A Simple Heat Method for Computing Geodesic Paths on General Manifold Representations

Nathan King
University of Waterloo
Waterloo, ON, Canada

Steven Ruuth
Simon Fraser University
Burnaby, BC, Canada

Christopher Batty
University of Waterloo
Waterloo, ON, Canada

1 Geodesics, Harmonic Maps, and Heat Flow

We propose a novel algorithm for computing geodesic paths on general manifolds given only the ability to perform closest point queries. Many computer graphics applications require the computation of geodesic paths (see e.g., [Sharp and Crane 2020]), which involves minimizing length (e.g., [Yuan et al. 2021]) or geodesic curvature (e.g., [Martínez et al. 2005]). However, as Yuan et al. [2021] point out, existing methods have mainly been designed specifically for meshes. Instead, we view geodesics in the setting of harmonic maps, which leads to an algorithm that can be applied to meshes, parametric surfaces, point clouds, level sets, exact closest point functions, and more. Figure 1 shows some geodesics (red) computed on different manifold representations with our algorithm.

A harmonic map $\mathbf{u}(\mathbf{x}) : \mathcal{M} \to \mathcal{N}$ is a mapping from a *source* manifold $\mathcal{M} \subseteq \mathbb{R}^m$ to a *target* manifold $\mathcal{N} \subseteq \mathbb{R}^n$ that minimizes

$$E_H(\mathbf{u}) = \frac{1}{2}\int_{\mathcal{M}} \|\mathbf{J}_{\mathbf{u}}^{\mathcal{M}}\|_{\mathcal{F}}^2 \, d\mathcal{M}, \tag{1}$$

i.e., the Dirichlet energy, where $\mathbf{J}_{\mathbf{u}}^{\mathcal{M}}$ is the intrinsic Jacobian of the map on $\mathcal{M}$ and $\|\cdot\|_{\mathcal{F}}$ is the Frobenius norm. Due to the constraint that $\mathbf{u} \in \mathcal{N}$, the gradient descent flow for (1) is [Mémoli et al. 2004]

$$\frac{\partial \mathbf{u}}{\partial t} = \Pi_{T_{\mathbf{u}}\mathcal{N}}(\Delta_{\mathcal{M}}\mathbf{u}), \tag{2}$$

where $\Pi_{T_{\mathbf{u}}\mathcal{N}}$ is the projection onto the tangent space of $\mathcal{N}$ at $\mathbf{u}$ and $\Delta_{\mathcal{M}}$ is the Laplace-Beltrami operator (applied componentwise).

We use a key fact stated by Eells and Lemaire [1978]: if $\dim \mathcal{M} = 1$, then harmonic maps are geodesics of $\mathcal{N}$. Such a map will yield a closed or open geodesic on $\mathcal{N}$ depending on if $\mathcal{M}$ is closed (e.g., a circle) or open (e.g., a line segment), respectively. Our approach is therefore to compute a geodesic on the manifold $\mathcal{N}$ by computing a harmonic map from the 1D line segment $\mathcal{M} = [0, 1]$ to $\mathcal{N}$.

We adopt the harmonic mapping approach of King and Ruuth [2017] to minimize (1) by evolving (2) via a splitting method. Our resulting algorithm requires only heat flow on the 1D line segment and the closest point projection $\mathrm{cp}_{\mathcal{N}}$ onto $\mathcal{N}$ — it inherits these attractive features from the method of King and Ruuth. Since they did not consider the geodesic problem, we discuss in Section 2 some intricacies involved in computing initial paths and stopping criteria.

For the line segment $x \in \mathcal{M} = [0, 1]$, the Laplace-Beltrami operator is simply $\Delta_{\mathcal{M}} = \frac{\partial^2}{\partial x^2}$. The initial condition for the gradient descent flow is some path $\mathbf{u}^0(x) \in \mathcal{N}$. Starting from $k = 0$, the geodesic path is computed by iterating the following two steps:

(I) Solve $\dfrac{\partial \mathbf{v}}{\partial t} = \dfrac{\partial^2 \mathbf{v}}{\partial x^2}$, $\mathbf{v}(x, 0) = \mathbf{u}^k(x)$, for one time step of size Δt using explicit Euler.

(II) Project $\mathbf{v}(x, \Delta t)$ onto $\mathcal{N}$ via $\mathbf{u}^{k+1}(x) = \mathrm{cp}_{\mathcal{N}}(\mathbf{v}(x, \Delta t))$.

The notation in step (I) means to solve the heat equation independently for each component of $\mathbf{v}$. For open geodesics with endpoints $\mathbf{p}, \mathbf{q} \in \mathcal{N}$, Dirichlet boundary conditions $\mathbf{u}(0) = \mathbf{p}$ and $\mathbf{u}(1) = \mathbf{q}$ are imposed. For closed geodesics, we use periodic boundary conditions $\mathbf{u}(0) = \mathbf{u}(1)$ on the line segment to avoid complicating step (I) when $\mathcal{M}$ is curved (e.g., a circle).

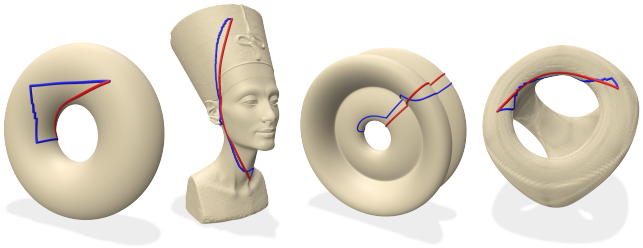


Figure 1: Our algorithm to compute geodesic paths is applicable to any manifold representation that supports closest point queries. The initial path (blue) is iteratively shortened to a geodesic (red) using our heat-based method. Manifold representation from left to right: exact closest points, mesh, parameterization, level set.

2 Discretization

The dimension n of the embedding space $\mathbb{R}^n$ for $\mathcal{N}$ is taken to be as small as possible, i.e., $n = \dim \mathcal{N} + 1$, for efficiency. For example, $n = 3$ if $\mathcal{N}$ is a surface (2D manifold). The line segment $\mathcal{M} = [0, 1]$ is discretized using equally spaced grid points $x_i = i\Delta x$ for $i = 0, 1, \ldots, N+1$ and $\Delta x = \frac{1}{N+1}$. The geodesic path $\mathbf{u} \in \mathcal{N} \subseteq \mathbb{R}^n$ is represented discretely as a polyline with vertices $\mathbf{u}_i = \mathbf{u}(x_i)$.

Step (I) of the algorithm is applied independently for each of the n dimensions of $\mathbf{u}$. Let u and v denote one of the particular n components of $\mathbf{u}$ and $\mathbf{v}$, respectively. Second-order centred differences are used to discretize $\partial^2/\partial x^2$. On iteration k, we set $\mathbf{v}_i = \mathbf{u}_i^k$, then apply one step of explicit Euler to the heat equation, with $\Delta t = 0.5\Delta x^2$:

$$v_i^{\Delta t} = v_i + \frac{\Delta t}{\Delta x^2}(v_{i-1} - 2v_i + v_{i+1}). \tag{3}$$

Then step (II) couples the n dimensions again for each vertex i of the path via $\mathbf{u}_i^{k+1} = \mathrm{cp}_{\mathcal{N}}(\mathbf{v}_i^{\Delta t})$. The closest point computation is different for each type of manifold representation, but is viewed as a black box. King et al. [2024, Appendix A] provide closest point computation details for many common representations.

Initial Path Construction. Since our method works for general manifold representations via closest point queries, our path initialization should also. We therefore construct a uniform grid of points $\mathbf{y}_i$ in the embedding space of $\mathcal{N}$ with $\|\mathbf{y}_i - \mathrm{cp}_{\mathcal{N}}(\mathbf{y}_i)\| \leq \frac{3h}{2}$, where h is the grid spacing. King et al. [2024] provide a memory

and runtime efficient algorithm to construct the grid, denoted Ω. We use Dijkstra's algorithm to compute a path between the nearest grid points in Ω to $\mathbf{p}$ and $\mathbf{q}$. We then replace the grid points $\mathbf{y}_i$ in Dijkstra's path with their previously computed $\mathrm{cp}_{\mathcal{N}}(\mathbf{y}_i)$. Finally, we spatially adapt the initial path, splitting and collapsing edges until all edge midpoints lie no further from $\mathcal{N}$ than a tolerance.

Stopping Criteria. Different stopping criteria have been used previously for iterative geodesic algorithms. Perhaps the most obvious stopping criterion would monitor the change in length of the curve. However, as Martínez et al. [2005, Section 3.3.1] discuss, the difference in lengths between consecutive iterations can be small even when the iteration has not converged. We observe the same behaviour in practice for our method. Instead, we measure the average change of the vertices in $\mathbf{u}$ between consecutive iterations and stop if a is less than a specified tolerance. We have observed that the individual vertices still move when not converged, even when the length of $\mathbf{u}$ only changes slightly.

3 Comparison to Yuan et al. [2021]

One variant of the method of Yuan et al. has some similarities to ours. However, their perspective is based on the length minimization property of geodesics. For a continuously differentiable curve $\gamma : [0,1] \to \mathcal{N}$ the functional

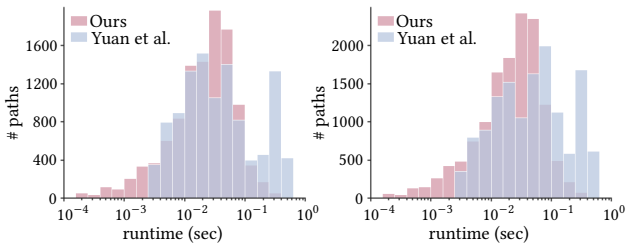
$$E_L(\gamma) = \int_0^1 H(\|\gamma'(t)\|)\, dt,$$

has the same critical points as the length functional, i.e., $E_L(\gamma)$ with $H(s) = s$. The function $H(s)$ must be convex and satisfy $H'(s) > 0$ and $H''(s) \geq 0$. Yuan et al. minimize $E_L(\gamma)$ using LBFGS and show (albeit with a simple example of shortening a "S" shaped curve in the plane) that $H(s) = s^2$ and $H(s) = e^{s^2} - 1$ require far fewer iterations than $H(s) = s$. They chose $H(s) = e^{s^2} - 1$ since slightly fewer iterations were needed compared to $H(s) = s^2$.

In the discrete setting, there are subtle differences between our method and theirs with $H(s) = s^2$. For this specific case, the ∇E_L used in LBFGS is a scalar multiple of the finite difference scheme we use for $\partial^2/\partial x^2$ in (3). The $\partial^2/\partial x^2$ term is related to the gradient of E_H, but only the specific combination in steps (I)-(II) give a first-order consistent discretization of the gradient descent flow (see King and Ruuth [2017, Section 3.2] for the proof).

Importantly, our approach does not require computing and applying the projection operator $\Pi_{T_{\mathbf{u}}\mathcal{N}}$. Yuan et al. use $\Pi_{T_{\mathbf{u}}\mathcal{N}}$ to project ∇E_L onto the tangent space of $\mathcal{N}$; otherwise their minimization does not converge. Even with these projected gradients their method can produce geodesics that are not strictly in $\mathcal{N}$. Therefore, after each iteration, they put the vertices of their path on $\mathcal{N}$ by computing closest points. They give no theoretical justification for using $\Pi_{T_{\mathbf{u}}\mathcal{N}}(\nabla E_L)$ or $\mathrm{cp}_{\mathcal{N}}$, whereas, our approach has firm theoretical justification for not needing $\Pi_{T_{\mathbf{u}}\mathcal{N}}$ and computing $\mathrm{cp}_{\mathcal{N}}$ in step (II).

Avoiding computation of $\Pi_{T_{\mathbf{u}}\mathcal{N}}$ is the principal reason we enjoy the faster runtimes shown in Figure 2. This result is significant since their method was faster than 9 others they compared against (only Dijkstra's algorithm was faster, but it does not provide a smooth path due to its restriction to edges). Yuan et al. did not, however, compare to [Sharp and Crane 2020], which only works on triangle meshes, but requires only ~10% of the runtime for Dijkstra's algorithm. We hope our approach can achieve similar runtimes in the future since both steps (I) and (II) are easily parallelizable.



(a) Fish **(b) Möbius Strip**

Figure 2: Computing geodesics from one vertex to every other vertex in the mesh of (a) a fish and (b) a Möbius strip. Our approach can achieve runtimes nearly two orders of magnitude faster than Yuan et al. [2021] starting from the same initial path. On average our algorithm is slightly faster than theirs.

Acknowledgments

Nathan King was supported in part by the QEII Graduate Scholarship in Science & Technology. Steven Ruuth was supported in part by a NSERC Discovery grant (RGPIN-2022-03302). Christopher Batty was supported in part by a NSERC Discovery grant (RGPIN-2021-02524) and the CFI-JELF program (Grant 40132).

References

James Eells and Luc Lemaire. 1978. A report on harmonic maps. *Bulletin of the London mathematical society* 10, 1 (1978), 1–68.

Nathan King, Haozhe Su, Mridul Aanjaneya, Steven Ruuth, and Christopher Batty. 2024. A Closest Point Method for PDEs on Manifolds with Interior Boundary Conditions for Geometry Processing. *ACM Transactions on Graphics* (2024).

Nathan D. King and Steven J. Ruuth. 2017. Solving variational problems and partial differential equations that map between manifolds via the closest point method. *J. Comput. Phys.* 336 (2017), 330–346.

Dimas Martínez, Luiz Velho, and Paulo C. Carvalho. 2005. Computing geodesics on triangular meshes. *Computers & Graphics* 29, 5 (2005), 667–675.

Facundo Mémoli, Guillermo Sapiro, and Stanley Osher. 2004. Solving variational problems and partial differential equations mapping into general target manifolds. *J. Comput. Phys.* 195, 1 (2004), 263–292.

Nicholas Sharp and Keenan Crane. 2020. You Can Find Geodesic Paths in Triangle Meshes by Just Flipping Edges. *ACM Trans. Graph.* 39, 6 (2020).

Na Yuan, Peihui Wang, Wenlong Meng, Shuangmin Chen, Jian Xu, Shiqing Xin, Ying He, and Wenping Wang. 2021. A variational framework for curve shortening in various geometric domains. *IEEE Transactions on Visualization and Computer Graphics* 29, 4 (2021), 1951–1963.