CSE306 Project 2: Fluid simulator

Romain Puech

May 2024

1 Introduction

The objective of this project was to implement several algorithms and techniques in computational geometry and optimal transport to create a 2D semi-discrete optimal transport fluid simulator with free surfaces. This involved coding the Voronoi Parallel Linear Enumeration algorithm in 2D, based on the Sutherland-Hodgman polygon clipping algorithm, and a semi-discrete optimal transport in 2D using L-BFGS. The detailed implementation and results are discussed in the following sections.

Throughout the report, the rendering times are reported for a laptop equipped with an Intel Core i7 8th gen. The code is compiled with the `-O3` flag.

1.1 Lab 6: Polygon clipping and Voronoï diagrams

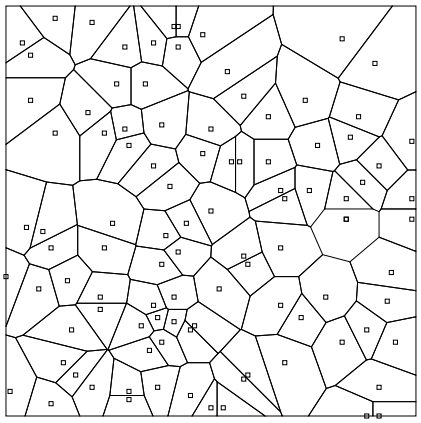
1.1.1 Voronoï Parallel Linear Enumeration algorithm

We implemented the Voronoï Parallel Linear Enumeration algorithm in 2D. This algorithm generates a Voronoï diagram by computing independently the Voronoï cell of each site. It is thus parallelizable. To compute each Voronoï cell, we start with a huge square representing the full space, and clip it along by the bisectors between the considered site and every other site. To clip along the bisector, I generated a square bigger than the Voronoï cell having the bisector as one of the edges, and used the Sutherland Hodgman polygon clipping algorithm on this square and the cell.

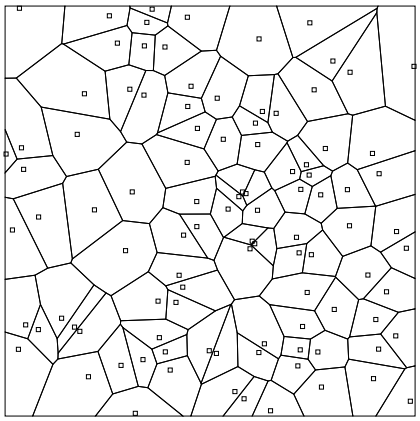
1.1.2 k-nearest-neighbor optimization

Instead of clipping each cell along every bisector, leading to a quadratic complexity, we start by finding the k nearest neighbors of the cell (in linear time for k fixed), and clip along the bisectors corresponding to these neighbors only. At any point, if the distance between the site and the neighbor is bigger than twice the distance between the site and any vertex of the Voronoï cell, we know that further cells will not contribute to the cell. If this criteria is not reached, we double k and start over. We start with $k = 7$.

Examples are provided in Figure 1.



(a) Voronoï diagram for 100 points generated uniformly at random. Rendered in about 20 miliseconds



(b) Voronoï diagram for 100 points generated following a normal distribution. Rendered in 30 miliseconds

Figure 1: Voronoï diagrams. The sites are represented by small squares.

1.2 Lab 7: Power diagrams and weights optimization

1.2.1 Power diagrams

We implemented a weighted variant of the Voronoï diagram named power diagram. A power cell of weight w_i is a Voronoï cell, where the distance between a point of the cell and the site is taken between the point and a circle centered on the site and of radius w_i. The algorithm to generate a power cell is the same as to generate the Voronoï cell, up to a modification on the points parallel to the bisectors and tangent to the cell.

1.2.2 Weight optimization using L-BFGS

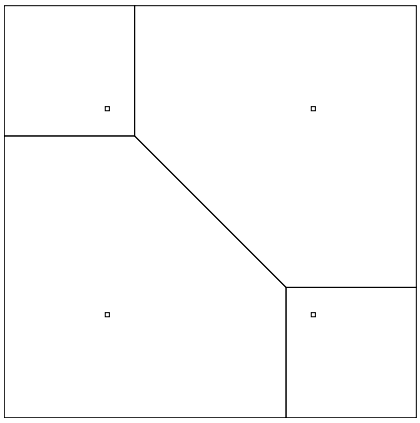
The weight of each cell can be adjusted so that each cell takes up a given share of the total area. The corresponding optimization problem can be modeled as a semi-discrete optimal transport problem and is detailed in the lecture notes section 4.4.4. We solved this optimization problem using the library libLBFGS[1].

Examples are given in the following figures.

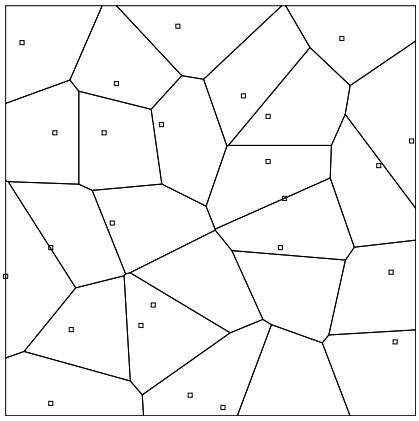
1.3 Lab 8: Computational fluid dynamics

We used the elements coded in Lab 6 and Lab 7 to implement a semi-discrete optimal transport fluid simulator with free surfaces. To this aimed, we generated N particles and optimized over $N+1$ variables, taking a weight for the air in the power diagram. We clipped the power cells of the fluid particles by an approximation of a disk of radius $\sqrt{W_i - W_{air}}$. This allowed us to simulate

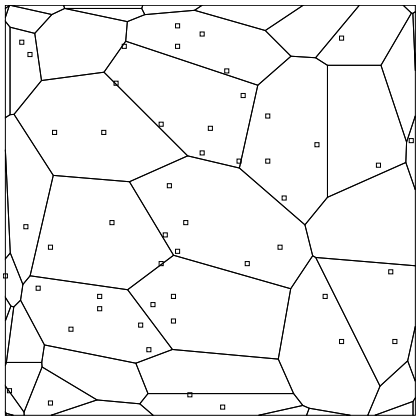
[1] GitHub: https://github.com/chokkan/liblbfgs/tree/master



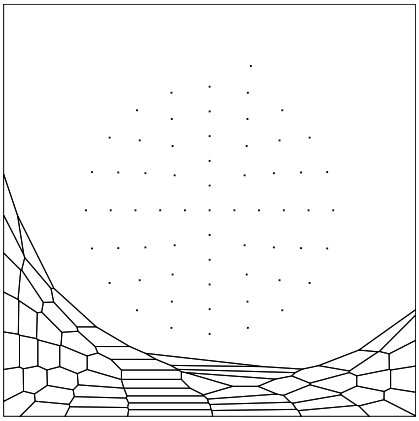
(a) Power diagram optimized for the area repartition $(0.4, 0.4, 0.1, 0.1)$. Generated in about 10 milliseconds



(b) Power diagram with weights so that each cell has the same are. 25 cells generated in about 40 milliseconds



(a) Uniform distribution of 25 sites. Area proportional to $exp(-\|C - y_i\|^2/0.05)$ with C the center point. Generated in about 40 milliseconds.



(b) Power diagram optimized so that the topmost cell takes up 0.8 of the total area, and the others share 0.2. Generated in about 100 milliseconds.

an infinite number of air particles, as explained in the lecture notes section 5.4. We adjusted our optimization function accordingly and used LBFGS to solve the optimization problem. Once the power diagram of the fluid cells was generated, we updated the fluid particle positions and velocities using the Gallouët Mérigot scheme (a discretization of the motion's differential equation). We took the spring force and the gravity into account. We re-iterated this two-time procedure, saving one frame at a time. When a particle hit a side of the bounding square (floor included), I considered that it bounced while keeping 80% of its speed. Some frames of the finale simulation are presented in Figure 4

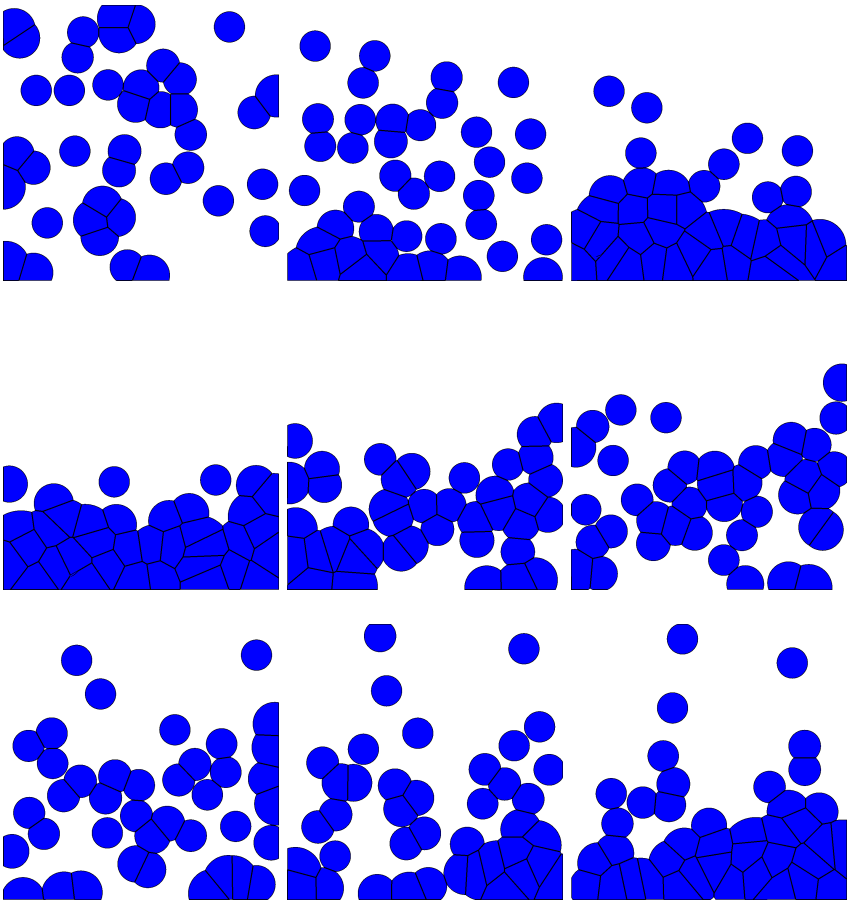


Figure 4: Fluid simulator with bouncing. Fluid area of 0.4, 40 particles, mass of 250, time step of 0.05 seconds, 200 frames. Rendered in about 10 minutes.