Geometry processing project
CSE306

Alexandru-Nicolae Serban

1 Introduction

Geometry processing is a field that deals with the transformation of geometric shapes and surfaces into digital models and stays at the core of virtual reality and of computer graphics. According to [2], one of the most famous problems in point location is the Post Office problem, which asks, given a set of post offices in a town, which post office is the closest to some given point. It has been shown that the answer for finding the closest point is computing the Voronoi diagram, which partitions the plane is areas of closest points.

In this project, we focused, firstly, on constructing the Voronoi diagram for a given set of points, then extending it to a power diagram and optimizing the transport using LBFGS. In the last part of the project, we simulated the semi-discrete optimal transport fluid simulator with free surfaces. Unfortunately, our simulator didn't work properly and we were not able to include a video for the movement of the particles.

For our task, we developed multiple classes, but the main two ones are `Polygon`, `Voronoi` and `VoronoiFluid`. The first one stores a polygon, given by its vertices in counterclockwise order and has methods defined for computing its area, its centroid and its edges. The second one stores a set of points in $[0, 1]^2$ and (their weights) and computes the Voronoi cells (power cells) for each point. The last class mimics the `Voronoi` class but it also adds the feature of clipping the cells of fluids with disks.

2 Voronoi cells

In lab 6, we dealt mostly with the implementation of the Sutherland-Hodgman clipping algorithm and its extension towards the Voronoï Parallel Linear Enumeration algorithm in 2D, for which we followed the directions from [1]. The first brick of our project consisted in the coding of clipping algorithm, which, however, was not used in this part. So, our initial focus moved towards the clipping by a bisector, which given the Voronoi cell at a given time of some point and another point from the set, it eliminates the space that is closer to the second point. Using this algorithm, we computed the Voronoi cells of a given set of points.

For testing our code we generated random points in the unit square and computed the Voronoi cells. The results can be observed in Figure 1 and Figure 2.

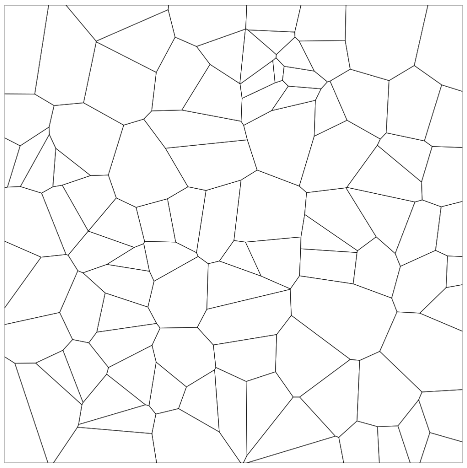

Figure 1: The resulting diagram for 100 points.

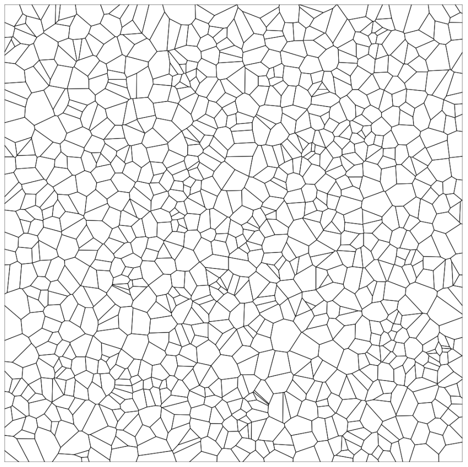

Figure 2: The resulting diagram for 1000 points. The time elapsed (not in parallel) was 2.57 seconds

3 Power diagrams and L-BFGS

In lab 7, we extended the results obtained for Voronoi cells by designing a power diagram. In this regard, we initially added weights to our points in the unit square and we changed the decision algorithm for our closest point. Instead of assigning a point in the square to the cell corresponding to its closest neighbour as before, we created small circles around the points and computed the closest tangent.

For testing our code, we generated random weights for our random points in the plane and we created the power diagrams. It can be observed in Figure 3 and Figure 4 that some of the points have no cell, as higher weights increase the size of the cells.

One of the most difficult parts of the project consisted in finding the optimal transport weights, using a quasi-Newton method called L-BFGS. Our implementation consisted on creating our own `evaluate` function for computing the objective function and its gradient (both negative as we want to maximize) and we relied on the L-BFGS library for the rest. Our implementation relied on the sample code provided in the repo of the library and we called the optimizer directly from `main`. After many hours of debugging, the evaluate function managed to properly handle the gradients and the objective function, giving the weights for obtaining cells of same area for each point.

For testing the implementation, we generated random points as before and we even assigned initial random weights and still obtaining convergence. The results can be observed in Figure 5 and 6

4 Fluid simulation

The last part of the project consisted in simulating the movement of some fluid particles in the unit square under the influence of a spring force and of gravity. Even though our simulation did not succeed, we still added our implementation in the repo. Our approach consisted in implementing a new class which handles the fluid particles, namely `VoronoiFluid`. This enabled us to compute the power diagrams for the fluid particles while taking into consideration the infinite air particles around them by clipping the cells with

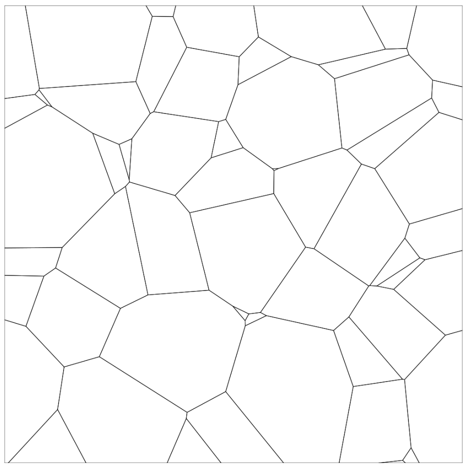

Figure 3: The resulting diagram for 100 points.

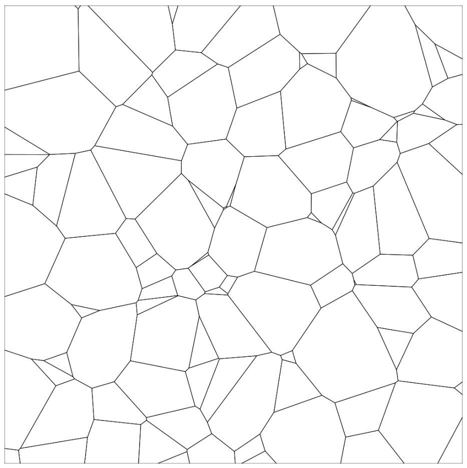

Figure 4: The resulting diagram for 1000 points. The time elapsed (not in parallel) was 1.50 seconds

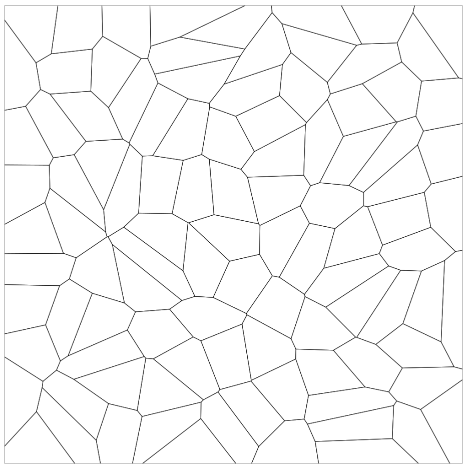

Figure 5: The resulting diagram for 100 points. The convergence took around 120 iterations for random weights.

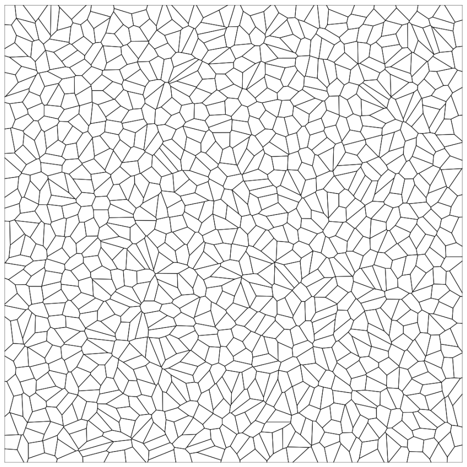

Figure 6: The resulting diagram for 1000 points. The convergence took around 320 iterations, each iteration lasting around 2 seconds(not in parallel)

disks. Here, we made use of the Sutherland-Hodgman that we implemented in Lab 6. We simulated a disk by a regular polygon with 30 vertices. The main issue that we encountered was in the new evaluate function, which had to deal with the new objective function. Even though we had a similar to approach to our previous L-BFGS, the main difference being that we added an extra weight, our optimizer did not converge. We also implemented the Gallouet Merigot scheme but since our optimizer was not converging, we didn't get any results. In order to debug the implementation, I discussed with my colleague Cezara Petrui, who was having similar issues but we couldn't fix it.

5 Course Feedback

I really enjoyed the course and especially implementing the algorithms. I loved the first project and I am happy that I managed to obtain my own raytracer. In the second one, I struggled with the optimization part but CSE306 ended up being on eof my favourites course in the bachelor program. Thank you very much for the effort you put on this course and I am glad I had the chance to discover the field of Computer Graphics.

6 Conclusions

In this report, we implemented a class for computing the Voronoi cell for some given points and we improved it for computing the power diagram and the L-BFGS optimization. We also attempted to simulate the movement of fluid particles. The code can be found at https://github.com/alexserban2002/CSE306.git.

For elaborating the project, I discussed some aspects of the implementation with my colleague Cezara Petrui, especially for L-BFGS and the fluid particles part.

References

[1] NICOLAS BONNEEL. Computer graphics - cse306 lecture notes. 2024.

[2] J Matusek. Algorithms, probability and computing lecture notes eth zurich. 2023.