École Polytechnique

BACHELOR OF MATHEMATICS AND COMPUTER SCIENCE

Computer Graphics: Project 2

Anatole Jacquin de Margerie, École Polytechnique

Academic year 2024/2025

Project 2 was quite fun, and I enjoyed seeing my work rewarded with the expected results, and cool shapes! Although, Project 1 did result in more beautiful images... Regardless, here is my project report, and I hope it is satisfactory!

For Lab 6, we were asked to implement the Voronoï Parallel Linear Enumeration algorithm in 2D, at least the naïve $O(N^2)$ version. For that, I implemented the Sutherland-Hodgman polygon clipping algorithm, following closely the algorithm described in the Lecture Notes (function $clip\_polygon$).

Then, we can construct our Voronoi cells simply by clipping a bounding box with the half-spaces defined by the bissectors between every 2 sites. This leads to two functions. The main function 'voronoi' loops over each site, and clipping the bounding box iteratively with each half-plane, which is constructed by the function 'createhalfplane'). We then obtain the following result, using the sites: Vector(0.25, 0.25), Vector(0.75, 0.25), Vector(0.5, 0.75), Vector(0.25, 0.75), Vector(0.75, 0.75), Vector(0.5, 0.5), Vector(0.25, 0.5), Vector(0.75, 0.5), Vector(0.1, 0.1) :

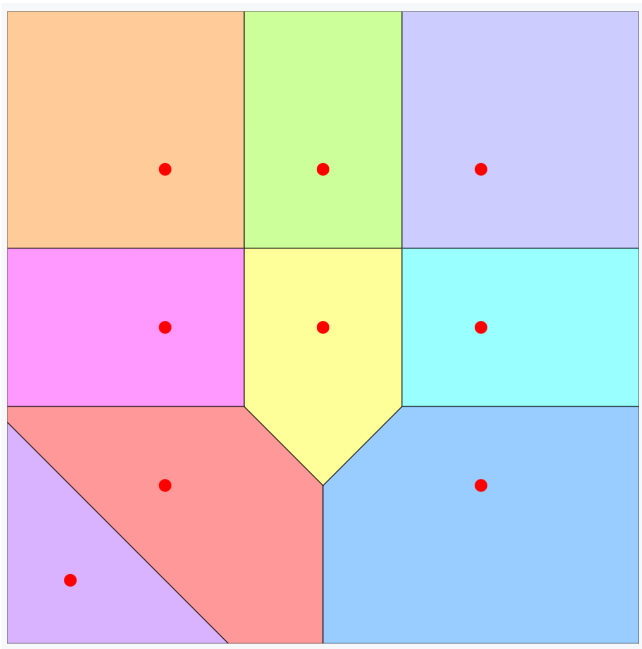

Figure 1: Voronoi diagram of a set of sites in red

We will use the same configuration for the following implementations.

The objective of lab 7 was to implement semi-discrete optimal transport in 2d using L-BFGS between a set of weights Diracs and a uniform density f. Working from our previous code from lab 6, we thus have to add the power diagram functionality, which can be done simply by adapting the midpoint M which defined our clipping half-plane, with a set of given weights. Using $r_i = \sqrt{w_i} * 1000$, (the svg render is scaled by 1000), we can obtain the following Power Voronoi diagram with arbitrary weights:

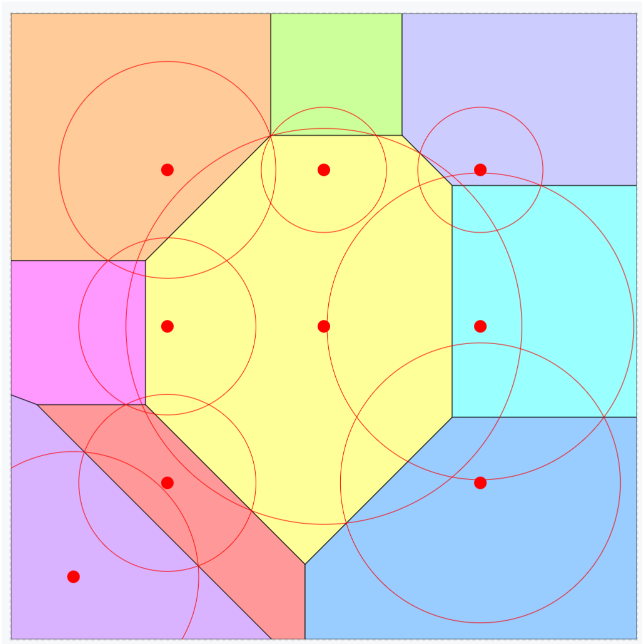

Figure 2: Power Voronoi diagram with weight circles

Then, we had to make use of the libLBFGS library and follow their sample code to implement the function evaluate(...), which will construct a power diagram whose weights are the variables passed in parameter to the "evaluate" function, return minus the gradient ∇ g of the objective function as well as the function -g itself.

The objective, using L-BFGS optimization, is to obtain (in the case of uniform density f), a set of weights for the power voronoi diagram, such that each cell has equal area. Which we obtain below:

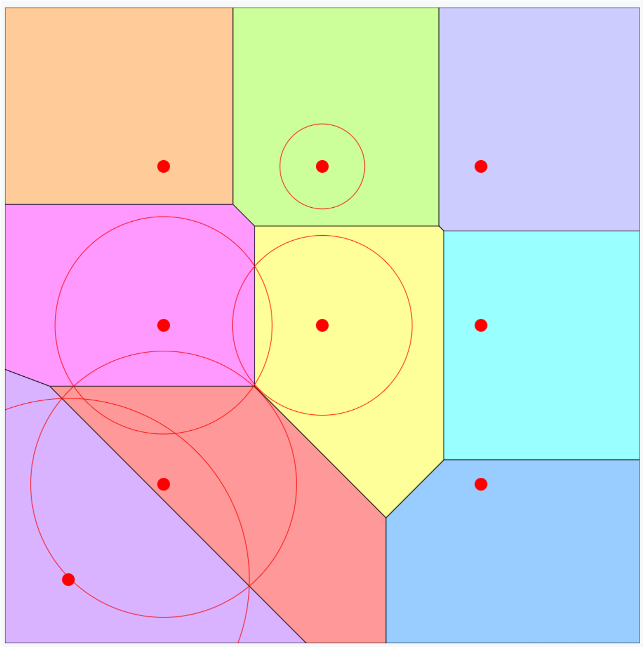

Figure 3: Semi-discrete Optimal Transport results (only positive weight circles displayed)

I really struggled to understand how we were supposed to use the libLBFGS library, and what exactly evaluate had to do, so ChatGPT did help steer me in the right direction of what to do, explaining to me the big idea of what evaluate had to do (and the parameter declaration list for evaluate), and defining the SDOTContext struct for me. The functions to compute the cell area and the integral (for which we were given a closed formula in the lecture notes) were done fully by me. However, because I kept getting the error code -998 from the L-BFGS optimization, and bad convergence, ChatGPT helped me by adding a function to ensure all polygons are counter-clockwise. I do not know if this was necessary, but after a lot of fumbling around, it eventually worked perfectly, giving the result above, where each cell has area $\frac{1}{9} = 1.1111..$ approximately.

I did not manage to implement fluid simulation, unfortunately.

Many thanks for your time, and for this fun course!