2D Centroidal Voronoi Tessellations with Constraints

Jane Tournois, Pierre Alliez, Olivier Devillers

2D Centroidal Voronoi Tessellations with Constraints

Jane Tournois, Pierre Alliez and Olivier Devillers

Abstract

We tackle the problem of constructing 2D centroidal Voronoi tessellations with constraints through an efficient and robust construction of bounded Voronoi diagrams, the pseudo-dual of the constrained Delaunay triangulation. We exploit the fact that the cells of the bounded Voronoi diagram can be obtained by clipping the ordinary ones against the constrained Delaunay edges. The clipping itself is efficiently computed by identifying for each constrained edge the (connected) set of triangles whose dual Voronoi vertices are hidden by the constraint. The resulting construction is amenable to Lloyd relaxation so as to obtain a centroidal tessellation with constraints.

1 Introduction

Voronoi diagrams have been extensively studied in the field of computational geometry [Aur91]. Given a set of points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, called sites or generators, the Voronoi diagram is defined as the space decomposition into cells according to the nearest site. Namely, the Voronoi cell associated to $\mathbf{x}_i$, denoted $\mathcal{V}_i$, is defined as

$$\mathcal{V}_i = \left\{ x \in \mathbb{R}^2 \mid d(x, \mathbf{x}_i) \leq d(x, \mathbf{x}_j), \forall j \in \{1..N\}, j \neq i \right\}.$$

One popular way to efficiently construct Voronoi diagrams consists in exploiting its duality property with the Delaunay triangulation: The Delaunay triangulation can be defined as the dual of the Voronoi diagram, as the triangulation obtained by creating a Delaunay edge $x_i x_j$ if the Voronoi cells $\mathcal{V}_i$ and $\mathcal{V}_j$ are neighbors (they share a Voronoi edge). A direct characterization of the Delaunay triangulation is also possible: a triangle defined by three points of $\mathcal{X}$ belongs to the Delaunay triangulation if none of any other point of $\mathcal{X}$ is located inside its circumcircle.

Centroidal Voronoi diagrams are commonly used in some applications which require a good sampling of an input domain. One way to distribute a set of points isotropically and in accordance with a density function is to apply the Lloyd iteration (described in Section 3) over an initial Voronoi diagram of the input points. Du et al. [DFG99] have shown how the Lloyd iteration transforms an initial ordinary Voronoi diagram into a centroidal Voronoi diagram, where each generator happens to coincide with the centroid of its Voronoi cell. This process is a way to trade global requirements (the density function) for local requirements of generating a locally uniform distribution of the Voronoi sites.

The Lloyd iteration assumes a Voronoi tessellation with bounded cells so that the centroid of each Voronoi cell is well defined. Assuming a bounded input domain Ω, one direct way to proceed consists of intersecting each Voronoi cell with Ω and computing the centroid of the resulting intersection (more specifically the connected component containing the cell generator). In addition to suffering from the usual robustness issues, the intersections may result in non-convex or non-simply connected cells and hence in centroids located outside Ω (see Figure 2-Left & Middle).

For cases where the input domain boundary is a polygonal line, one solution consists of relying on a *constrained Delaunay triangulation* (CDT). It is a generalization of the Delaunay triangulation which allows the addition of constrained line segments appearing as edges of the triangulation [Che87]. The end points of those line segments are also the generators of the dual Voronoi tessellations. In a CDT, a triangle is valid if its circumcircle does not contain any point of $\mathscr{X}$ visible from inside the triangle. To define the visibility notion, the input domain boundary is considered as a set of occluding barriers (see Figure 1-Left).

One of our goals is to consider these geometric constraints in a generic manner so as to handle inner isolated constraints, evolving cracks, etc. Among others, natural elements and natural neighbor methods [SMB98, YRLC04, MCA+04, YCLR05] need to handle this type of constraints. In these applications the constraints can move in an unpredictable manner. Robustness is thus a key point of our work, since at every step convex Voronoi cells are required.

We now have to deal with *Voronoi cells with constraints*. As explained above, Delaunay triangulation and Voronoi diagram are dual structures. For our purpose defining a dual of the CDT would thus be useful. It is possible to construct the usual Voronoi diagram from the Delaunay triangulation with the following dual rule:

Voronoi vertices are constructed at circumcenters of Delaunay triangles and Voronoi edges are drawn between dual of neighboring Delaunay triangles.

The standard dual of the CDT is the *constrained Voronoi diagram* (CVD), defined by a slight modification of this rule:

constrained Voronoi vertices are constructed at circumcenter of constrained Delaunay triangles and constrained Voronoi edges are drawn between dual of Delaunay triangles which are neighbors through a non constrained Delaunay edge

(see Figure 1-Middle). Note that this definition allows the Voronoi diagram to cross the constraints, or more exactly, some part of the Voronoi diagram (dashed in Figure 1-Middle) continue on the wrong side of the constraints as if they were on the right side. Several approaches have been proposed [WS87, Lin89, Sei88]. We base our work on Seidel's [Sei88] definition of the *bounded Voronoi diagram* (BVD) which clips the CVD with constraints (see Figure 1-Right and 2-Right).

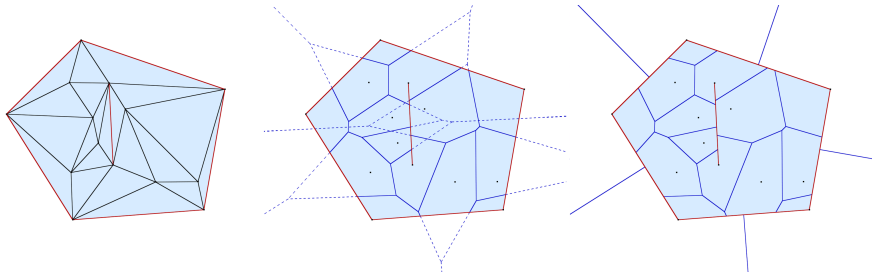

Figure 1: (Left) CDT: Constrained Delaunay triangulation of a set of vertices. (Middle) Constrained Voronoi diagram: CVD, standard dual of the CDT. (Right) BVD: Bounded Voronoi Diagram.

Contribution

We propose a simple and efficient algorithm to construct the BVD from the CDT, as follows. First, mark the triangles whose dual BVD vertices belong to the CVD but not to the BVD. Then, extract the BVD cells from the CVD. Notice that a Voronoi edge of the CVD may be clipped by constraints which are not close to its dual Delaunay edge (see Figure 3). This BVD construction makes possible to take the constraints into account in an efficient manner during the Lloyd iterations, as explained in Section 3.

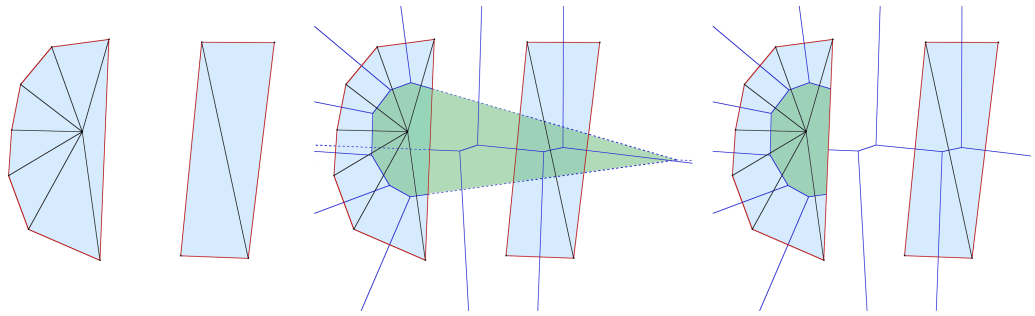

Figure 2: (Left) CDT: Constrained Delaunay triangulation of a set of vertices. (Middle) A CVD cell (green), from standard dual of the CDT. Its intersection with the domain has two connected components. (Right) The suitable BVD clipped cell.

2 Bounded Voronoi Diagram

While the ordinary Voronoi diagram does not take constraints into account, we wish here to prevent the Voronoi regions to cross over the constraints. To be able to run Lloyd iterations, each Voronoi cell must be convex and simply connected. To this aim we use a *bounded Voronoi diagram*, defined by Seidel [Sei88] as a pseudo-dual to the *constrained Delaunay triangulation* [Che87, She96] (see Section 1).

The common duality between Delaunay triangulation and Voronoi diagram links each triangle to its circumcenter. In our context, each triangle $\triangle$ may have its circumcenter c on the other side of a constrained edge. Hence, c is the dual of $\triangle$ if it is on the same side of the constraint. Otherwise, it is a pseudo-dual, and some Voronoi edges of $\triangle$ must be clipped by the constraint. The BVD is defined as follows: each cell $\mathcal{V}_i$ of a generator $\mathbf{x}_i$ is composed by the points of the domain Ω which are closer to $\mathbf{x}_i$ than to any other generator. As for the constrained Delaunay triangulation, the distance incorporates visibility constraints. The distance $d_S(x, y)$ between two points x and y of $\mathbb{R}^2$ is defined as:

$$d_S(x,y) = \begin{cases} ||x-y||_{\mathbb{R}^2} & \text{if } x \text{ “sees” } y, \\ +\infty & \text{otherwise.} \end{cases}$$

In this definition, x “sees” y when no constrained edge intersects the segment $[x, y]$. This visibility notion can be extended to triangles. We will see later how the notion of triangle sight, or symmetrically triangle “blindness”, is important to construct the bounded Voronoi diagram. Figure 4 illustrates a constrained Delaunay triangulation and its pseudo-dual bounded Voronoi diagram. Notice that trying to construct the naïve Voronoi diagram by joining

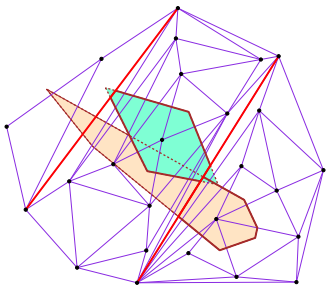

Figure 3: Two Voronoi cells and their clipped bounded Voronoi counter parts. These cells are generated by a site which is far from the constraints and must be clipped.

the circumcenters of all pairs of incident triangles would not even form a partition. The notion of triangle blindness is pivotal for constructing the bounded Voronoi diagram.

Definition 2.1 (Blind triangle)**.** *A triangle $\triangle$ is said to be* blind *if the triangle and its circumcenter c lie on the two different sides of a constrained edge E. Formally, $\triangle$ is* blind *if and only if there exists a constrained edge E such that one can find a point p in $\triangle$ (not an endpoint of E), such that the intersection $[p, c] \cap E$ is non-empty.*

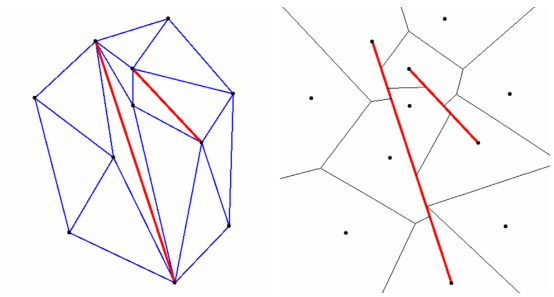

Figure 4: Constrained Delaunay triangulation of a set of points (left) and its pseudo-dual bounded Voronoi diagram (right).

The BVD construction algorithm initially tags all triangles of the triangulation as being blind or not blind (Algorithm 1). It then constructs each cell of the diagram independently using these tags (Algorithm 2). Finally, all cells are assembled to build the complete bounded Voronoi diagram of a given set

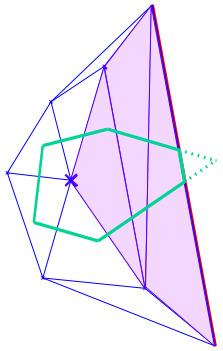

Figure 5: Construction of a cell of the bounded Voronoi diagram. The standard Voronoi diagram is truncated on the constrained edge (colored triangles are blind).

of points and constrained edges.

Algorithm 1 tags all triangles of the triangulation as being either *blind* or *non-blind*. In addition, each *blind* triangle stores which constrained edge in the triangulation acts as a visibility obstacle, *i.e.*, which edge prevents it to *see* its circumcenter (it is the first constraint intersected by the oriented line joining any point of the triangle to its circumcenter). Notice how the algorithm only needs to iterate over the constrained edges of the triangulation, as all sets of blinded triangles form connected components incident to constrained edges. Indeed, the Voronoi diagram of the vertices of the blind triangles on one side of any constraint is a tree rooted from the dual ray of the constraint.

We define a robust predicate, called $\mathscr{P}$ in the sequel, to test if a triangle is blinded by a constrained edge. More specifically, $\mathscr{P}$ takes as input a triangle and a segment, and returns a Boolean indicating whether or not the circumcenter of the triangle lies on the same side of the segment than the triangle. The circumcenter is never constructed explicitly in order to obtain a robust tagging of the blind triangles. Each cell of the bounded Voronoi diagram can be constructed by circulating around vertices of the triangulation, and by choosing as cell vertex either circumcenters or intersections of the standard Voronoi edges with the constrained edges. Note that we do not need to construct bounded Voronoi cells incident to input constrained vertices as the latter are constrained and therefore not relocated by the Lloyd iteration. Algorithm 2 describes this construction, and Figure 5 illustrates the construction of a single bounded Voronoi cell. Figure 6 illustrates a bounded Voronoi diagram.

Algorithm 1 Tag blind triangles

Input: Constrained Delaunay triangulation cdt.

Tag all triangles non-blind by default.
for each constrained edge e of cdt **do**
  Create a stack: $triangles$
  for both adjacent triangles f_e to e tagged non-blind **do**
    Push f_e into $triangles$
    while $triangles$ is non-empty **do**
      Pop f from stack $triangles$
      if f is blinded by e (use $\mathscr{P}$) **then**
        Tag f as blinded by e
        for each adjacent triangle f' to f **do**
          if f' is finite and tagged non-blind
          & the common edge between f and f' is unconstrained **then**
            Push f' into $triangles$.

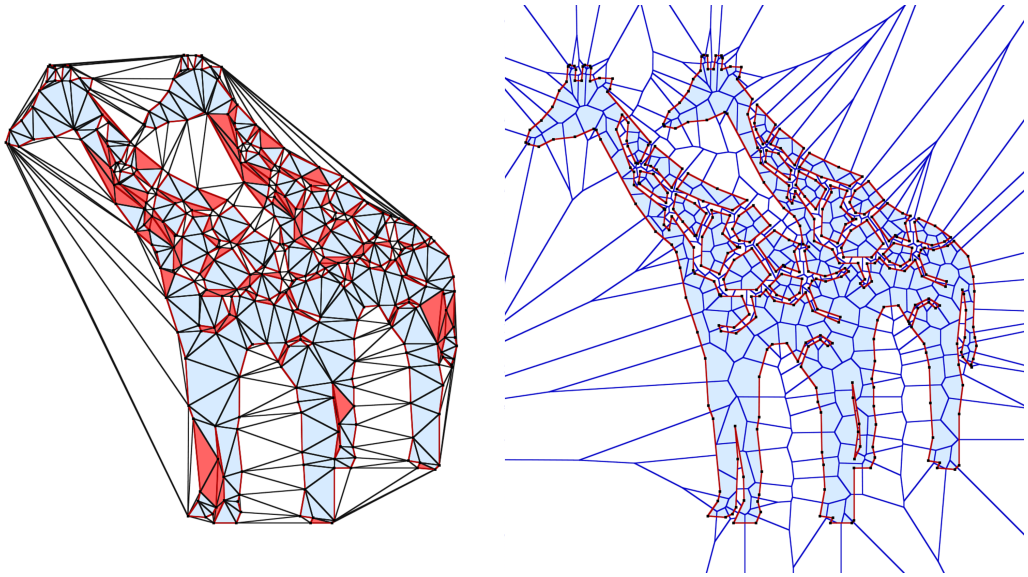

Figure 6: CDT with its blind triangles, and bounded Voronoi diagram of a PSLG.

Algorithm 2 Construct a BVD cell

Input: Unconstrained vertex $\mathbf{x}$ of the constrained Delaunay triangulation cdt.

Call P the polygon (cell) in construction,
Call f_{init} a triangle incident to $\mathbf{x}$,
Let f be initialized to f_{init}
Call f_{next} the next triangle counterclockwise around z,
Call $\mathscr{L}_{f,f_{next}}$ the line going through the circumcenters of f and f_{next}.
repeat
  if f is tagged non-blind **then**
    Insert the circumcenter of f into P.
    if f_{next} is blind **then**
      Call $S_{f_{next}}$ the constrained edge blinding f_{next},
      Insert point $\mathscr{L}_{f,f_{next}} \cap S_{f_{next}}$ into P.
  else
    Call S_f the constrained edge blinding f.
    if f_{next} is tagged non-blind **then**
      Insert $\mathscr{L}_{f,f_{next}} \cap S_f$ into P.
    else
      Call $S_{f_{next}}$ the constrained edge blinding f_{next},
      if $S_f \neq S_{f_{next}}$ **then**
        Insert $\mathscr{L}_{f,f_{next}} \cap S_f$ and $\mathscr{L}_{f,f_{next}} \cap S_{f_{next}}$ into P.
  $f \leftarrow f_{next}$
  Call f_{next} the next triangle counterclockwise around z,
until $f = f_{init}$

Output: Bounded Voronoi cell of $\mathbf{x}$ in counterclockwise order.

3 Lloyd iteration

Energy minimization The Lloyd iteration [Llo82] is a minimizer for the energy functional:

$$\mathscr{E} = \sum_{i=1}^{N} \int_{y \in \mathscr{V}_i} \rho(y) ||y - \mathbf{x}_i||^2 dy,$$

where ρ is a density function defined over the domain Ω, $\{\mathbf{x}_i\}_{i=1}^N$ the generators and $\mathscr{V}_{i\,i=1}^N$ the corresponding Voronoi cells.

It minimizes this energy by alternately moving the generators to the centroid of their Voronoi cells, and recomputing the Voronoi diagram. The cen-

troid $\mathbf{x}^\star$ of the cell $\mathcal{V}$ is defined as:

$$\mathbf{x}^\star = \frac{\int_{\mathcal{V}} y\rho(y)dy}{\int_{\mathcal{V}} \rho(y)dy}.$$

After convergence, the space subdivision obtained is a *centroidal Voronoi Tessellation* (CVT) [DFG99, DEJ06]. As it corresponds to a critical point of the energy $\mathcal{E}$, it is a necessary condition for minimizing $\mathcal{E}$. Figure 7 is an illustration of the evolution during Lloyd iterations.

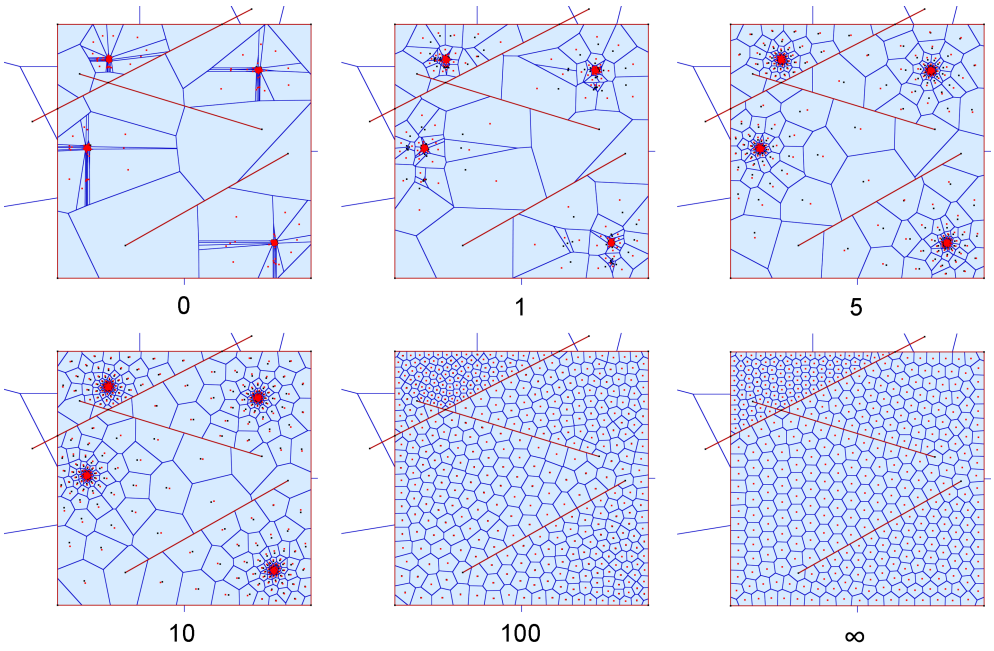

Figure 7: Lloyd iteration with a uniform density. Four clusters of vertices (0), randomly located inside a square with constraints. Black points are the generators, red points are the cells' centroids. In the reading order, generators and centroids after 1, 5, 10, 100 of iterations; and after convergence of the Lloyd iteration, when generators and centroids coincide.

Convergence criterion We choose to stop the Lloyd iteration when all generators move less than a user-defined distance threshold. We first define the notion of move ratio of a generator $\mathbf{x}$ in its Voronoi cell $\mathcal{V}$ as $mr(\mathbf{x}) = \frac{||\mathbf{x}-\mathbf{x}^\star||}{diameter(\mathcal{V})}$. The Lloyd iteration is stopped when $\max_{\mathbf{x}\in\{\mathbf{x}_i\}_{i=1}^N} mr(\mathbf{x}) < p$ (p is typically set to 1%).

The Lloyd iteration requires computing centroids of (possibly bounded) Voronoi cells in 2D. Such computations require a quadrature formula when a variable density function is specified, *i.e.*, when the input sizing function is not uniform. The density function ρ and the sizing function μ are linked by the following formula [DW03, DW06]: $\mu(x) = \frac{1}{\rho(x)^{d+2}}$, where d is the dimension of the domain. In 2D, this yields

$$\mu(x) = \frac{1}{\rho(x)^4}.$$

The key idea behind a quadrature is to decompose a simple domain (a convex polygon in our case) into smaller sub-domains (so-called quadrature primitives) where simple interpolation schemes are devised. The number n of quadrature primitives used for each element allows the user to tune the computation accuracy of the centroids. In practice this number is increased as the iterations go. During first iterations, vertices are moving a lot, inside their cells and inside the domain. The computation of the centroid location needs only a low precision, and $n = 10$ typically is enough. Later in the course of iterations, the Voronoi tessellation is getting close to be centroidal. To ensure the convergence of the tesselation to a CVT through Lloyd iterations, the precision needs to be increased. Otherwise, and in particular when the chosen sizing field is highly graded, it might happen that the Lloyd algorithm does not converge to a stable CVT. In practice, we run final steps with $n = 100$.

We use the midpoint approximation rule in 2D, with a decomposition of each bounded Voronoi cell into n sub-triangles. More precisely, an initial step triangulates the cell by joining each of its vertices to its generator. The next step recursively bisects the longest edge of these triangles until the number of quadrature triangles reaches n. On each quadrature triangle, the midpoint approximation formula is applied:

$$\int_{\triangle} f(x)dx \approx \frac{|\triangle|}{3}(f(x_{12}) + f(x_{23}) + f(x_{13})),$$

where x_{12}, x_{23} and x_{13} are the midpoints of a quadrature triangle edges. Finally, we sum the integrals on each quadrature triangle in order to obtain an approximate centroid of the whole bounded Voronoi cell. Our algorithm is implemented in C++ using the *Computational Geometry Algorithms Library* CGAL [CGA08].

4 Results

Algorithms 1 and 2 construct the bounded Voronoi diagram of a set of points, with respect to a set of line segment constraints. Running Lloyd algorithm on

the bounded Voronoi diagram turns it into a centroidal and bounded Voronoi tessellation. This optimization process alternates relocating each vertex to its cell centroid, and updating the tessellation. This is made possible thanks to the BVD properties: each bounded cell is convex and simply connected.

Figures 8, 10, and 9 show centroidal Voronoi tessellations generated from random initial point sets. Figure 8 highlights the fact that vertices lying in a bounded region are not allowed to cross the constraints.

Figures 10 and 9 show examples where the density function used in the Lloyd iteration can be either constant, or automatically adapted and k-Lipschitz. The automatically adapted density function that we use derives from the sizing function described in [ACSYD05] as:

$$\mu(x) = \inf_{s \in \partial\Omega} [kd(s,x) + size(s)],$$

where $\partial\Omega$ is the domain boundary, d the Euclidean distance, $size(s)$ the prescribed size at s, and k a user-defined constant. It is shown to be the maximum k-Lipschitz function that is smaller or equal to $size(s)$ on the boundary of the domain.

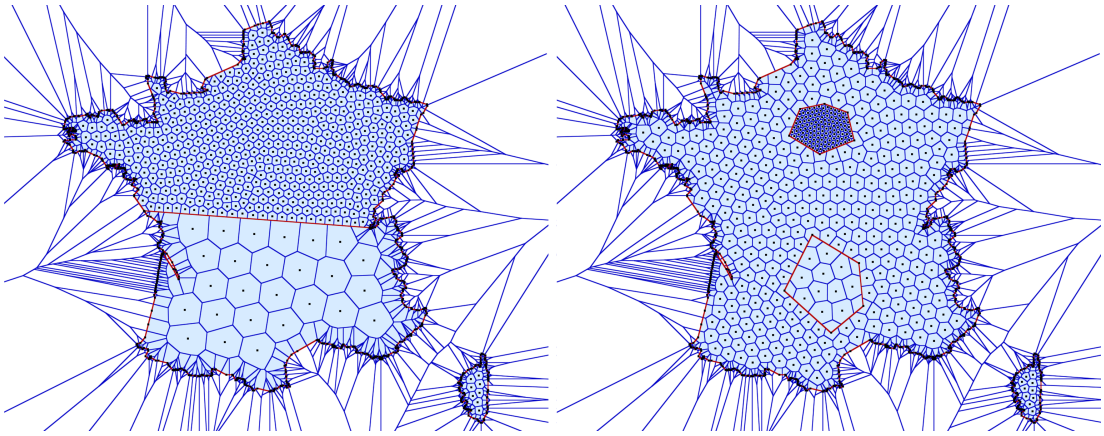

Figure 8: France. CVT inside the domain. Vertices lying in a bounded region are not allowed to cross the constraints.

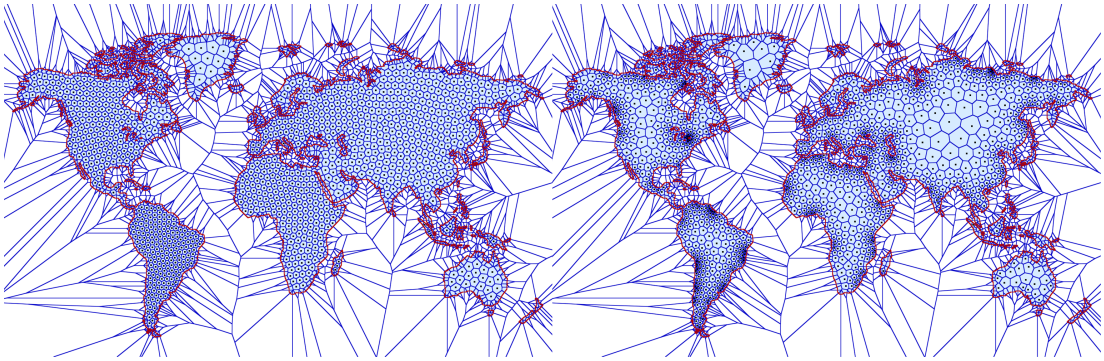

Figure 9: World. CVT inside the domain. The sizing function is chosen as being uniform (left) and adapted, with $k = 0.1$ (right).

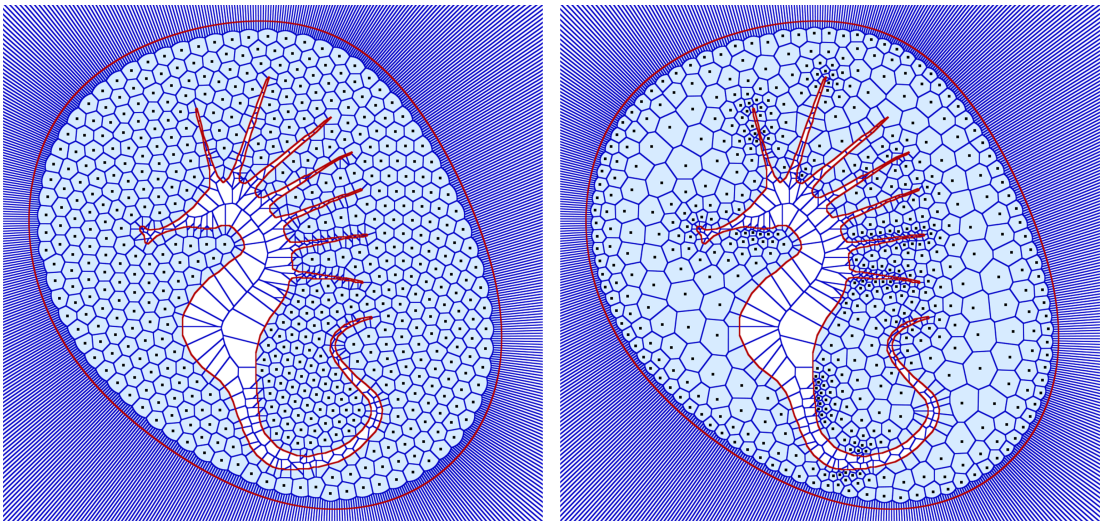

Figure 10: Sea horse. CVT inside the bubble domain. The sizing function is chosen as being uniform (left) and adapted, with $k = 0.1$ (right). Both CVT's contain 2000 points. Starting from a uniformly distributed initial set of points, Lloyd optimization reaches convergence in about 100 iterations. In the uniform case (left), it takes about 7 seconds, and in the adaptive case (right), it takes about 80 seconds.

5 Conclusion

We have proposed a robust and simple algorithm to compute 2D centroidal Voronoi tessellations with constraints. This algorithm is among others motivated by meshless simulation applications which require computing natural neighbor interpolation over bounded Voronoi diagrams. As future work we wish to elaborate upon a fully dynamic construction of the bounded Voronoi diagram such that removing a constrained edge leads to local updates of the blind triangles. Finally, we plan to elaborate upon the efficient computation of 2D natural neighbor interpolation coordinates based upon the proposed bounded Voronoi diagram.

References

[ACSYD05] P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 10, New York, NY, USA, 2005. ACM.

[Aur91] F. Aurenhammer. Voronoi diagrams. A survey of a fundamental geometric data structure. *ACM Computing surveys*, 23(3), 1991.

[CGA08] CGAL Editorial Board. *CGAL User and Reference Manual*, 3.4 edition, 2008. www.cgal.org.

[Che87] L.P. Chew. Constrained Delaunay triangulations. *Proceedings of the 3rd Annual ACM Symposium on Computational Geometry*, pages 215–222, 1987.

[DEJ06] Q. Du, M. Emelianenko, and L. Ju. Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations. *SIAM Journal on Numerical Analysis*, 44(1):102–119, 2006.

[DFG99] Q. Du, V. Faber, and M. Gunzburger. Centroidal Voronoi tessellations: applications and algorithms. *SIAM review*, 41(4):637–676, 1999.

[DW03] Qiang Du and Desheng Wang. Tetrahedral mesh generation and optimization based on centroidal voronoi tessellations. *International Journal for Numerical Methods in Engineering*, 56(9):1355–1373, 2003.

[DW06] Q. Du and D. Wang. Recent progress in robust and quality Delaunay mesh generation. *Journal of Computational and Applied Mathematics*, 195(1-2):8–23, 2006.

[Lin89] A. Lingas. Voronoi diagrams with barriers and the shortest diagonal problem. *Information Processing Letters*, 32(4):191–198, 1989.

[Llo82] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[MCA+04] MA Martinez, E. Cueto, I. Alfaro, M. Doblare, and F. Chinesta. Updated Lagrangian free surface flow simulations with natural neighbour Galerkin methods. *International Journal for Numerical Methods in Engineering*, 60(13), 2004.

[Sei88] R. Seidel. Constrained Delaunay triangulations and Voronoi diagrams with obstacles. *Report of the Institute for Information Processing, TU Graz*, 260:178–191, 1988.

[She96] J.R. Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. *Lecture notes in computer science*, 1148:203–222, 1996.

[SMB98] N. Sukumar, B. Moran, and T. Belytschko. The natural element method in solid mechanics. *International Journal for Numerical Methods in Engineering*, 43:839–887, 1998.

[WS87] C. Wang and L. Schubert. An optimal algorithm for constructing the Delaunay triangulation of a set of line segments. In *Proceedings of the 3rd Symposium on Computational Geometry*, pages 223–232. ACM New York, NY, USA, 1987.

[YCLR05] J. Yvonnet, F. Chinesta, P. Lorong, and D. Ryckelynck. The constrained natural element method (C-NEM) for treating thermal models involving moving interfaces. *International Journal of Thermal Sciences*, 44(6):559–569, 2005.

[YRLC04] J. Yvonnet, D. Ryckelynck, P. Lorong, and F. Chinesta. A new extension of the natural element method for non-convex and discontinuous problems: the constrained natural element method (C-NEM). *International Journal for Numerical Methods in Engineering*, 60(8), 2004.