INF574 - Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow

BOUILLÉ Aude - GLEISBERG Antoine

November and december 2022

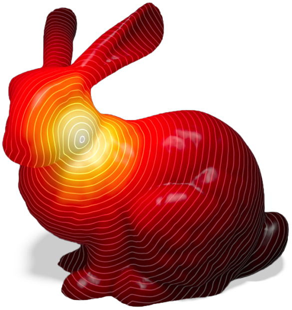
Introduction - Goal

1. The *heat method*
   - Overview
   - Outline of the algorithm
   - In greater detail
   - Precisions on our implementation

2. Current results

Conclusion - To do next

Goal



- Compute the geodesic distance to a specified subset (e.g., point or curve) of a given domain
- via the *heat method* (Crane et al., 2013)
- compare it to a naive method based on Dijkstra's algorithm
- on meshes

Original figure from: Keenan Crane, Clarisse Weischedel, Max Wardetzky. 2013. *Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow.* doi.org

https://doi.org/10.1145/2516971.2516977

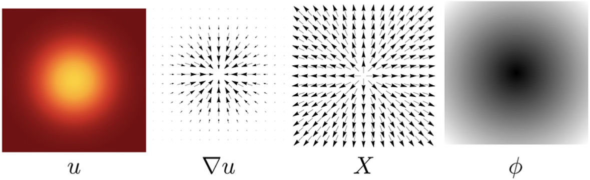
Overview

- Based on the solution of the heat equation
- Numerous advantages:
  - converges to the exact distance in the limit of refinement
  - efficient, since systems can be prefactored once
    and subsequently solved in near-linear time
    (based on solving standard linear elliptic systems)
  - can be applied to any type of geometric discretization
    (even point clouds)

Outline of the algorithm

As seen in Lecture 6:

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



Original figure from: Keenan Crane, Clarisse Weischedel, Max Wardetzky. 2013. *Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow*. doi.org

https://doi.org/10.1145/2516971.2516977

In greater detail

1. Integrate the heat flow $\dot{u} = \Delta u$ for time t

Compute L_C and A so that $L = A^{-1} L_C$

- L_C sparse matrix of Laplacian operator
- L_C sparse matrix of cotan operator
- A_f diagonal matrix of Voronoï areas

Initialise the temperature u^0 to 0 excepted at source point(s) to 1

For $\dfrac{t}{dt}$ steps, compute new temperature by solving one of

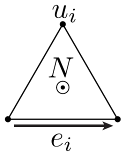
- $u^{k+1} = u^k + dt\, A^{-1} L_C$ (explicit)
- $(I - dt\, A\, L_C)\, u^{k+1} = u^k$ (semi-implicit)

In greater detail

2. Evaluate the vector field $X = -\frac{\nabla u}{|\nabla u|}$

Compute then normalize $\nabla u = \frac{1}{2A_f} \sum_i u_i \left(N \times e_i \right)$

- ∇u gradient in a given triangle
- A_f area of the face (doesn't need to be computed)



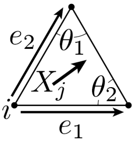
In greater detail

3. Solve the Poisson equation $\Delta\phi = \nabla \cdot X$

Solve the symetric Poisson problem $L_C\phi = b$

- L_C sparse matrix of cotan operator
- φ vector of geodesic distances
- b vector of (integrated) divergences of the normalized vector field X

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

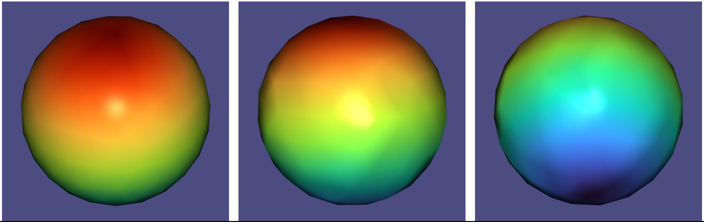


Precisions on our implementation

- Half-edge data structure (`HalfedgeDS`)
  move locally on the mesh in an efficient way
- Sparse Matrices (`SparseMatrix`)
  only save the non-zero entries
- Built-in solver (`SimplicialCholesky`)
  solve easily large sparse linear systems
- dt: max length of edges (after bringing mesh to $[0,1]^3$)
- t: by rule of thumb
  (when heat seems to have propagated enough)

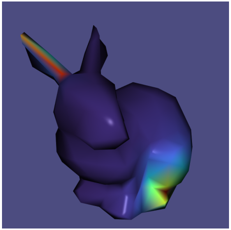
Positive side

- *heat method* implemented
- fast
- work on "simple" meshes (ex: the sphere mesh)
  - enough and not too much vertices (ex: 300)
  - similar lengths of edges



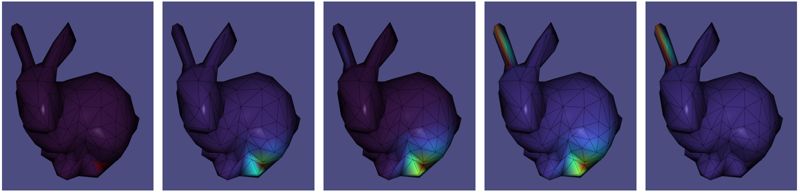
```
Total computing time from the loading of the data to the obtention of the geodesic distance:15.0181 s
Including computing time for a total of 1000000 steps of heat diffusion: 14.983 s
On a mesh of 162 points
```

Negative side

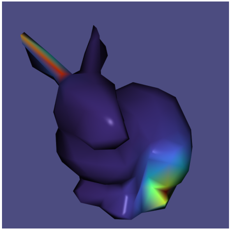


On other meshes:

- solution looks wrong
- even the heat diffusion can look abnormal



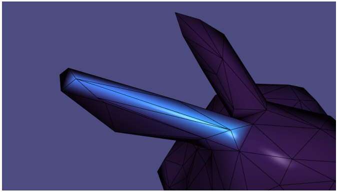
Negative side

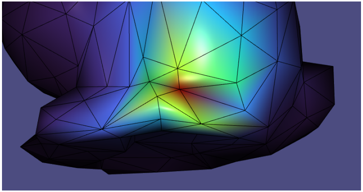


On other meshes:

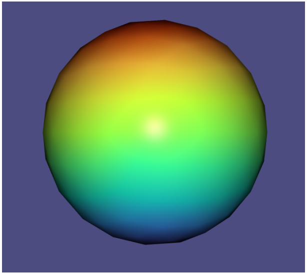
- solution looks wrong
- even the heat diffusion can look abnormal

Uneven lengths of edges on the bunny mesh





To do next



- Obtained:
  seemingly functionnal *heat method*
  on some simple meshes
- Make it work for more meshes (arbitrary ones)
- Maybe change dt
- Find the right t
- Implement a naive method based on Dijkstra's algorithm
- Compare them rigorously