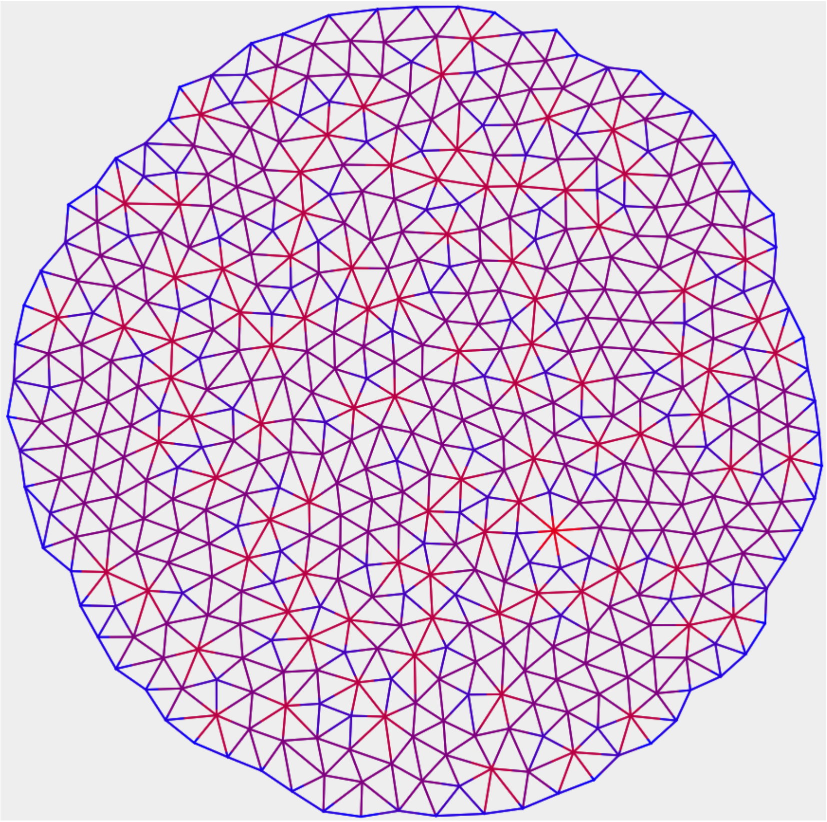
Canning for Amorphous Blob Computing

TER report

Auteur :
Lucas Labouret
M1 QDCS, Université Paris-Saclay
lucas.labouret@universite-paris-saclay.fr

Encadrant :
Frédéric Gruau
LISN
frederic.gruau@universite-paris-saclay.fr



Contents

A What is blob computing ?

This TER is part of a long term research project about the concept of ”blob computing”[1]. To understand the exact subject of this TER, it is important to first introduce what blob computing means and how it works.

The ultimate goal of the project is to develop a new paradigm of computation that is inherently arbitrarily scalable through the use of *space* as a resource. This kind of computation is becoming increasingly useful, for example with the recent development of swarm robotic[2].

We aim to achieve this through the use of arbitrarily many processing elements (PEs) distributed through space and locally connected. These PEs, which can be severely lacking in power on their own, create a computing medium where virtual ”blobs” can form and evolve. These blobs are the main primitive we use for making computations.

A.1 Definition and structure of a computing medium

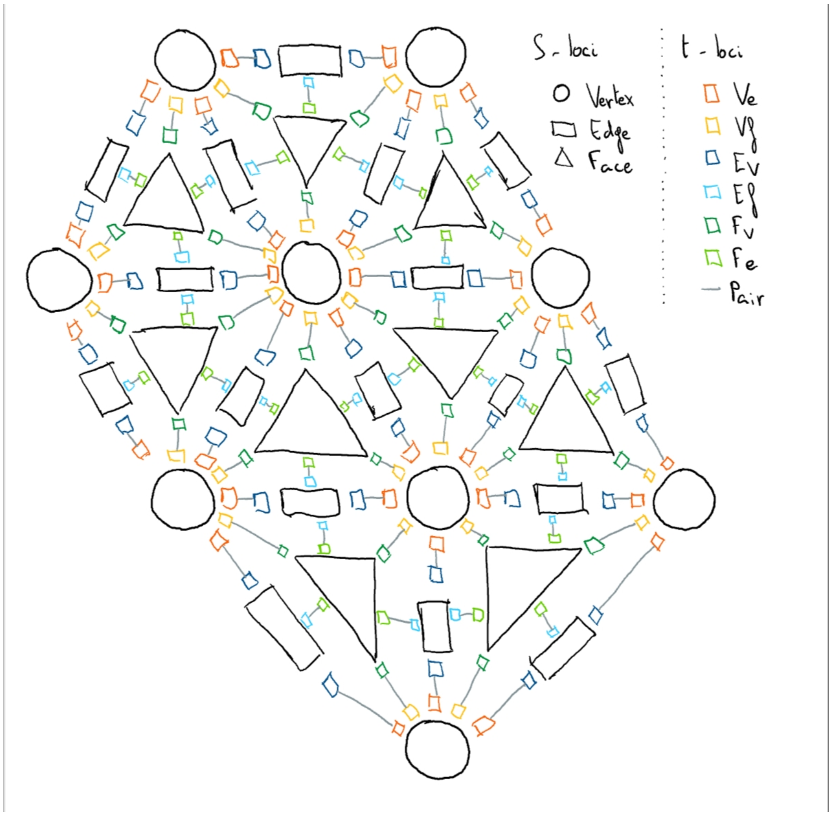


Fig. 1: Example structure of a medium.

A computing medium is a weakly Delaunay-triangulated planar graph where each vertex, edge, and face is associated with a processing element capable of executing minimal computation, storing information, and communicating with its neighbors.

Vertices, edges and faces are called ”simplicial loci” (or ”S-loci” for short). They determine the overall structure of their medium. Each S-locus controls a set of ”transfer loci” (or ”t-loci”). T-loci come in pairs that handle the communication between two S-loci. For example, if a vertex was connected to an edge, there would be a pair of t-loci (one belonging to the vertex and the other belonging to the edge) between them.

Obviously, there are 3 types of S-loci : Vertex, Edge, Face. T-loci are divided into 6 types :

- Ve linking a Vertex to an Edge
- Vf linking a Vertex to a Face
- Ev linking an Edge to a Vertex
- Ef linking an Edge to a Face
- Fv linking a Face to a Vertex
- Fe linking a Face to an Edge

Figure 1 show an example of how all these loci are put together to form a medium.

A medium is said to be crystalline if a repeating structure emerges in the placement if its vertices. It is amorphous when no such structure emerges. Additionally, it is homogeneous if the average point density is approximately constant and there are no “holes” or “clusters”, and it is isotropic if the directions of the edges of the medium follow a uniform distribution over $[0, \pi[$. This report mainly focuses on amorphous homogeneous isotropic (AHI) media.

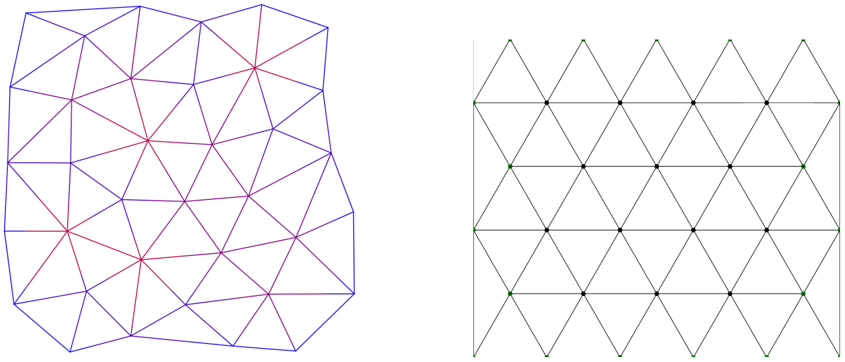


Fig. 2: An AHI medium (left) VS a crystalline hexagonal medium (right).

For now, we consider PEs to be immobile and synchronous, although lifting these two constraints will probably be the subject of future research.

A blob of a given medium M is a connected sub-graph M' of M such that all the vertices of M' share a common property (e.g.: all their values are even, or all their values are 1, etc.). Depending on the program, these blobs are able to grow, shrink, merge, divide and more, which is the computational basis of blob computing.

A.2 An example of computation : the (discrete) Voronoï diagram

The article "A parallel data-structure for modular programming of triangulated computing media"[3] develops tools to program on computing media. As an example, it shows how to build a Voronoï diagram from a set of "seeds" in a medium.

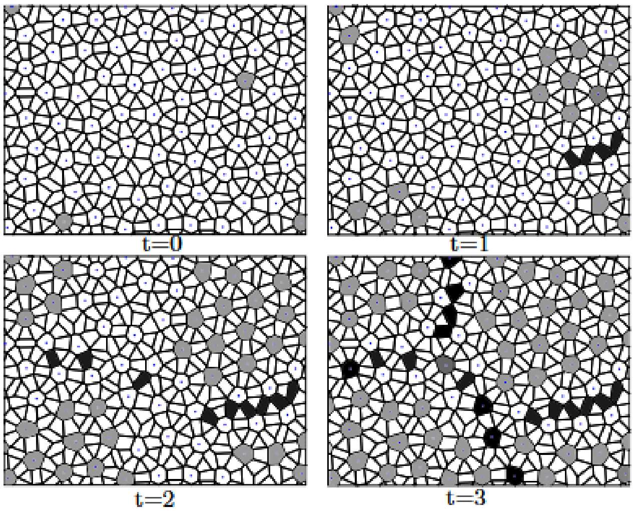


Fig. 3: Computation of a discrete Voronoï diagram.
Cells marked with a dot are the vertices of the medium. Gray cells form blobs. Black cells mark the frontiers between blobs. At t=0, blobs only cover the seeds. At t=3, blobs cover the Voronoï regions of the medium.

The basic idea is the following : we start with a set of "seeds" distributed over the vertices of the network. Each seed is now the center of a blob. Then, at each synchronous step, the blobs will spread to their adjacent vertices if this wouldn't cause them to merge with another blob, until no blob can spread further. It can be implemented with very few types of local operations that are applied simultaneously to every S-locus in the network :

- Multi-cast
  The S-locus copies its value into its t-loci of a given type.

- Rotation
  Since t-loci are spatially arranged around their S-loci, we can define a

clockwise and counter-clockwise neighbor. T-loci can perform logical operations on their and their neighbors' values.

- Reduction
  The S-locus stores the result of a commutative and associative operation applied to the values of its t-loci of a given type.

- Transfer
  Each of the S-locus' t-loci exchanges its value with the value of its pair.

In fact, clever association of these primitives allows to program everything a computing medium is capable of, such as sorting arrays of integer or multiplying matrices[4].

A.3 Objective : Cannings

Up until now, Gruau mainly focused on hexagonal crystalline media. He built a plateform allowing efficient simulation of blob computation on classical hardware using SIMD[3][5].

The point of this TER is to allow the use of SIMD to optimize the simulation of AHI media as well. We do this through the development of the notion of "canning". Given a medium M, a canning is a set of 9 injective maps which assign some integer coordinates to each locus of M :

1. $\{\text{Vertex}_M\} \mapsto \mathbb{N}^2$
2. $\{\text{Edge}_M\} \mapsto \mathbb{N}^3$
3. $\{\text{Face}_M\} \mapsto \mathbb{N}^3$
4. $\{\text{Ve}_M\} \mapsto \mathbb{N}^3$
5. $\{\text{Vf}_M\} \mapsto \mathbb{N}^3$
6. $\{\text{Ev}_M\} \mapsto \mathbb{N}^4$
7. $\{\text{Ef}_M\} \mapsto \mathbb{N}^4$
8. $\{\text{Fv}_M\} \mapsto \mathbb{N}^4$
9. $\{\text{Fe}_M\} \mapsto \mathbb{N}^4$

Through misuse of language, the term "canning" will designate either this set of maps, or the algorithm used to build it.

The objective is twofold : firstly, we want to find a way to measure the efficiency of a canning on a given medium. Secondly, we want to create a canning that is efficient on most -if not all- media.

B Preliminary work

In this section, I detail some work I did last year during another internship with Gruau to generate AHI media. While it isn't the main focus of this year's interniship, it is what made it possible in its current form.

B.1 Delaunay Triangulation

B.2 Farthest Point Optimization

The Farthest Point Optimization algorithm [6] allows for the construction of the irregular yet homogeneous sets of points that we need for blob computing. It yields better results than more common algorithm like Poisson-Disk sampling or Lloyd algorithm, and fits naturally into the project as it relies heavily on the Delaunay triangulation.

The implementation of FPO in my case wasn't straightforward. This is because the article applied the algorithm to sets of points in the unit torus, while the media use bounded subsets of the euclidean plane instead. In particular, I had to adapt the algorithm to work around the existence of borders with no predetermined shape, and that can contain fixed points that cannot be moved even though they can still influence the placement of other points.

1 GUI

A large portion of the TER was focused on the development of a software equipped with a GUI that allows the user to

1. Randomly generate new media, save them, and load them.
2. Visualize different aspects of the media.
3. Compute and visualize different cannings.

If developing a GUI for this application ended up being time-consuming, it was incredibly helpful all around while debugging everything else. Overall it was probably a net positive on development time alone, and that is still neglecting the advantage of being able to see what i am doing.

2 Borders and media

2.1 Choice of border

2.2 Different types of medium

3 Canning and evaluation

3.1 Partial cannings

3.2 Total cannings

3.3 Evaluation

References

[1] Frédéric Gruau et al. "BLOB computing". In: *Proceedings of the 1st Conference on Computing Frontiers.* CF '04. Ischia, Italy: Association for Computing Machinery, 2004, 125–139. ISBN: 1581137419. DOI: `10.1145/977091.977111`. URL: `https://doi.org/10.1145/977091.977111`.

[2] Iñaki Navarro and Fernando Matía. "An Introduction to Swarm Robotics". In: *International Scholarly Research Notices* 2013.1 (2013), p. 608164. DOI: `https://doi.org/10.5402/2013/608164`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.5402/2013/608164`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.5402/2013/608164`.

[3] Frédéric Gruau. "A parallel data-structure for modular programming of triangulated computing media." In: *Natural Computing* 22.4 (2023), pp. 753–766. ISSN: 1572-9796. DOI: `10.1007/s11047-022-09906-1`. URL: `https://doi.org/10.1007/s11047-022-09906-1`.

[4] Frédéric Gruau. "Artificial cellular development in optimization and compilation". In: *Towards Evolvable Hardware.* Ed. by Eduardo Sanchez and Marco Tomassini. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 48–75. ISBN: 978-3-540-49947-3.

[5] Frédéric Gruau. *platform-CA2.* URL: `https://github.com/fredgruau/platform-CA2`.

[6] Thomas Schlömer, Daniel Heck, and Oliver Deussen. "Farthest-point optimized point sets with maximized minimum distance". In: *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics.* HPG '11. Vancouver, British Columbia, Canada: Association for Computing Machinery, 2011, 135–142. ISBN: 9781450308960. DOI: `10.1145/2018323.2018345`. URL: `https://doi.org/10.1145/2018323.2018345`.