Computing Geodesic Distance Based on Heat Flow

Hussein Houdrouge

Ecole Polytechnique - INF 555 Final Project

December 2017

Contents

1 Introduction

This project is the implementation of a new method that computes the geodesic distance based on the Heat Flow as it is proposed in [1]. The core idea of this method relies on the relation between the heat diffusion on the geodesic distance. It is like imagining a hot needle touches a single point on a surface and see how the heat diffuse over time, then works to extract the distance from this diffusion.
This method is described mathematically in the following manner (assuming a smooth manifold).

1. Integrate the heat flow $\dot{u} = \Delta u$. for some fixed t.
2. Evaluate the vector field $X = \frac{-\nabla u}{|\nabla u|}$
3. Solve the Poisson equation $\Delta \phi = \nabla . X$.

where ϕ encodes an approximation of the geodesic distance.
The next section will describe the application of this method on the discrete setting.

2 Method Description

The method to compute the geodesic distance will be described on a triangular mesh by the following steps.
Step one is computing the time that we want to compute the heat diffusion on it. the time t is estimated as the following formula.

$$t = h^2$$

where h is the average length of edges.
Step two is computing the heat flow on every vertex by solving the following equation.

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

where $A \in R^{|V| \times |V|}$ is a diagonal matrix of vertex areas. The vertex area is defined as one third of the area of the faces incident at that vertex. $L_c \in R^{|V| \times |V|}$ is the cotan operator and it is defined as follow.

$$L_c = \begin{cases} w_{ij} = -\frac{1}{2}(cot(\alpha_{ij} + cot(\beta_{ij})) & \text{if } i \text{ and } j \text{ are adjacent} \\ \sum_j w_{ij} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

where α_{ij} and β_{ij} are the opposing angle for the triangles that share the side (i, j). and δ_γ is a Kronecker delta, and u is the heat function on every vertex at a time t.
After solving the previous equation and computing the heat diffusion, step three is computing the gradient of the heat u in each face as follow.

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

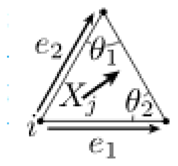
where the sum is taken over the vertices of the face. N is the unit norm of the face (counter clockwise). e_i is the vector opposing the vertex i whose heat value is u_i. A_f is the area of the face.
Step four is computing the normalised vector field X.

$$X = \frac{\nabla u}{|\nabla u|}$$

Step five is computing the integrated divergence associated with each vertex of the normalised vector field. This computation is done as the following.

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

The sum is taken on the incident faces for a given vertex i (as in the following figure).



Finally, the last step is solving the following Poisson equation.

$$L_c\phi = \nabla \cdot X$$

where ϕ approximates the geodesic distance.

3 Implementation Description

The implementation of this method has similar structure to the method proposed by [1] and described is the previous section. To set the right environment for the implementation of the heat method and for visualising the result, we used several libraries and platforms. We implemented the project using Java, and we used libraries such as Processing to visualise the meshes. In addition, we used the libraries JCG and TC provided in the TD of the INF555 to manipulate the triangular meshes. Moreover, to solve the linear system described in the previous section, we used Parallel COLT library.

Concerning the structure of the program, The program is divided into three main components. The Viewer, this component visualises the result and does the rendering of the mesh. The Heat Method algorithm implements the steps described in the previous section. Finally, an Utility Component helps the heat algorithm in solving elementary geometric and computational problems such as computing the angles and the area of a triangle. For instance, to compute the angles we used the fact that,

$$\langle u, v\rangle = \|u\|\,\|v\|\,cos\theta$$

where θ is the angle between u and v. And to compute the are of triangle, we used Heron's formula.

$$s = \frac{a+b+c}{2}$$

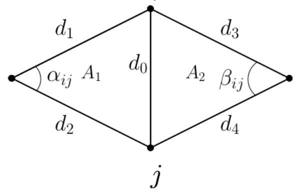
and

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

where A is the area of the triangle and a, b, and c are its sides.

Concerning the algorithmic approach. Most of the algorithms involved in solving the problem and the sub-problems are either solving a linear system or iterating over the mesh in order to compute the components of the linear systems.
For instance, the first problem is solving the equation $(A - tL_c)u = \delta_\gamma$. Dealing with this problem, require computing A, t, and L_c. Computing t is straight forward, it just iterating over the edges. Similarly computing A and L_c is straightforward. However, we used simple equivalent variant for computing the entries of L_c. we just used the fact that:

$$\frac{1}{2}(cot\alpha_{ij} + cot\beta_{ij}) = \frac{1}{8A_1}(d_0^2 - d_1^2 - d_2^2) + \frac{1}{8A_2}(d_0^2 - d_3^2 - d_4^2)$$

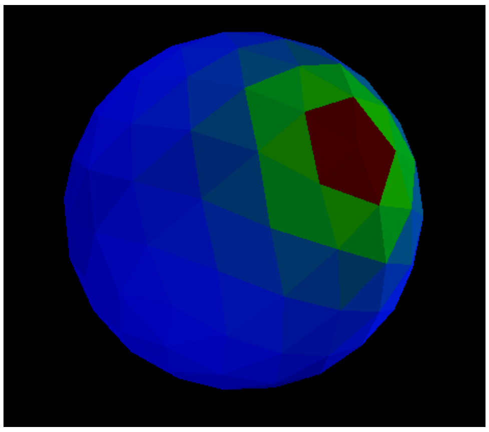


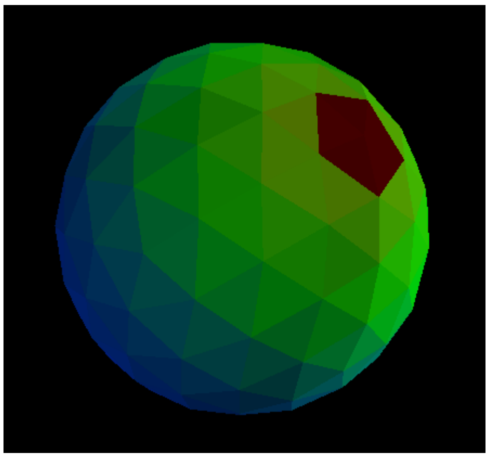
In solving this problem, we can notice that neither A, t, or L_c depends on the input source, and that L_c is used twice along the process of finding the geodesic distance. For this purpose, we computed this three

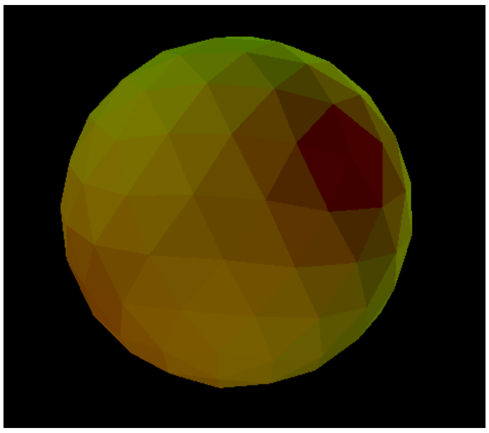
components one time at the beginning of the program. In addition, we computed the LU decomposition of $M = A - tL_c$ and L_c once at the initialisation of the program. Thus, we can use the LU decomposition of the matrix M to solve the following equation $Mu = \delta_\gamma$ and compute the heat flow in each vertex. The output of this problem will be used as the input of the next one.
The next sub problems are computing the gradient of the heat diffusion on each face, and the integrated divergence associated with each vertex. These two problems are also straightforward. They follow from their formula in the precious section, with addition to some elementary techniques to find angles normal vector, and to do some computations.
The last step which sum up all the previous work is to compute the approximate geodesic distance by solving the last equation $L_c\phi = \nabla \cdot X$. For this purpose, we used the already computed LU decomposition of L_c, and we solve for $\nabla \cdot X$.
At the last step we might get non stable solution for ϕ, the ϕ return is somehow shifted by a factor of ϵ, Thus we shift the value of ϕ by $-\epsilon$ to get an better result.

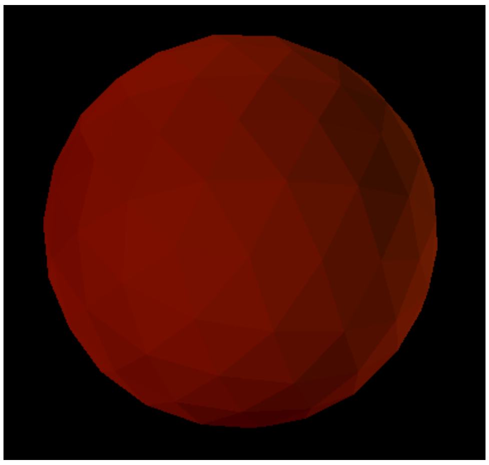
4 Results

We conducted our testing of the result over meshes that have number of faces of orders of 100 to the order of 10000. First, we will show how the heat diffuse over a sphere for different value of $t = mh^2$ where h is the average edge length and m is a constant.


(a) m = 1

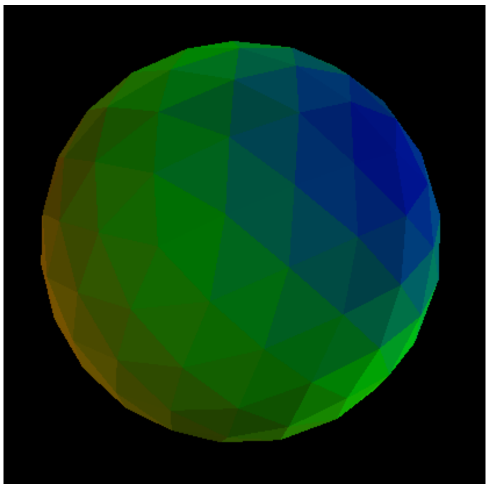

(b) m = 10

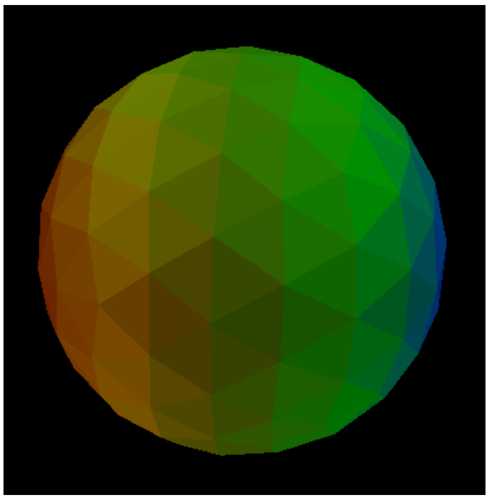

(c) m = 100


(d) m = 1000

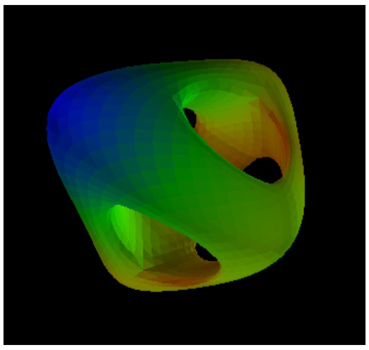
In these figures, the red area represents the hot faces which are the closest to the source, and blue one the most far which are colder than the others. These figures represent how the heat evolves over time, and for a long period of time it becomes constant over the whole surface. The following figure represent the geodesic distance on the sphere.

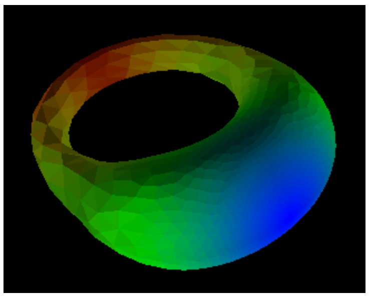
In these figures, the blue area are the closest to the source point and the red one are the most far. In the second picture, one can notice how the sphere is divided into multiple circular ranges of the same colour from the right to the left.

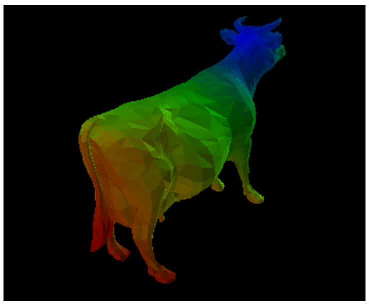

(a)

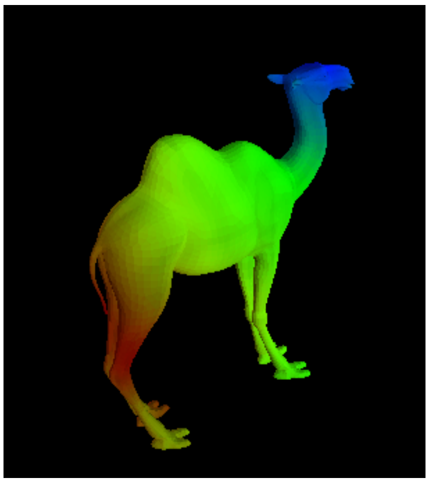

(b)

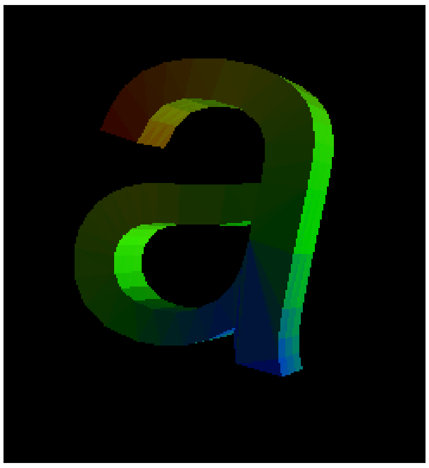
The following are the visualisation of the distances on different meshes of different sizes (The colour code is the same as before).


(a) High genus


(b) bague


(c) cow


(a) camel


(b) letter a

The rest of the results deal with performance of each part of the system.

Model	Triangles	Pre-computation	LU for M	LU for L_c	Computing X	Computing ϕ
Letter a	552	0.045	0.235	0.141	0.021	0.007
High genus	3328	0.357	4.241	2.892	0.102	0.241
Bague	5304	0.954	24.509	19.087	0.266	0.387
Cow	5804	1.034	49.19	39.134	0.503	0.501
Camel	19536	10.026	655.423	589.258	8.487	4.58

The table above shows the time spends on each step in order to compute the geodesic distance. The Pre-computation phase stands for constructing the matrix $M = A - t * L_c$ as we state in the previous sections. The other intervals are for the LU decomposition to solve the linear equations, and finally the X vector the integrated divergence of the heat, and ϕ that encodes the geodesics distance. One can notice that the bottleneck of the process is the computation of the *LU* decomposition for both matrices M and L_c. This bad performance is caused by the use of the Parallel COLT library. It turned out to be not effective as we expected. However, any better choice for the linear solver will increase drastically the performance of the process. For the other intervals, most of them are reasonable and quite efficient. However, it is possible to improve them. For instance, the pre-computation can be improved by computing A and L_c, and consequently M at the same time instead of computing them in a sequential manner even though the sequential manner look more cleaner and organised.

5 Conclusion

The Heat method is very straightforward method to approximate the geodesic distance. It is easy to implement and to follow. However, its efficiency depends on solving the linear equation. A better choice of the linear solver will lead to a significant improvement.

References

[1] Keenan Crane, Clarisse Weischedel, Max Wardetzky. ”Geodesics in Heat: A New Approach to Computing Distance Based on Heat”, ACM Trans. Graph. 32, 5, 2013, ACM, New York, NY, USA