# Project report

## 1 Clear description of generally what you want to do and why. (fsvd, etc.)

## 2 Clear description of the method proposed and why. (envelopes, etc.)

For the 2 previous, introductory concepts, feel free to get inspired by existing papers/guides/etc, just mention them.

## 3 A timeline of all steps and approaches done.

### 3.1 You should mention all issues faced one by one.

### 3.2 How you attempted to solve them initially and why it didn't work out.

### 3.3 How you solved them, finally.

### 3.4 For all issues not solved, mention them clearly.

### 3.5 Why you didn't manage to solve them, what were the main difficulties.

### 3.6 How would you have solved them if you had more item (drop ideas).

### 3.7 Special cases you didn't have time to tackle and ideas for dealing with them.

### 3.8 What would further work on this topic include.

In general keep in mind that someone would ideally read your report and clearly understand what your problem was, how you dealt with it, which difficulties you faced, how did you tackle them (or not) and how your work could be extended.

# 4 Overview

The goal of this project was to develop an extension for the drawing application Ipe that allows to draw the Farthest Line Segment Voronoi Diagram (FSVD); such an extension is called an Ipelet.
The Computational Geometry Algorithms Library (CGAL) for `C++` offers a good support to develop Ipelets.
This project's repository is available on Github at https://github.com/Spyridox/UROP2017-FSVD.

## 4.1 FSVD

The Farthest Line Segment Voronoi Diagram is a planar arrangement constructed for a set of line segments. It has faces, edges and vertices and it represents the area of points that are farthest from a given segment, for all segments (for some, this area might be empty); for some segments, this area could also be disjoint, for example constituted by two separate parts. The edges of a FSVD always form a tree, and are therefore never disconnected (unlike the Nearest Line Segment Voronoi Diagram, where the edges can be in up to $n-1$ disconnected parts (for a diagram for $n$ segments). See examples below (figure 1).

(a) Nearest (b) Farthest

Figure 1: Voronoi diagrams with line segments

## 4.2 CGAL and Ipelets

In this project two new functions were added to an Ipelet that already featured functions to construct Voronoi diagrams of points, Voronoi diagrams for segments, points and polygons using the $L_\infty$ metric, farthest color and Hausdorff Voronoi diagrams.
The two functions added are to construct, under $L_2$ metric, the nearest or the farthest Voronoi diagram for line segments.
For both cases the strategy used is to get the distance function of each segment as a surface, then take either the upper or the lower envelope of these surfaces (for FSVD and SVD respectively).

## 4.3 The envelopes strategy

For a single segment on the $xy$-plane, for every point plot on the $z$ axis the distance of that point to the segment. This creates a 3D surface. For a point, this surface is simply a cone; for a segment, it's

two haflcones originating at the endpoints of the segment and two planes originating from the inner part of the segment (figure 2).

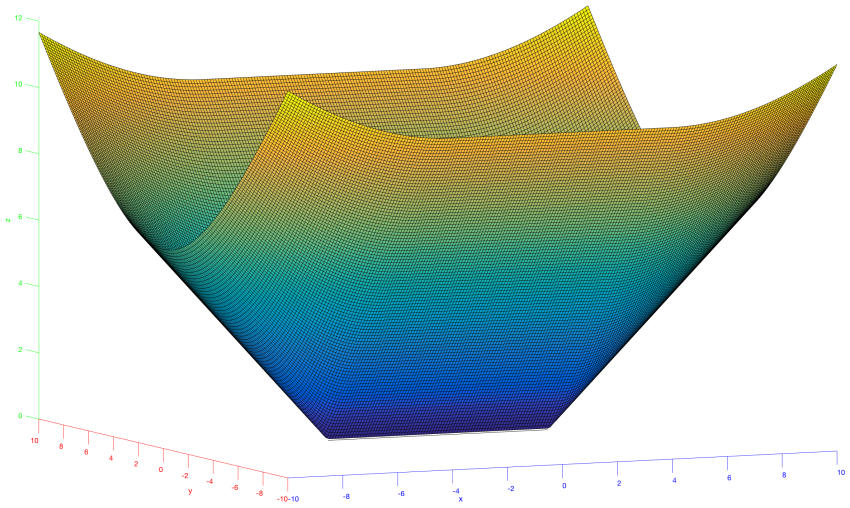

Figure 2: Distance function surface for a segment[1]

For two distinct segments, these distance function surfaces intersect, and their intersection projected back onto the $xy$-plane is simply the planar euclidean bisector of the two segments.
For many segments, given all their distance function surfaces, if the lower envelope is taken then the $xy$-projection of this envelope is the nearest Voronoi diagram, whereas the $xy$-projection of the upper envelope is the farthest Voronoi diagram.

# 5 Envelope package and `Envelope_diagram_2<EnvTraits>`

CGAL features a package to compute projections of envelopes. The methods `upper_envelope_3` and `lower_envelope_3` output their resulting diagram in a `Envelope_diagram_2` object. This diagram class is parametrised by a traits class (it has to be a model of the concept `EnvelopeTraits_3`) that needs to support computation of the projected boundary of a surface, the projection of the intersection of two surfaces, as well as other functions to determine the z-order of two surfaces in specific cases. There are already traits classes to support construction of envelopes of spheres, triangles and planes.

## 5.1 Traits class

The `EnvelopeTraits_3` concept allows the surfaces to be any kind of object, as long as the required functions do their task. Because of this, for FSVD (and SVD too, they can use the same traits class) the surfaces are simply the segments themselves, since they contain all the information needed for the computations. In the function that requires to compute the projected intersection of two surfaces, we can in fact simply compute the planar euclidean bisector of the two segments. The bisector is formed, in the general case, by unbounded rays, segments and parabolic arcs (figure 5).

[1] created using MATLAB script by Martin Suderland

### 5.2 Arrangement traits class

`Arr_conic_traits2`

### 5.3 Parabola class

A parabola class was implemented to support construction of parabolas, computation of intersection with lines and construction of arcs on the parabolas. Other predicates such as `has_on` were also implemented.

## 6 Bisector computation

### 6.1 Unbounded rays

### 6.2 Internal part

### 6.3 Challenges and solutions

## 7 Results and future work

### 7.1 Ipelet issues

### 7.2 Known limitations

### 7.3 Alternatives

#### 7.3.1 `Arr_linear_segment_traits_2`

#### 7.3.2 Own Arrangements class