Picture: [CrWeiWar]

The Heat Method for Distance Computation

By Keenan Crane, Clarisse Weischedel, and Max Wardetzky

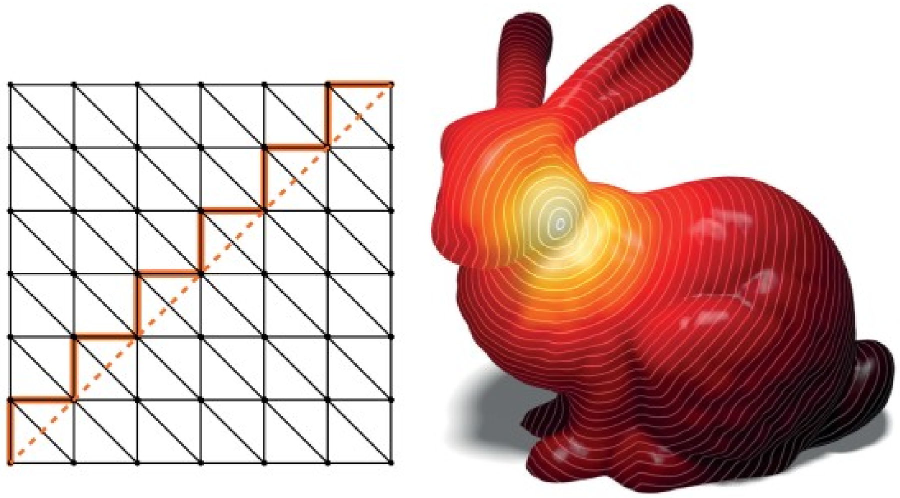
Seminar: Recent Advances in 3D Computer Vision

Talk by Philipp Kretz

Abstract

We introduce the *heat method* for solving the single- or multiple-source shortest path problem on both flat and curved domains. A key insight is that distance computation can be split into two stages: first find the direction along which distance is increasing and then compute the distance itself. The heat method is robust, efficient, and simple to implement since it is based on solving a pair of standard sparse linear systems. These systems can be factored once and subsequently solved in near-linear time, substantially reducing amortized cost. Real-world performance is an order of magnitude faster than state-of-the-art methods, while maintaining a comparable level of accuracy. The method can be applied in any dimension, and on any domain that admits a gradient and inner product—including regular grids, triangle meshes, and point clouds. Numerical evidence indicates that the method converges to the exact dis-

Figure 1. In contrast to algorithms that compute shortest paths along a graph (left), the heat method computes the distance to points on a continuous, curved domain (right). A key advantage of this method is that it is based on sparse linear equations that can be efficiently prefactored, leading to dramatically reduced amortized cost.



contents

The heat method is a ...

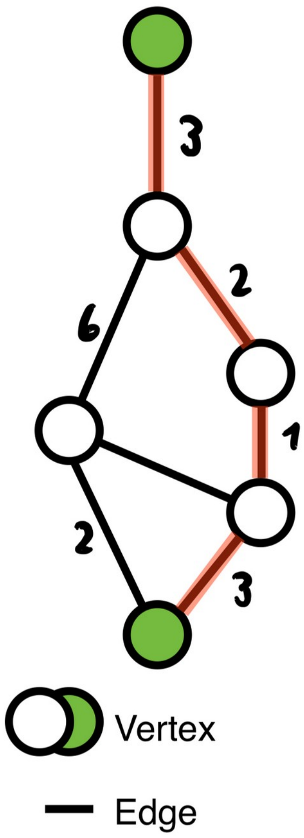
“[...M]ethod for solving a […] shortest path problem”. It is …

- “[…] robust, efficient, and simple to implement […]”.
- “[…] faster […]”
- “[Applicape in any dimension and] domain which admits a gradient and inner product […]” (SOURCE paper)

Source: [CrWeiWar]

What are shortest path problems?

- Find the shortest path (distance) in a weighted graph
- Graph is structure comprising nodes connected by edges
- Weighted graph does considere different weightings (distances) for each edge
- Typicall applications: street map, bus lines or flight plans [MaTUM]



Picture: [CrWeiWar]

And visualization:

- Reconstruct a scan/image/data consisting of regular or irregular grids or point clouds.
- Visualize animated elements (e.g. software by PIXAR) [YTvideo]

Picture: [CrWeiWar]

And visualization:

- Reconstruct a scan/image/data consisting of regular or irregular grids or point clouds.
- Visualize animated elements (e.g. software by PIXAR) [YTvideo]

Basic principle: Varadhan's Formula

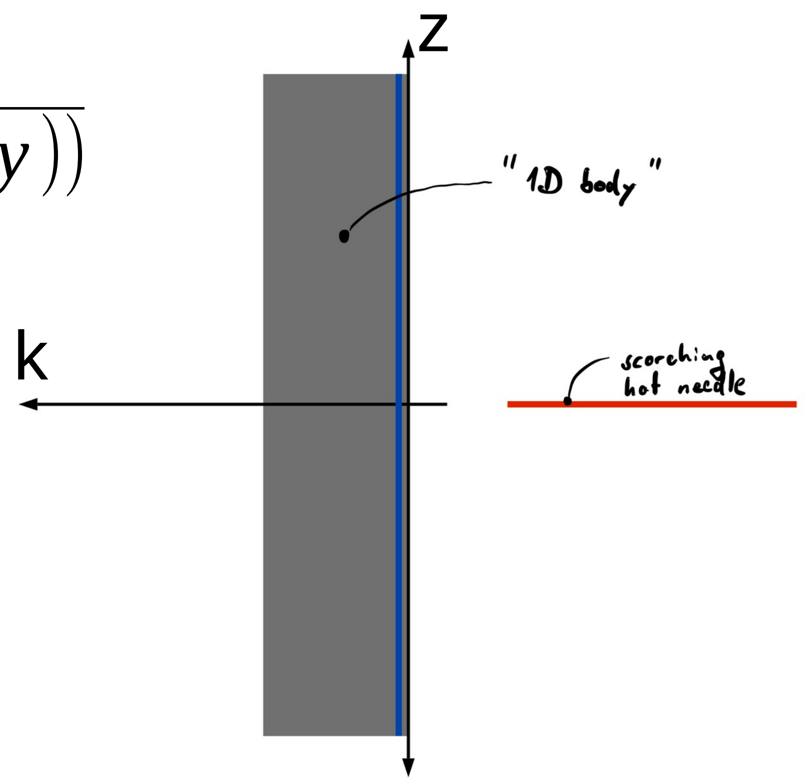
$$\Phi(x,y)=\lim_{t\to 0}\sqrt{-4*t+\log(k_{t,x}(y))}$$

t : time

k : temperature

Φ : distance

x,y : points in domain



Basic principle: Varadhan's Formula

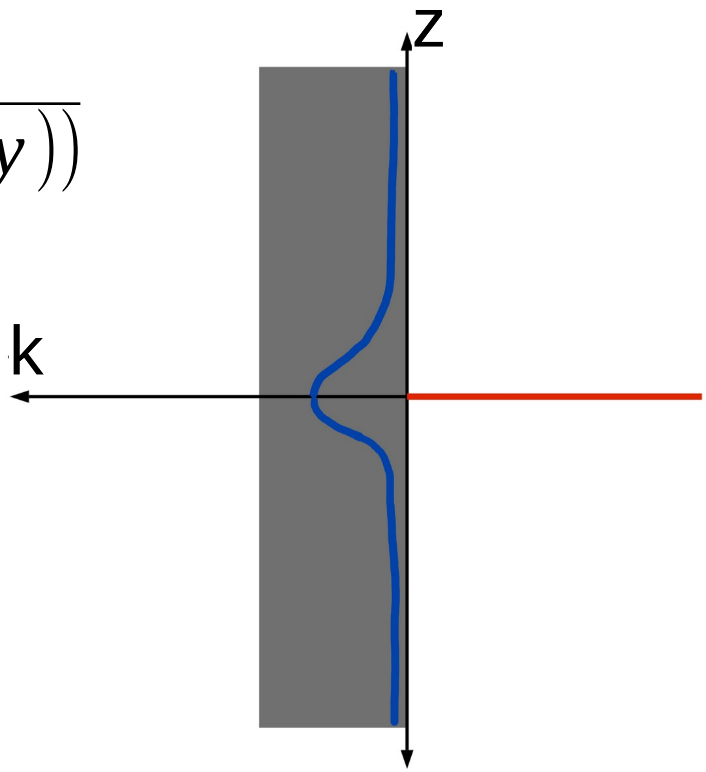
$$\Phi(x,y)=\lim_{t\to 0}\sqrt{-4*t+\log(k_{t,x}(y))}$$

t : time

k : temperature

Φ : distance

x,y : points in domain



Basic principle: Varadhan's Formula

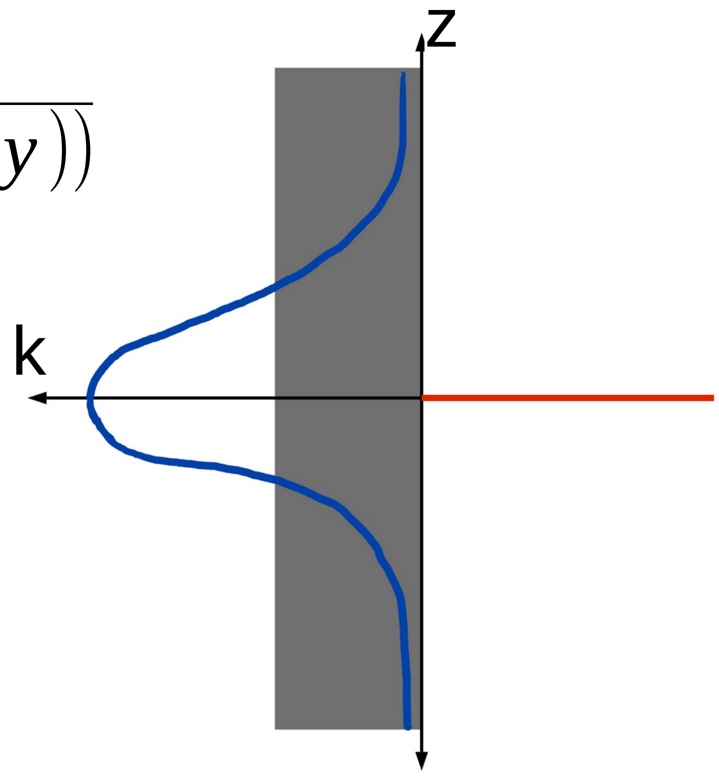
$$\Phi(x,y) = \lim_{t \to 0} \sqrt{-4 * t + \log(k_{t,x}(y))}$$

t : time

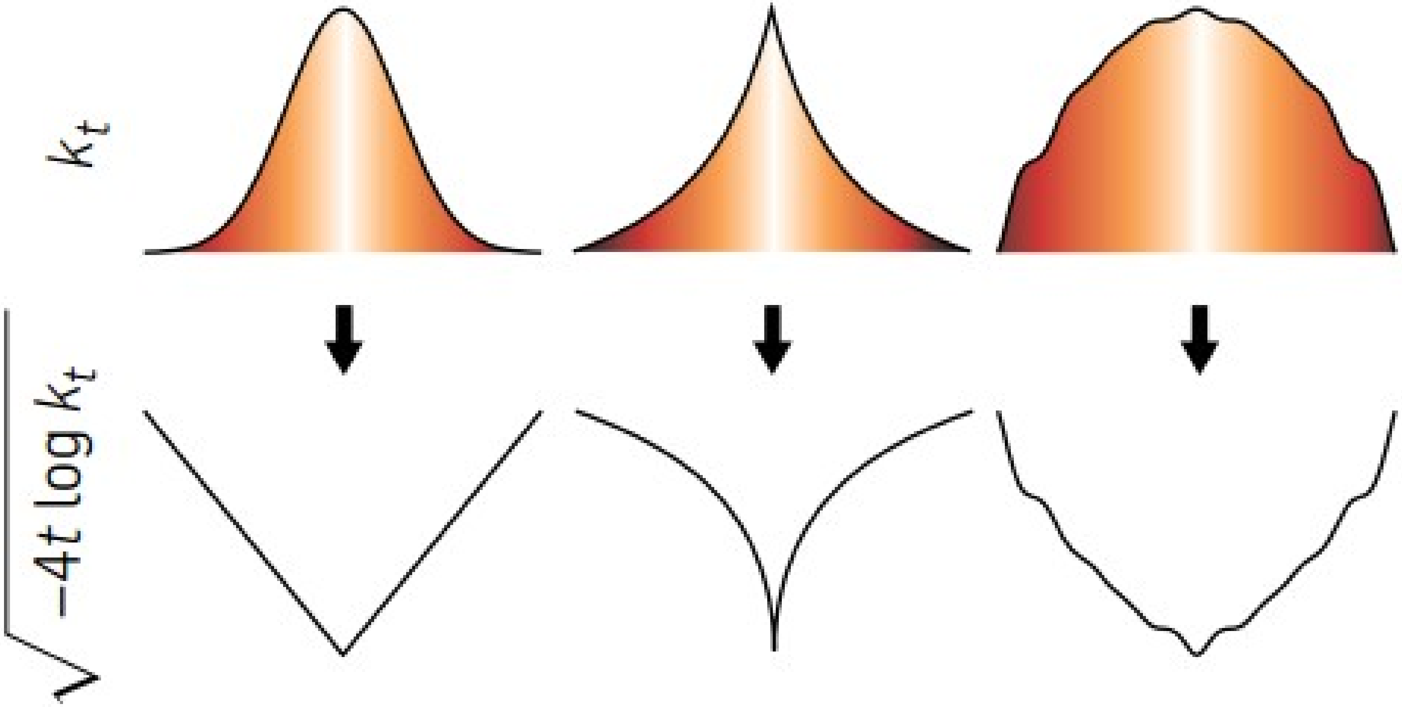
k : temperature

Φ : distance

x,y : points in domain



Basic principle: Varadhan’s Formula

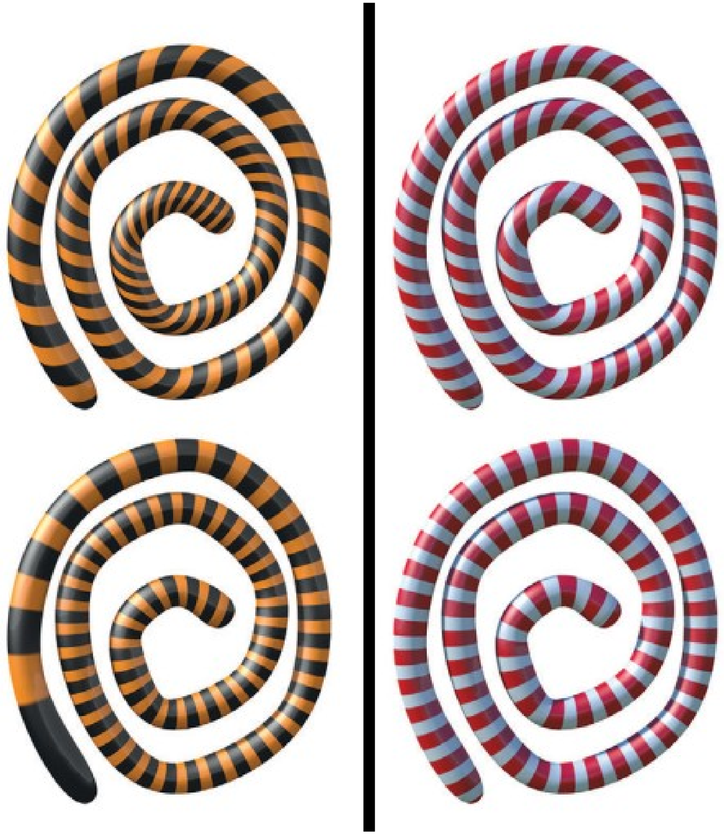


Picture: [CrWeiWar]

Approximation error

Varadhan with t1

Varadhan with t2 > t1



Heat method

Smoothed distance function

Picture: [CrWeiWar]

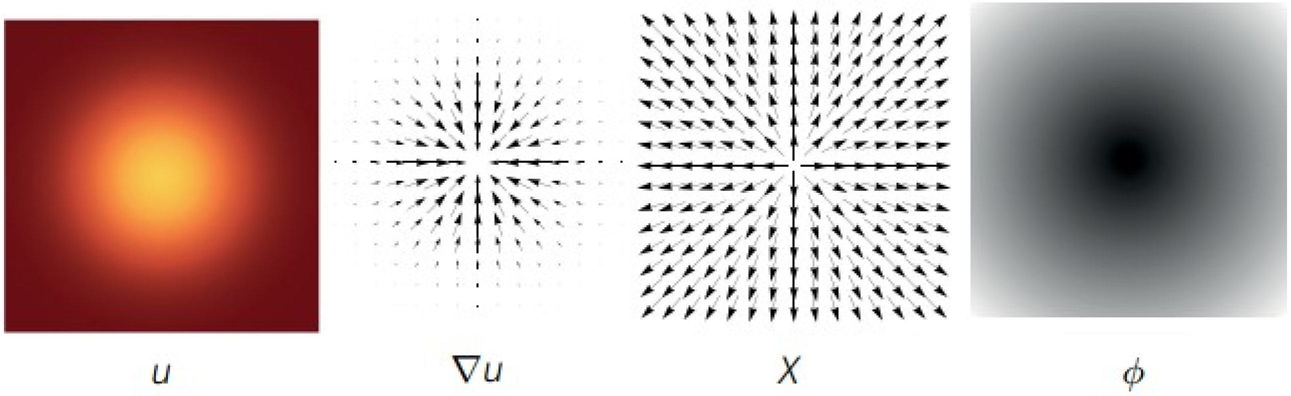
Eikonal Equation

$$|\nabla \Phi| = 1$$

Steps of the 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 $X = -\nabla u_t / |\nabla u_t|$.
III. Solve the Poisson equation $\Delta\phi = \nabla \cdot X$.



[CrWeiWar]

Time discretization

- Backward/Implicit Euler
- Solve:

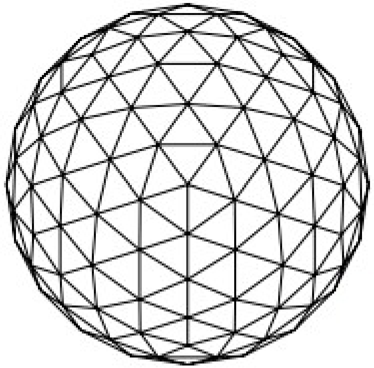
$$\left(id - t * \Delta\right) u_t = u_0$$

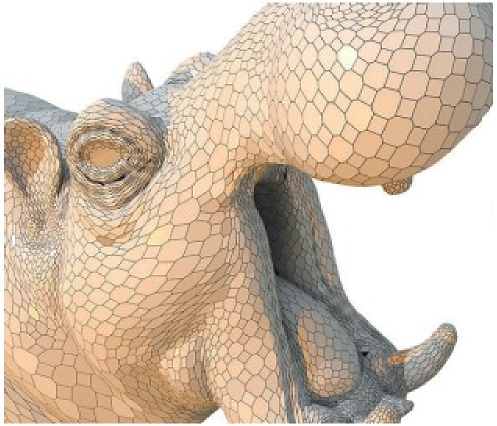
- Example: Time step size from empirical experiments: $t := h^2$

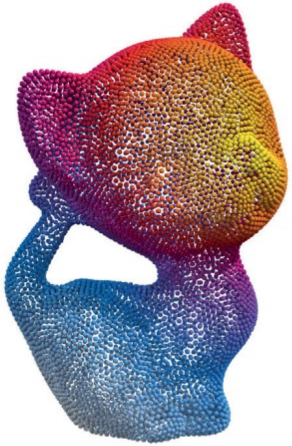
[CrWeiWar]

Spatial discretization

- It depends on the application
- It depends on the type of domain / data







Picture: [CrWeiWar]

Examples

Picture: [CrWeiWar]

Examples

Picture: [CrWeiWar]

Performance

- Sparse linear systems in step 1 and step 3 can be prefactored
  - Most of the computational work can be reused

		Heat method				Fast marching			
Model	**Triangles**	**Precompute (s)**	**Solve**	**Max error (%)**	**Mean error (%)**	**Time (s)**	**Max error (%)**	**Mean error (%)**	**Exact time (s)**
Bunny	28k	**0.21**	**0.01s (28x)**	3.22	**1.12**	0.28	**1.06**	1.15	0.95
Isis	93k	**0.73**	**0.05s (21x)**	1.19	**0.55**	1.06	**0.60**	0.76	5.61
Horse	96k	**0.74**	**0.05s (20x)**	1.18	**0.42**	1.00	**0.74**	0.66	6.42
Kitten	106k	**1.13**	**0.06s (22x)**	0.78	**0.43**	1.29	**0.47**	0.55	11.18
Bimba	149k	**1.79**	**0.09s (29x)**	1.92	0.73	2.62	**0.63**	**0.69**	13.55
Aphrodite	205k	**2.66**	**0.12s (47x)**	1.20	**0.46**	5.58	**0.58**	0.59	25.74
Lion	353k	**5.25**	**0.24s (24x)**	1.92	0.84	10.92	**0.68**	**0.67**	22.33
Ramses	1.6M	**63.4**	**1.45s (68x)**	0.49	**0.24**	98.11	**0.29**	0.35	268.87

Table: [CrWeiWar]

Accuracy

Exact:

Fast Marching:

Heat Method:

Picture: [CrWeiWar]

Robustness

- The heat method is really robust:
  - Unconditionally stable time discretization
  - Elliptic formulation (not hyperbolic)

Picture: [CrWeiWar]

Whats the progress?

- There are papers: [CrWeiWar] [ShaSolCr]
- There are a lot of applications [YTvideo]
- There are Implementations for different coding languages [CodeInfo]
- Since the method is build on linear PDEs, it immediately takes advantage of innovation in solving PDEs [CrWeiWar]

Summary

- The heat method is not only a theoretical study, but a tool, which is succesfully and widely used in practice
- The heat method is faster and roughly as precise as its competitors
- It is robust and simply to implement
- There are a lot publications and interesting information available, but there is also further research ongoing

[CrWeiWar]

Index / sources

[CrWeiWar] 10.10.2022 – 2:41pm:

https://www.cs.cmu.edu/~kmcrane/Projects/HeatMethod/paperTOG.pdf

[ShaSolCr] 10.10.2022 – 2:46pm:

http://www.cs.cmu.edu/~kmcrane/Projects/VectorHeatMethod/paper.pdf

[CodeInfo] 10.10.2022 – 2:47pm:

https://www.cs.cmu.edu/~kmcrane/Projects/HeatMethod/

[YTvideo] 10.10.2022 – 2:50pm:

https://www.youtube.com/watch?v=4IZ-ykGnIRc

[MaTUM] 10.10.2022 – 2:52pm:

https://algorithms.discrete.ma.tum.de/

[ViTUM] 10.10.2022 – 2:55pm:

https://vision.in.tum.de/research/shape_analysis

Questions?