CSE306 - Assignment 2 Report

Duc Nguyen

1. Introduction

In this assignment, I have implemented the following features:

- Voronoi diagram solver, using Sutherland-Hodgman algorithm;
- A kd-tree optimization from scratch,
- Semi-discrete Optimal Transport solver with LBFGS

2. Code Organization

The code base is arranged as follow:

"include" folder contains important header files:

- **class_vector.h**: Declaration of class vector and related operations
- **class_polygon.h**: Declaration of class polygon and its method (clip with an infinite line supported by 2 points, clip by disc or polygon)
- **utils.h**: Contains function to write images
- **voronoi.h**: Contains function to compute the voronoi diagram
- **optimal_transport.h**: Declaration of Optimal Transport solver

"src" folder contains definition of function and operation in header files

"optim" folder contains implementation of kd-tree optimization

"lib" folder contains external header files (stb_image_write.h, lbfgs.h, …)

You can choose to turn on or off the kd-tree optimization by comment the KDTREE_OPTIM flag defined inside optim/optim.h file.

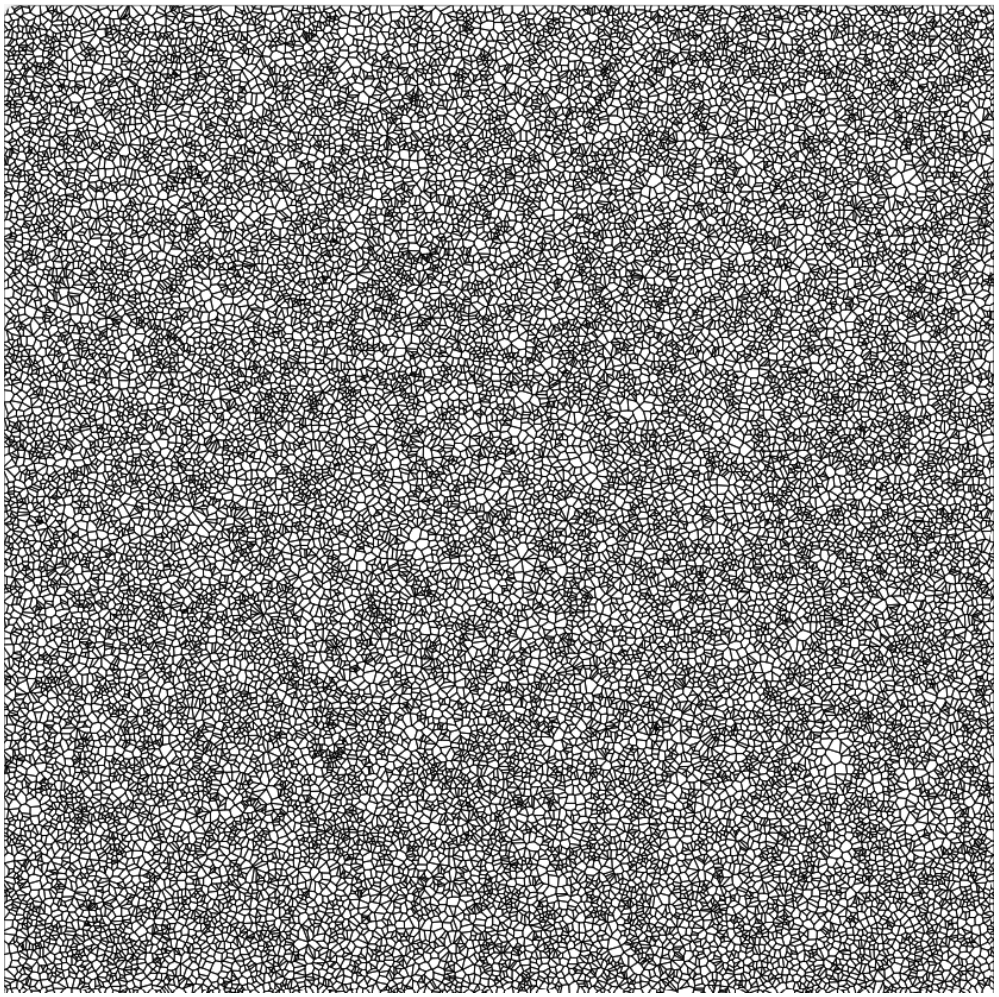
3. Technical Condition

The rendering process is conducted on the my personal laptop:

Chip: 11th Gen Intel(R) Core(TM) i5-11400H @ 2.70GHz 2.69 GHz
RAM: 8GB
OS: Windows 11

4. Result Images and Performance

- Rendering result of basic Voronoi Diagram:
    - 3000 initial points
    - Naive algorithms: 1.048 s (with parallel)
    - kd-tree optimization: 0.333 s (with parallel)



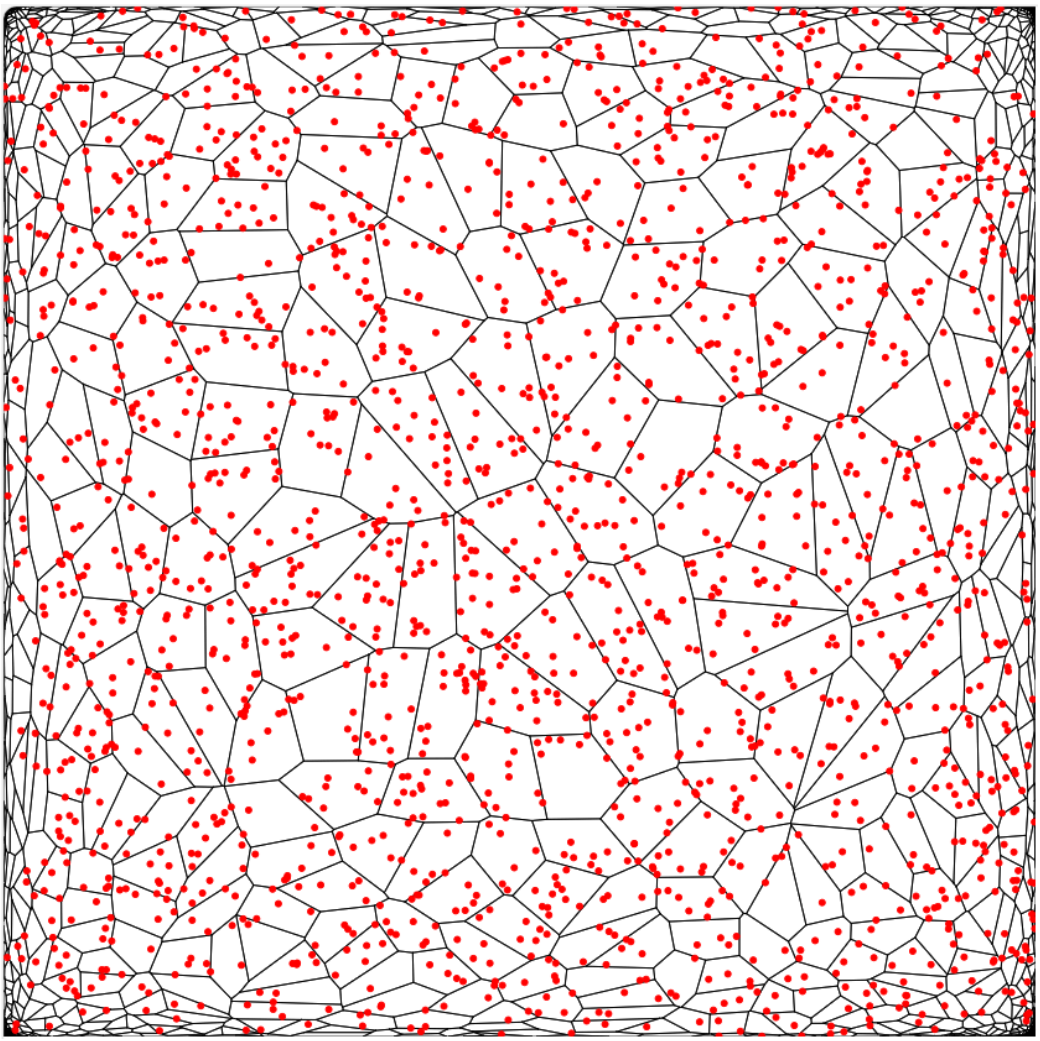
- Rendering result of power diagram with optimized weights:
    - 2000 points
    - Naive algorithm: 1m46.188s
    - Kd-tree optimization: 3m6.174s

I measured everything after finishing the fluid simulation, but I just did not find out which I changed so the Kd-tree optimization seems not doing its role this time.

I support at the corner of the square, number of points that should be query to create the polygon increase drastically, so the the complexity for those points will be:

$$K \log(n) + 2K \log(n) + \dots$$

which at the end would yield a complexity that is worse than the naive algorithm.
So, depending on the optimizing target, the kd-tree is not always the best choice



- Rendering result for fluid simulation:
    - 500 fluid particles
    - 400 frames
    - implement the scenario where number of air particles tend to infinity
    - merge to 24 fps video, using ffmpeg
- KD-tree optimization: 8m28.431s
- Naive iteration: 7m48.589s

  The demo video can be found under /images/ folder: fluid.mp4