximates geodesic distance, approaching the exact distance as t goes to zero. Initial conditions $u_0 = \delta(x)$ for some point $x \in M$ recover the distance to a single source as in Figure 1, but in general we can compute the distance to any subset γ by setting u_0 to a distribution on γ (see Figures 2 and 14, right). Once the function ϕ is known, we can trace geodesic curves by simply following the gradient $\nabla \phi$ (Figure 14, left).

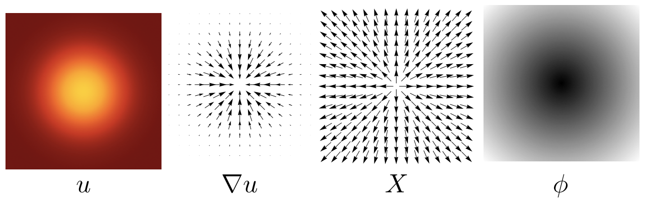


Figure 3: *Outline of the heat method.* Left to right: *(I) heat u is allowed to diffuse for a brief period of time; (II) the temperature gradient ∇u is normalized (and negated) to get a unit vector field X pointing along geodesics; (III) a function ϕ whose gradient follows X recovers the final distance.*

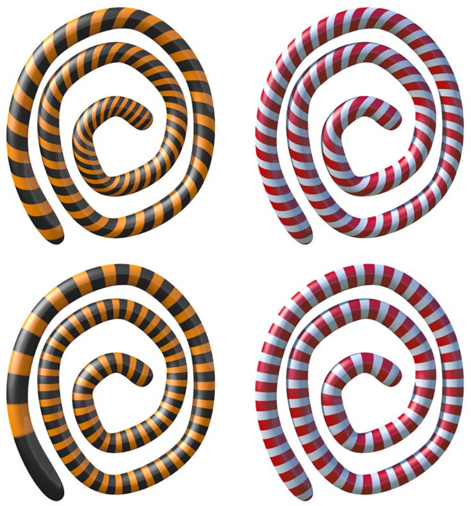


Figure 4: Top left: *even for very small values of t, simply taking the logarithm of the heat kernel may not provide an accurate approximation of geodesic distance.* Top right: *normalizing the heat kernel gradient results in a more accurate solution, as indicated by more evenly spaced isolines.* Bottom: *for large values of t Varadhan's formula produces uneven spacing (left); normalization becomes critical for producing a smoothed distance function (right).*

The heat method can be motivated as follows. Consider the solution u to the heat equation for a fixed (small) time t and with initial conditions $u_0 = \delta(x)$. Varadhan's formula asserts that if t is sufficiently small, then the quantity $\tilde{d} := \sqrt{-4t \log u}$ approximates the shortest geodesic distance to x. One numerical difficulty is that $\tilde{d}$ cannot be reliably evaluated for arbitrarily small values of t, since heat decays exponentially as we move away from the source x; for more moderate values of t, Varadhan's formula may not yield an accurate approximation of geodesic distance (see Figure 4). To remedy these difficulties, we observe that (i) the gradients ∇u and $\nabla \tilde{d}$ are parallel (despite having different magnitudes) and (ii) the gradient of the true distance function has unit length (as implied by the eikonal equation). We therefore normalize the gradient of u itself (yielding the vector field X), then compute the closest scalar potential ϕ by minimizing $\int_M |\nabla\phi - X|^2$. This final step, related to *Helmholtz-Hodge decomposition*, is equivalent to solving the Poisson equation $\Delta\phi = \nabla \cdot X$. (Note that ϕ is unique only up to an additive constant and should be shifted such that the smallest distance value is zero.) The overall procedure is depicted in Figure 3.

3.1 Time Discretization

We discretize the heat equation from step I of Algorithm 1 using a single backward Euler step. In practice, this means that we solve the linear (elliptic) equation

$$(\mathrm{id} - t\Delta)u = u_0. \tag{2}$$

Note that backward Euler maintains a *maximum principle*, preventing unphysical oscillations in the solution [Wade et al. 2005]. This scheme is not only simple to implement, but is also closely connected to Varadhan's original proof that heat flow recovers geodesic distance as $t \to 0$. Indeed, consider a region Ω on the smooth surface M, and let γ be the boundary of Ω. Fix t, and let v_t be the solution to the elliptic boundary value problem

$$\begin{aligned} (\mathrm{id} - t\Delta)v_t &= 0 \quad \text{on} \quad \Omega \\ v_t &= 1 \quad \text{on} \quad \gamma\,. \end{aligned} \tag{3}$$

An essential step of Varadhan's proof is to show that for any $x \in \Omega$,

$$\lim_{t\to 0} -\frac{\sqrt{t}}{2}\log v_t(x) = d(x,\gamma)\,, \tag{4}$$

where $d(x,\gamma)$ is the shortest geodesic distance from x to γ [Varadhan 1967]. If we use Eq. (3) as the starting point for our algorithm (rather than the better-known formula introduced in Section 1), then we do not even have to consider the question of time discretization. Eq. (4) also ensures the validity of steps II and III since the gradient of $\hat{d}_t := -(\sqrt{t}/2)\log v_t$ remains parallel to ∇v_t.

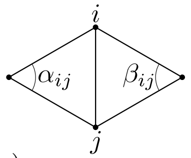
3.1.1 Time Step

The accuracy of the heat method relies in part on the choice of time step: for large t we get only a smooth approximation of geodesic distance; pick t too small and we may run into numerical issues. For triangulated surfaces, the time step $t^* := A_M/|F|$ captures the most important scaling laws, where A_M is the total surface area and $|F|$ is the number of faces. The numerator A_M ensures that t^* is *scale-independent*, i.e., resizing the mesh does not affect the appearance of the result. The denominator $|F|$ accounts for *resolution independence* – scaling the surface down by a factor s effectively increases the time step by a factor s^2 (consider the matrix A^{-1} from Section 3.2.1). Since the number of triangles also increases by a factor s^2, we divide by $|F|$. The only remaining degree of freedom is a constant factor c in front of t^*; simply using $c = 5$ works remarkably well in practice.

3.2 Spatial Discretization

In principle the heat method can be applied to any domain with a discrete Laplacian Δ and gradient operator ∇. Here we take a quick look at several possible discretizations on common domains.

3.2.1 Simplicial Meshes

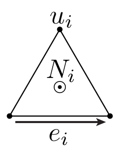


Let $u \in \mathbb{R}^{|V|}$ specify a piecewise linear function on a triangulated surface. A standard discretization of the Laplacian at a vertex i is given by

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

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 [Duffin 1959]. 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 $L_C \in \mathbb{R}^{|V|\times|V|}$ is the *cotan operator* representing the remaining sum. Heat flow can then be computed by solving the system $(I - tL)u = u_0$, (where I is the identity matrix) or equivalently

$$(A - tL_C)u = Au_0.$$



The advantage of the latter system is that it is symmetric and can be more efficiently solved or prefactored. (Note that a Dirac delta appears as a literal *one* in this system since we are effectively working with integrated quantities.) The gradient in a given triangle can be expressed succinctly as

$$\nabla u = \frac{1}{2A_f}\sum_i u_i(N \times e_i)$$

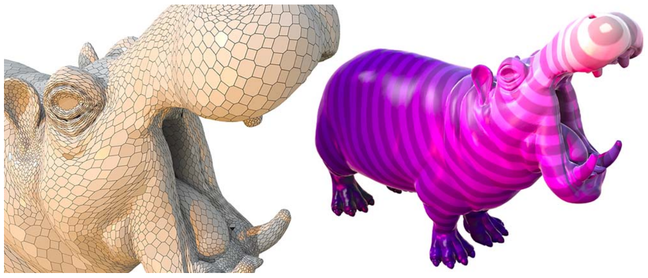
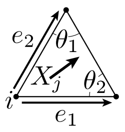


Figure 5: *Since the heat method is based on well-established discrete operators like the Laplacian, it is easy to adapt to a variety of geometric domains. Above: distance on a hippo composed of high-degree nonplanar (and sometimes nonconvex) polygonal faces.*



where A_f is the area of the face, N is its unit normal, e_i is the ith edge vector (oriented counter-clockwise), and u_i is the opposing value of u. 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 and θ_2 are the opposing angles. If we call the vector of (integrated) divergences d, then the final distance function is computed by solving the symmetric Poisson problem

$$L_C \phi = d.$$

Note that the heat method can be applied to a triangulated manifold of any dimension by using the standard Laplacian for piecewise linear functions; this matrix can easily be built using discrete exterior calculus via $L = \star_0^{-1} d_0^T \star_1 d_0$ [Desbrun et al. 2008].

3.2.2 Polygonal Surfaces

For a mesh with (not necessarily planar) polygonal faces, we use the polygonal Laplacian recently introduced 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). If we assume that ∇u is constant over a given face, then

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

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 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). Figure 5 demonstrates distance computed on an irregular polygonal mesh.

3.2.3 Point Clouds

For a discrete sample $P \subset \mathbb{R}^n$ 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. [2011], which extends previous work of Belkin et al. [2009a]. In this formulation,

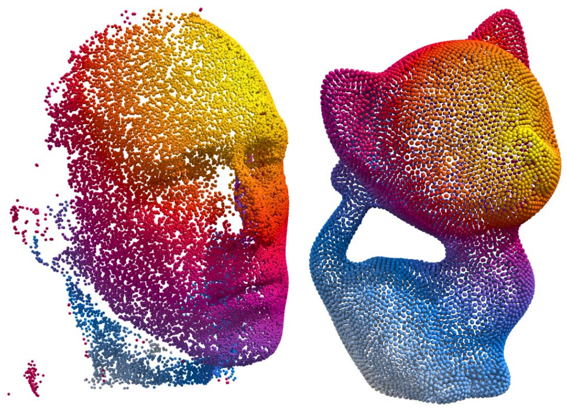


Figure 6: *The heat method can also be applied directly to scattered point clouds that lack connectivity information (yellow points are close to the source).* Left: *face scan with holes and noise; disconnected components receive a constant value (red)* Right: *kitten surface from Figure 10 with connectivity removed (for reference).*

the Laplacian is represented by $A^{-1}L_{PC}$, where A is a diagonal matrix of approximate Voronoi areas A_p associated with each point, and L_{PC} is a symmetric positive semidefinite matrix (see [Liu et al. 2011], 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 2003]. 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 $\phi \mapsto D\phi$, taking any scalar function to its gradient. To find the best-fit scalar potential ϕ (step III), we solve the quadratic minimization problem

$$\min_\phi \sum_{p \in P} A_p \|X(p) - D\phi(p)\|^2 ,$$

which again amounts to solving a linear, positive-semidefinite Poisson equation. The distance resulting from this approach is depicted in Figure 6. Other discretizations of ∇ are certainly possible; 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 such as machine learning where m may be significantly larger than n.

3.3 Smoothed Distance

The exact 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 "cusps" in the level lines of the distance function, giving the appearance of two wavefronts crashing together (see Figure 7). 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.

A number of 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 latter reference for a more detailed discussion).

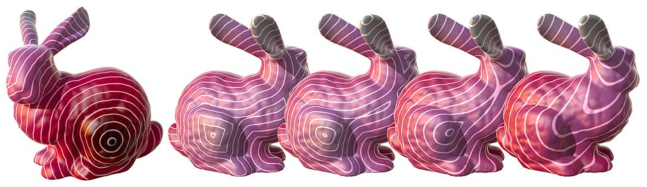


Figure 7: *A source on the front of the bunny results in nonsmooth "cusps" on the opposite side. By running our heat flow for progressively longer durations t, we obtain smooth approximations of the exact geodesic distance* (right).

These distances can be difficult to evaluate accurately, requiring one to compute a large number of Laplacian eigenvectors ($\sim 150 - 200$ in practice) or else solve a linear system for each vertex of the mesh. Moreover, although these distances satisfy a number of important properties (smoothness, isometry-invariance, *etc.*), they tend to poorly approximate the true geodesic distance, as indicated by uneven spacing of isolines (see Figure 8, middle).

In contrast, one can rapidly construct smoother versions of the exact distance by simply applying the heat method for large values of t (Figure 7). As before, the computational cost is a single linear solve; isolines stay evenly spaced for *any* value of t due to the normalization step. (Normalization also makes it challenging to prove that the resulting functions satisfy the properties of a metric, but numerical tests indicate that this is likely the case in practice.) Note that these smoothed functions are isometrically invariant – but not conformally invariant – since geometrically they depend only on the intrinsic Laplace–Beltrami operator.

Existing smooth distance functions can also be understood in terms of numerical approximations to 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 (respectively):

$$d_C(x,y)^2 = g_C(x,x) - 2g_C(x,y) + g_C(y,y),$$

$$d_B(x,y)^2 = g_B(x,x) - 2g_B(x,y) + g_B(y,y).$$

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 as one- and two-step approximations of heat flow via backward Euler as t goes to infinity:

$$g_C = \lim_{t\to\infty} (\mathrm{id} - t\Delta)^\dagger \delta,$$

$$g_B = \lim_{t\to\infty} (\mathrm{id} - 2t\Delta + t^2\Delta^2)^\dagger \delta.$$

(The symbol $\dagger$ 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

As t approaches zero, vanishing Neumann or Dirichlet boundary conditions both yield the exact geodesic distance (see [von Renesse 2004], Corollary 2 and [Norris 1997], Theorem 1.1, respectively). For larger values of t, however, the choice of boundary conditions matters and substantially alters the behavior of our smoothed geodesic distance – Figure 9 illustrates this behavior. We advocate the use of the *Robin boundary conditions* obtained by taking the mean of the Neumann solution u_N and the Dirichlet solution u_D,

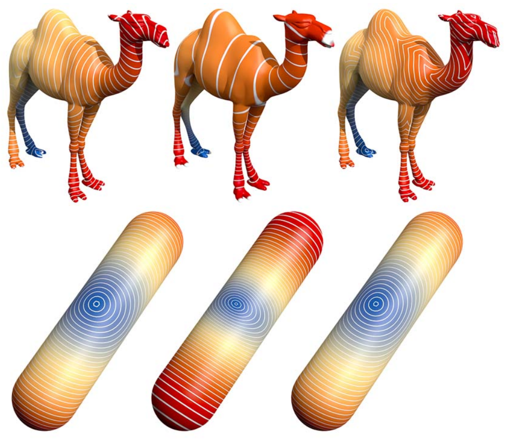


Figure 8: Top row: *our smooth approximation of geodesic distance (left) and biharmonic distance (middle) both mitigate sharp "cusps" found in the exact geodesic distance (right), but notice that isoline spacing of the biharmonic distance can vary dramatically. Bottom row: biharmonic distance (middle) 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).*

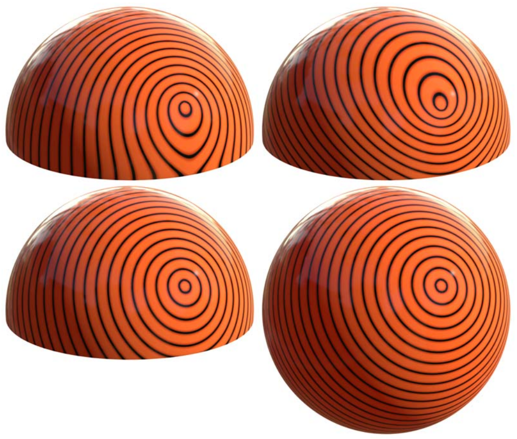


Figure 9: *For* smoothed *geodesic distance, boundary conditions substantially alter behavior. Pictured here are Neumann (top-left), Dirichlet (top-right) and Robin (bottom-left) boundary conditions. Note that Robin conditions qualitatively capture the behavior of the same surface without boundary.*

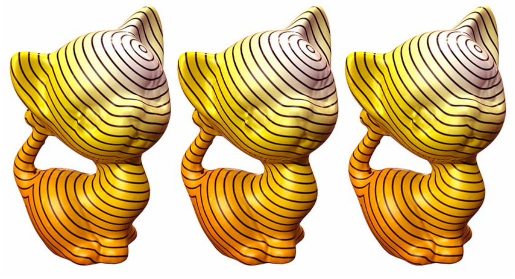
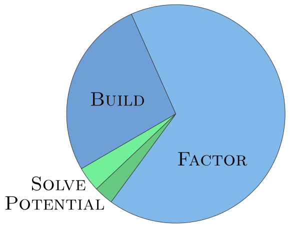


Figure 10: *Visual comparison of accuracy.* Left: *exact geodesic distance. The heat method* (middle) *and fast marching (right) both produce results of comparable accuracy, here within less than 1% of the exact distance – see Table 1 for a more detailed comparison.*

i.e., $u = \frac{1}{2}(u_N + u_D)$. The intuition behind these conditions 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. Robin conditions mimic the behavior of a domain without boundary: the number of walkers leaving equals the number of walkers returning. Figure 11 shows how different 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 describing heat flow and Helmholtz-Hodge decomposition can be prefactored. Our implementation uses sparse Cholesky factorization [Chen et al. 2008], which in theory has sub-quadratic complexity for Poisson-type problems 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 type of geometric domain, the choice of discrete Laplacian, particular geometric data structures, and so forth. As a typical example, we compared our simplicial implementation (Section 3.2.1) to the first-order $O(n \log n)$ fast marching method of Kimmel & Sethian [1998] and the $O(n^2 \log n)$ 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 [2005]; the heat method was implemented in C++ using a half edge data structure. All timings were taken on a 2.4 GHz Intel Core 2 Duo machine using a single core – Table 1 gives timing information. Note that for a single distance computation the heat method tends to outperform fast marching as mesh size increases; more importantly, updating distance via the heat method for new subsets is consistently at least an order of magnitude faster than both fast marching and the exact algorithm.

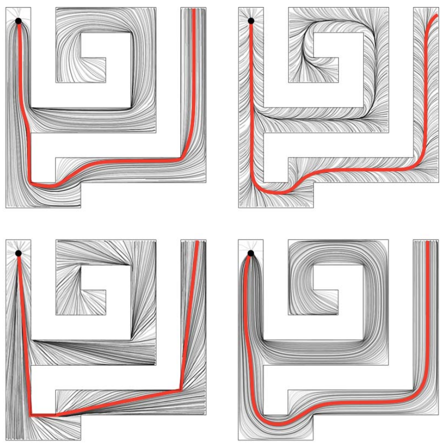


Figure 11: *For path planning, the behavior of geodesics can be controlled by altering boundary conditions and the integration time t.* Top-left: *Neumann conditions encourage boundary adhesion.* Top-right: *Dirichlet conditions encourage boundary avoidance.* Bottom-left: *small values of t yield standard straight-line geodesics.* Bottom-right: *large values of t yield more natural trajectories.*

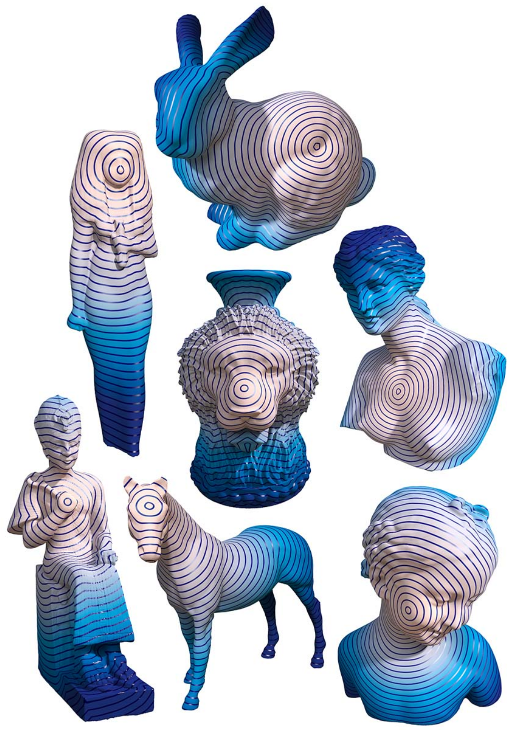


Figure 12: *Meshes used to test performance and accuracy (see Table 1). Top to bottom, left to right:* BUNNY, ISIS, LION, APHRODITE, RAMSES, HORSE, BIMBA.

4.2 Accuracy

The accuracy of distance computation relies primarily on the choice of spatial discretization. In the case of the heat method this means the choice of differential operators; although we explore only low-order operators in this paper, many interesting choices are available [Belkin et al. 2009b; Hildebrandt and Polthier 2011]. Accuracy is also affected by the choice of time step – see Section 3.1.1. 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]).

As a baseline for comparison in the simplicial case, we used the exact (piecewise linear) geodesic distance computed using the algorithm of Mitchell *et al.* [1987]. Table 1 gives accuracy information for a number of examples – MAX is the maximum absolute error as a percentage of mesh diameter and MEAN is the mean relative error at each vertex. Note that fast marching tends to achieve a smaller maximum error, whereas the heat method does better on average. Figure 10 gives a visual comparison of accuracy; the only notable discrepancy is that the heat method produces level sets that are slightly smoother at sharp cusps.

MODEL	FACES	HEAT METHOD				FAST MARCHING			EXACT
		FACTOR	SOLVE	MAX	MEAN	TIME	MAX	MEAN	TIME
BUNNY	28k	0.29s	0.02s	1.65%	**0.74%**	**0.28s**	**1.05%**	1.16%	0.99s
ISIS	93k	1.27s	0.11s	1.29%	**0.54%**	**1.08s**	**0.61%**	0.85%	5.61s
HORSE	96k	0.99s	0.07s	1.16%	**0.38%**	**1.01s**	**0.76%**	0.73%	6.28s
APHRODITE	106k	**1.13s**	**0.08s**	1.95%	**0.93%**	2.38s	**0.90%**	1.04%	4.68s
BIMBA	149k	**2.45s**	**0.15s**	1.35%	0.90%	2.78s	**0.61%**	**0.65%**	14.14s
LION	353k	**7.05s**	**0.37s**	**0.68%**	**0.44%**	10.93s	0.74%	0.68%	22.36s
RAMSES	1.6M	**26.47s**	**1.27s**	1.59%	**0.46%**	104.86s	**0.42%**	0.47%	110.17s

Table 1: *Comparison of the heat method with fast marching and exact geodesic distance on triangle meshes.* MAX *and* MEAN *are maximum absolute (w.r.t. mesh diameter) and mean relative error;* FACTOR, SOLVE *and* TIME *give execution time. Best overall time/accuracy for a single boundary set is indicated in bold. Notice that when updating the boundary set the heat method outperforms fast marching by an order of magnitude in all cases (compare* SOLVE *and* TIME *columns). Meshes pictured in Figure 12.*

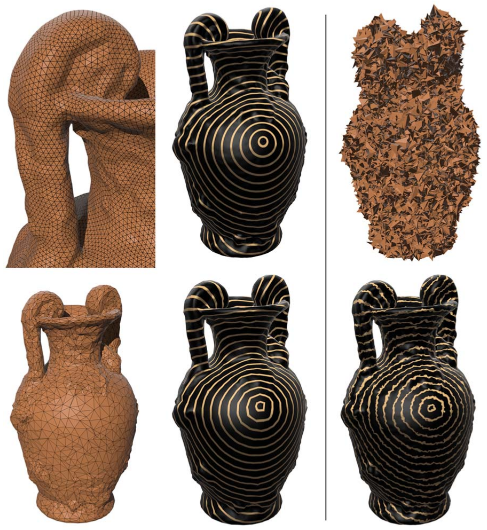


Figure 13: *Tests of robustness.* Left: *smoothed approximation of distance (Section 3.3) appears similar on meshes of different resolution.* Right: *even for meshes with severe noise* (top) *we recover a good approximation of the distance on the original surface* (bottom, visualized on noise-free mesh).

The approximate algorithm of Surazhsky *et al.* also provides an interesting comparison since it is on par with fast marching in terms of performance and produces results substantially closer to the exact piecewise linear distance (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.

4.3 Robustness

Two substantial factors contribute to the robustness of the heat method, namely (1) the use of an unconditionally stable implicit time-integration scheme that satisfies a maximum principle and (2) the fact that we are solving an elliptic PDE. Figure 13 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 approximation; the same behavior is observed for small t values.

5 Conclusion

The heat method is a simple, general method than can easily be incorporated into a broad class of algorithms. However, a great deal remains to be explored – in particular, a better quantitative understanding of numerical accuracy and a more thorough investigation of possible discretizations. Another obvious question is whether the same type of transformation can be applied to a more general class of Hamilton-Jacobi equations. We would also like to explore *weighted* distance computation, which appears to be a straightforward extension of the present algorithm.

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. 2009. Constructing Laplace operator from point clouds in R^d. In *ACM-SIAM Symp. Disc. Alg.*, SODA '09, 1031–1040.

BELKIN, M., SUN, J., AND WANG, Y. 2009. 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* (October), 22:1–22:14.

CIARLET, P. G. 1978. *The Finite Element Method for Elliptic Problems*. Studies in Mathematics and its Applications. North-Holland, Amsterdam.

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*, A. I. Bobenko, P. Schröder, J. M. Sullivan, and

G. M. Ziegler, Eds., Vol. 38 of *Oberwolfach Seminars*. Birkhäuser Verlag, 287–324.

DUFFIN, R. 1959. Distributed and Lumped Networks. *J. Math. Mech. 8*, 793–826.

DZIUK, G. 1988. Finite elements for the Beltrami operator on arbitrary surfaces. In *Partial Differential Equations and Calculus of Variations*, Vol. 1357 of *Lec. Notes Math.* Springer, 142–155.

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.

HILDEBRANDT, K., POLTHIER, K., AND WARDETZKY, M. 2006. On the convergence of metric and geometric properties of polyhedral surfaces. *Geometricae Dedicata 123*, 89–112.

HYSING, S., AND TUREK, S. 2005. The Eikonal Equation: Numerical Efficiency vs. Algorithmic Complexity on Quadrilateral Grids. In *Proc. Algoritmy*, nn, 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* (July), 27:1–27:11.

LIU, Y., PRABHAKARAN, B., AND GUO, X. 2011. Point-Based Manifold Harmonics. *IEEE Transactions on Visualization and Computer Graphics 99*, Preprint.

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 R^d and Point Clouds. *SIAM J. Appl. Math. 65*, 4, 1227–1260.

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

NEALEN, A. 2003. An As-Short-As-Possible Introduction to the Moving Least Squares Method for Scattered Data Approximation and Interpolation. *Comp. Meth. Appl. Mech. & Eng.*, 1, 0–2.

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. *Progress in Nonlinear Differential Equations and Their Applications*, Vol. 63. Springer, ch. 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, 1314–1338.

SCOTT, R. 1976. Optimal L^∞ Estimates for the Finite Element Method on Irregular Meshes. *Mathematics of Computation 30*, 136, 681–697.

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.*, ACM, STOC '04, 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.

VON RENESSE, M.-K. 2004. Heat Kernel Comparison on Alexandrov Spaces with Curvature Bounded Below. *Potential Analysis 21*, 2, 151–176.

WADE, B., KHALIQ, A., SIDDIQUE, M., AND YOUSUF, M. 2005. Smoothing with positivity-preserving Padé schemes for parabolic problems with nonsmooth data. *Numerical Methods for Partial Differential Equations 21*, 553–573.

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, 104:1–104:16.

A Convergence

We sketch a proof that the heat method recovers geodesic distance on a smooth surface in the limit of spatial refinement and as $t \to 0$. Consider a triangulated surface M_h that is *closely inscribed* into a smooth surface $M \subset \mathbb{R}^3$, *i.e.*, the vertices of M_h reside on M, and M_h lies within the so-called reach of M. As usual, h denotes the longest edge length of M_h. Let $\Delta_h = A^{-1}L_C$ denote the cotangent Laplacian of M_h (see Sec. 3.2.1). For fixed t, let $v_{h,t}$ be the solution of (3) with Δ_h in place of Δ. Here we assume that γ is sampled with straight edges of M_h and that the triangulation M_h is uniformly shape regular, independent of h. Let v_t be the solution of (3). Then due to the ellipticity of the operator $L_t = (id - t\Delta)$, the convergence analysis in [Dziuk 1988; Hildebrandt et al. 2006] seamlessly carries over to our setting, yielding

$$\|v_t - v_{h,t}\|_0 + h\|\nabla v_t - \nabla v_{h,t}\|_0 \leq \frac{C}{t}h^2 \ ,$$

where the constant $C = C(M, \gamma)$ is independent of t and h, and where $\|\cdot\|_0$ denotes the L^2-norm. Notice that due to the normalization step (II) of the heat method, these estimates do not yet suffice for our purpose. However, well-known estimates for the maximum-norm error for linear elements (see, *e.g.*, [Ciarlet 1978; Scott 1976]) yield that $\|\nabla v_{h,t} - \nabla v_t\|_\infty \to 0$ as $h \to 0$, implying that $X_{h.t} := \nabla v_{h.t}/\|\nabla v_{h,t}\|$ converges to $X_t := \nabla v_t/\|\nabla v_t\|$ in the maximum norm as well. This implies that the L^2-best-fit scalar potential $\phi_{h,t}$ of $X_{h,t}$ converges to the L^2-best-fit scalar potential ϕ_t of X_t, which together with (4) implies convergence of $\phi_{h,t}$ to the geodesic distance on the limit surface M, *provided that* $h \to 0$ at a considerably faster rate than $t \to 0$. We leave further analysis for future investigation.

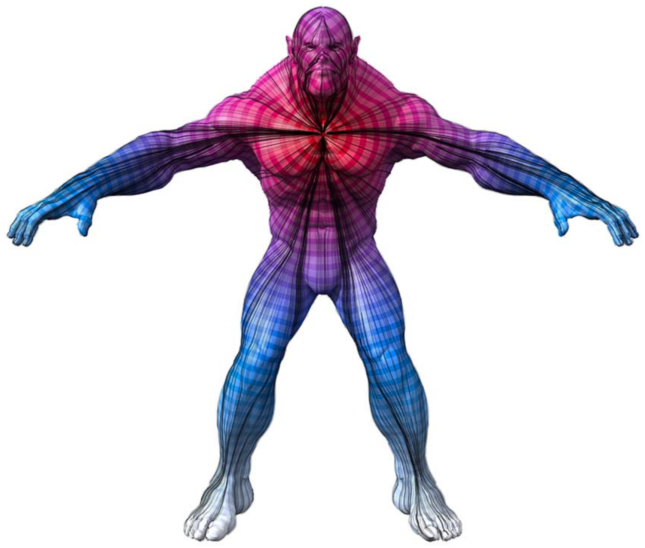


Figure 14: *Tracing geodesics is simply a matter of following the gradient of the distance function ϕ.*

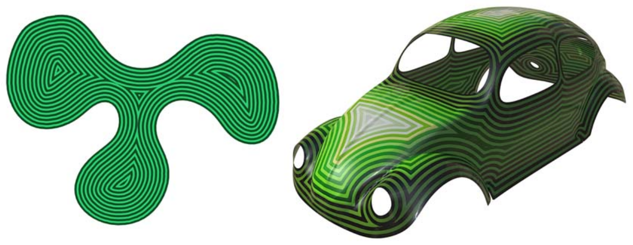


Figure 15: *Distance to the boundary on a region in the plane (left) or a surface in $\mathbb{R}^3$ is achieved by simply placing heat along the boundary curve. Note good recovery of the* cut locus*, i.e., points with more than one closest point on the boundary.*

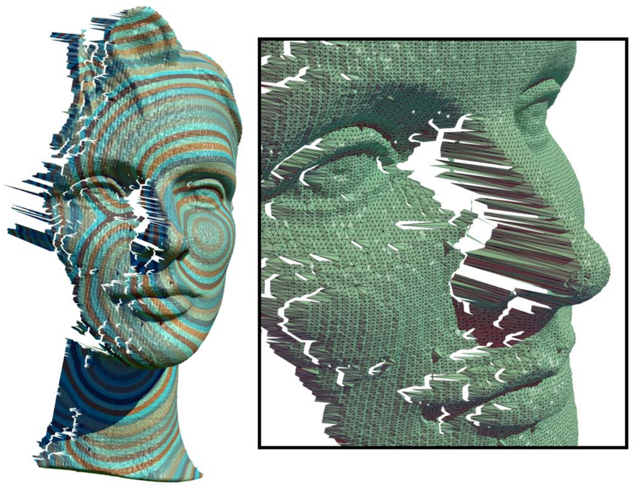


Figure 16: *Smoothed geodesic distance on an extremely poor triangulation with significant noise – note that small holes are essentially ignored. Also note good approximation of distance even along thin slivers in the nose.*