GEODESICS IN HEAT IMPLEMENTATION

Joshua Waha

Contents

1 Paper

1.1 Summary

Geodesic distance is described as the shortest distance between two points along a surface. The paper 'Geodesics in Heat' [1], looks at a method of approximating geodesic distance on surface manifolds by setting up 2 linear problems and solving them. The inspiration for this paper came from how heat diffuses across surfaces over the time and shows a way of examining this diffusion to recover the geodesic distance between points.

The method has two major benefits, with one being its broad applicability to various geometric discretizations. This is because for this method to be possible you only need to be able to calculate the discrete version of the Laplace-Beltrami operator. The other benefit is its computational efficiency, as the matrices used in the linear systems can be pre-factored and reused for solving for different source points.

The heat method has a dependence on the type of spatial discretization used, even suggesting that better results could be achieved with alternate formulations of the discrete Laplace-Beltrami. I would also critique the fact that the papers parameter selection for the time step t is based on a heuristic estimate rather than a proven optimal choice.

1.2 Method

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 $\Delta \phi = \nabla \cdot X$.

Figure 1: Algorithm 1: The Heat Method [1]

The method can be broken down into 3 parts as shown above 1. This method was then discretized in the paper so that it could be applied to popularly used geometric discretizations such as triangular meshes and point clouds.

The first step is to solve the heat equation over a set time (t) which gives us the heat flow over the surface manifold (u).

Then the gradient of u is taken which represents the direction of fastest temperature increase. However, it's highly unlikely the magnitude of the gradient vectors will correctly represent the rate of decay needed to predict the true distance. Fortunately we can ignore the magnitude of these vectors by normalising as the Eikonal equation ($|\nabla \phi| = 1$) states that the magnitude of the gradients of the true distance function has unit norm. Note that we take the negative of this normalised vector field as the gradient will point to the steepest 'uphill path' towards the heat source.

For the final step, the paper proposes solving the minimization $\int_M |\nabla \phi - X|^2$ to solve

for the scalar potential (ϕ). ϕ assigns a real scalar number for every point on the surface manifold, which in this case will be the approximated geodesic distance. By minimising the gradient of ϕ against the normalised vector field (X) it aligns ϕ such that if you were to stand on the surface and walk in the direction of the steepest slope you would be walking along the geodesic paths indicated by X. To solve this discretely we can solve the Poisson equation stated as step 3 in figure 1. Finally, the values of ϕ are shifted so that the smallest distance is equal to 0.

2 My Implementation

My implementation of the heat method was for triangular simplicial meshes. I used `libigl` [2] cotan discretization of the Laplace-Beltrami so that I could test different versions of it during the evaluation step.

2.1 Step 1 - Discrete Heat Flow

To compute the discrete heat flow over the mesh, I solve the linear equation:

$$(A - tL_C)u = \delta_\gamma$$

where A is the mask matrix, t is the stated time step, L_c is the cotangent Laplacian matrix, u is is the vector of unknown temperatures at time t and δ_γ is the Dirac delta which represents the initial heat distribution. Note that δ_γ will be a vector of 0s apart from at the index of the vertex which is the designated source point where its value will be 1. The value of t is recommended by the paper to be set to h^2 where h is the average distance between adjacent vertices. I used the iterative conjugate gradients method from `scipy` [3] to solve the linear system.

We solve $A - tL_C$ instead of $I - t\Delta$ as the former is symmetric positive-definite, by construction, so is much easier to solve. The reason solving this equation give This matrix forms the basis of the backward Euler method for solving the heat equation. The term $I - t\Delta$ term effectively computes how much heat remains at the original source point and how much diffuses to neighboring points based on the constructed Laplace-Beltrami of the mesh.

2.2 Step 2 - Normalized vector field

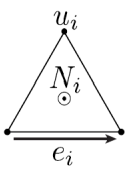


Figure 2: Reference triangle to calculate ∇u [1]

Next to calculate the vector field of the heat flow I implemented the equation from the

paper:

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

using figure 2 as a reference. It had to be noted that the vertices around the triangle needed to be referenced with anticlockwise ordering. Using the `libigl` library to load the mesh ensured the retention of this ordering. Whilst calculating the gradient I negated and normalized it resulting in the cell returning the array X.

2.3 Step 3 - Divergence of the vector field

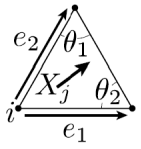


Figure 3: Reference triangle to calculate $\nabla \cdot X$ [1]

We now calculate the right hand side of the Poisson equation $\nabla \cdot X$, i.e the divergence of the vector field at each vertex. In terms of the heat method the divergence at each vertex represents the cost to get to the one-ring neighbours from that vertex. High positive divergence at a vertex suggests that the geodesic paths locally diverge from that point, while negative divergence would indicate converging paths. To calculate this we solve the equation:

$$\nabla \cdot X = \frac{1}{2} \sum_j \left(\cot \theta_1 (e_1 \cdot X_j) + \cot \theta_2 (e_2 \cdot X_j) \right)$$

using the reference figure 3. This weights the contributions of the adjacent faces to each vertex.

2.4 Step 4 - Poisson Equation

We now have all the components to solve the equation:

$$L_C \phi = \nabla \cdot X$$

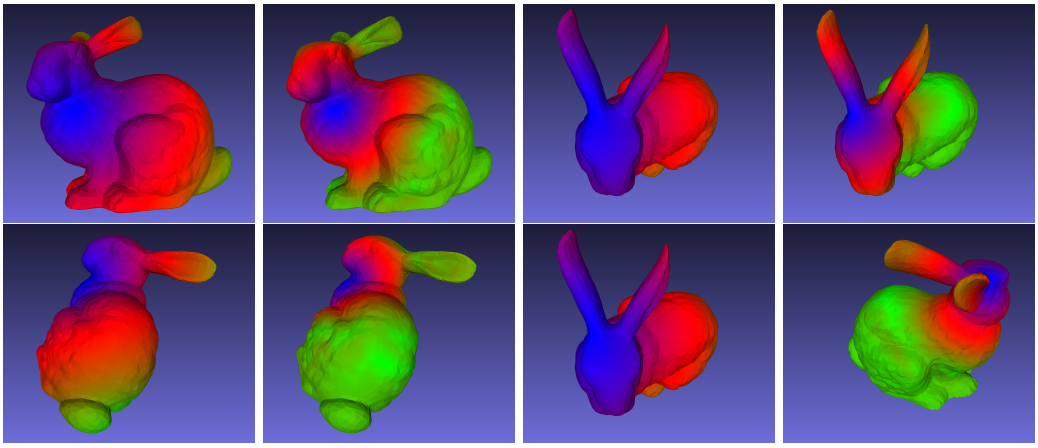
which we do using the iterative conjugate gradients method again. The values of ϕ are then shifted so that the smallest value equals 0.

To present the geodesic distance values on the mesh I normalized the values between 0 and 1, projected them into `matplotlib`'s [4] `brg` color map values and set the vertex colour's of the mesh to these values.

3 Evaluation

The evaluations were carried out on the bunny mesh from the `Stanford Computer Graphics Laboratory`.

3.1 Varying timestep

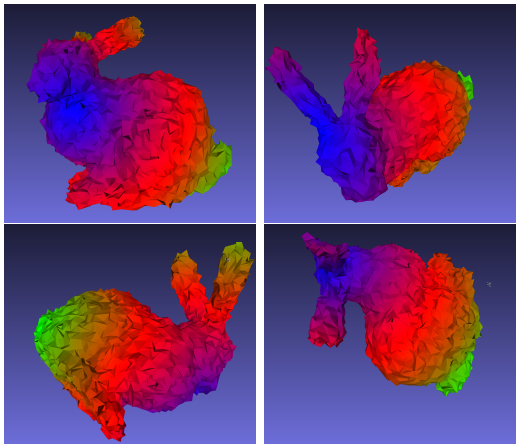


$t = h$, vertex 500 $t = h^2$, vertex 500 $t = h$, vertex 2000 $t = h$, vertex 2000

Figure 4: Varying timestep (t) in Geodesic Heat Method

In figure 4, I observe how altering the timestep used to calculate the heat flow over the surface manifold affects the geodesic distance approximation for 2 different source points (the vertex with index 500 and 2000). We observe that it is a more smoothed approximation of the distance when $t = h$, whereas when $t = h^2$ we can observe the difference between the closer points to the source with more clarity. I believe that the result is further smoothed when $t = h$ as the timestep in this case is larger as $h < 1$. This larger timestep means that the calculated heat flow (gradient field) across the manifold will have spread further and their will be less 0 values in u. For example when calculating the heat flow for $t = h^2$ on vertex 2000, the amount of zero values in u is 1714 in 3485 vertices, meaning that when the vector field is calculated there will be multiple areas with zero gradient. This will influence the final geodesic distance minimisation with the value not increasing after a certain distance from the source point due to it all being seen as an area of 0 gradient in the vector field. This result can be empirically seen by a lot of the bunny being green when $t = h^2$.

3.2 Effect of noise



$\sigma = 0.01$, vertex 500 $\sigma = 0.01$, vertex 2000

Figure 5: Adding noise to mesh in Geodesic Heat Method

Figure 5 shows the Geodesic Heat Method applied to the bunny mesh with added noise. Gaussian noise was added to each vertex of the mesh with a standard deviation affected by the bounding box dimensions in order to ensure sensical noise was added. It can empirically be seen that the added noise does not have a negative effect on the approximated geodesic distance.

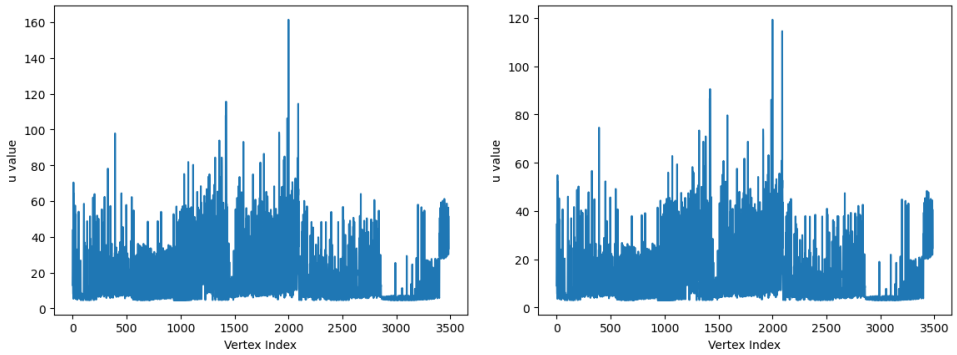


Figure 6: Left: Heat flow values on regular mesh Right: Heat flow values on mesh with noise added

Figure 6 shows the calculated heat flow values for the mesh with and without noise. Interestingly, we can see the values have changed but the order of them has stayed nearly the same. This means that the calculated gradients will have very similar directions but different magnitudes. As these gradients are then normalised this magnitude difference becomes a non issue, which could be the reason why adding noise does not have a large negative effect on the proposed method.

3.3 Effect of different Laplace-Beltrami Discretizations

To examine the effect of changing the Laplace-Beltrami discretization, I used `libigl` to implement the uniform discretization Laplace-Beltrami and set the mass matrix to the identity matrix.

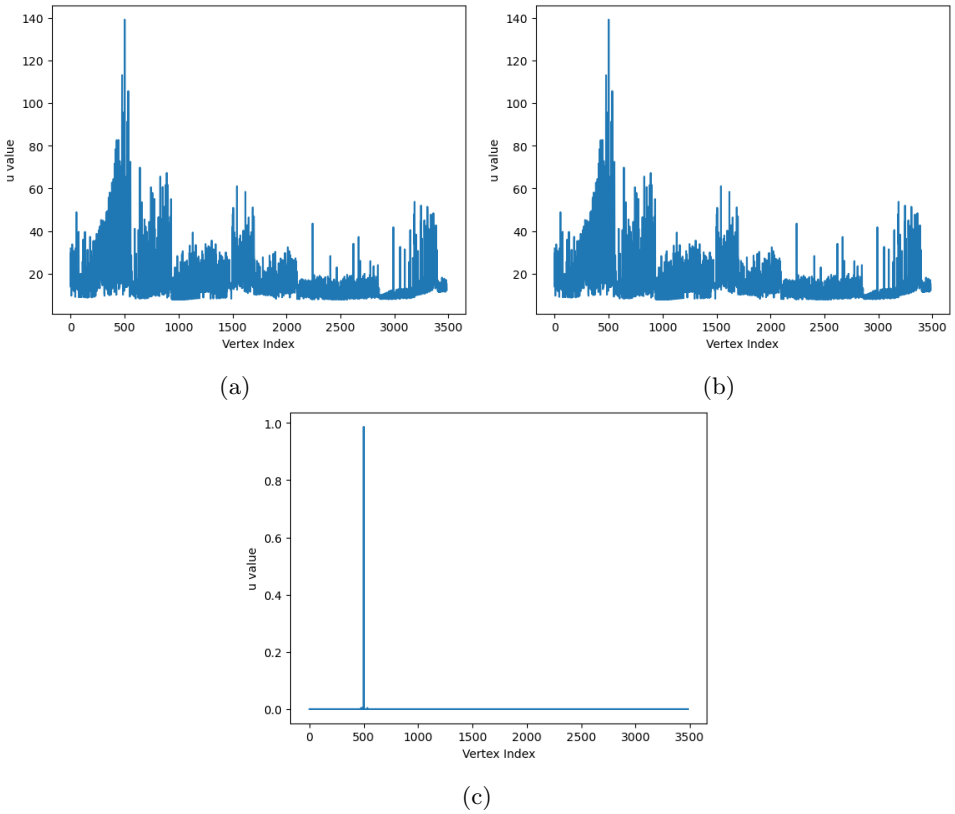
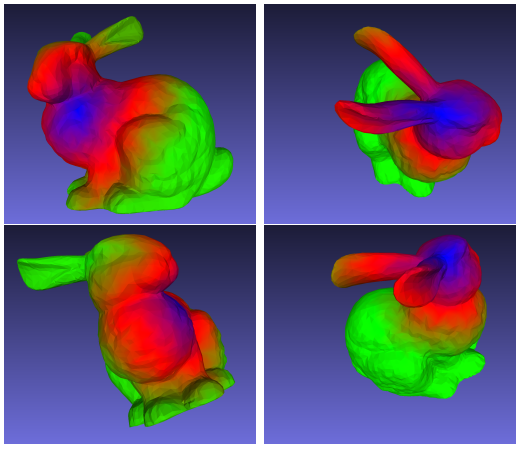


Figure 7: (a) Heat flow values Cotan Discretization with Voronoi Areas (b) Heat flow values Cotan Discretization with Barycentric Areas (c) Heat flow values on Uniform Discretization

Figure 7 shows the calculated heat values at each vertex for $t = h$ for the 2 different Laplace-Beltrami discretizations with Cotangent discretization using Voronoi (a) and barycentric (b) areas for the mass matrix. From here we can observe that the heat flow is simulated poorly when using the uniform Laplace-Beltrami (c) making the Geodesic Heat method fail.

Following from this investigation, I looked at whether increasing the time step when using the uniform Laplace-Beltrami would allow for the heat flow to be simulated.



vertex 500 vertex 2000

Figure 8: Uniform Laplace-Beltrami Discretization for the Geodesic Heat Method

Figure 8 shows the geodesic heat method using the uniform Laplace-Beltrami, with $t = 1$. We can observe how the accuracy of the approximation goes down where the distance doesn't look to spread radially from the source point as you would expect. I believe this is due to the oversimplification of the Laplace-Beltrami where the triangles are assumed to be regular. This means the irregularity of the mesh is not taken into account and the heat diffusion approximation is a lot worse.

4 Appendix

The libraries I used were:

- **python==3.9.18**
- **scipy==1.11.4**
- **trimesh==4.0.9**
- **libigl==2.5.1**
- **numpy==1.26.3**
- **matplotlib==3.8.0**

References

[1] K. Crane, C. Weischedel, and M. Wardetzky, "Geodesics in heat: A new approach to computing distance based on heat flow," *ACM Transactions on Graphics*, vol. 32, no. 5, p. 1–11, Sep. 2013. [Online]. Available: http://dx.doi.org/10.1145/2516971.2516977

[2] T. libigl Team, "libigl python bindings," https://libigl.github.io/libigl-python-bindings/, 2024.

[3] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[4] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.