Constructing L_∞ Voronoi Diagrams in 2D and 3D

D. R. Bukenberger   K. Buchin   M. Botsch

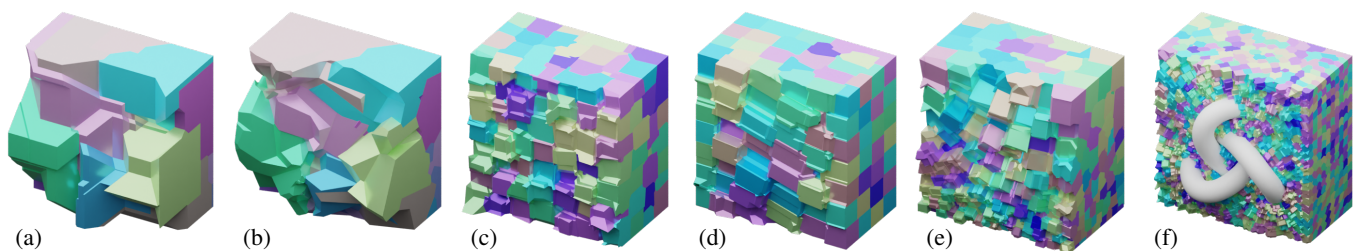TU Dortmund University

(a)   (b)   (c)   (d)   (e)   (f)

Figure 1: *Our algorithm constructs generalized 3D L_∞ Voronoi diagrams. With the non-euclidean L_∞ metric, diagram construction is no longer agnostic to orientations: This allows to incorporate individually oriented and anisotropically weighted sites. Results shown here are generated with: (a) random positions, axis-aligned orientations; (b) random positions, random orientations; (c) regular grid positions and orientations, both jittered; (d,e) CVTs on smooth 3D anisotropic weighting/orientation fields (f) induced by a trefoil knot input object (white).*

Abstract

Voronoi diagrams and their computation are well known in the Euclidean L_2 space. They are easy to sample and render in generalized L_p spaces but nontrivial to construct geometrically. Especially the limit of this norm with $p \to \infty$ lends itself to many quad- and hex-meshing related applications as the level-set in this space is a hypercube. Many application scenarios circumvent the actual computation of L_∞ diagrams altogether as known concepts for these diagrams are limited to 2D, uniformly weighted and axis-aligned sites. Our novel algorithm allows for the construction of generalized L_∞ Voronoi diagrams. Although parts of the developed concept theoretically extend to higher dimensions it is herein presented and evaluated for the 2D and 3D case. It further supports individually oriented sites and allows for generating weighted diagrams with anisotropic weight vectors for individual sites. The algorithm is designed around individual sites, and initializes their cells with a simple meshed representation of a site's level-set. Hyperplanes between adjacent cells cut the initialization geometry into convex polyhedra. Non-cell geometry is filtered out based on the L_∞ Voronoi criterion, leaving only the non-convex cell geometry. Eventually we conclude with discussions on the algorithms complexity, numerical precision and analyze the applicability of our generalized L_∞ diagrams for the construction of Centroidal Voronoi Tessellations (CVT) using Lloyd's algorithm.

CCS Concepts
• Computing methodologies → Mesh geometry models; Mesh models; Volumetric models;

1. Introduction

Key motivation for the generation of Voronoi diagrams is the analysis of spatial data and, therefore, generalized Voronoi diagrams find application in various scientific disciplines whenever closest-point associations or proximity based spatial partitioning are involved: E.g., ranging from graph labeling [WTLY13] and voxelization [VKK*03] over urban planning [Now15] and models of biological cells [BTKA10] to applications in Very Large Scale Integration [PL01]. Special L_p Voronoi diagrams are the $p = 2$ case with planar bisectors between two adjacent cells or $p = 1$ and $p = \infty$ where the bisectors are at least partially planar and not curved

as with other ps. The level-set of the L_∞ metric is a hypercube and lends itself to applications in quad- and hex-meshing, i.e., employed in Lloyd relaxations for constructing Centroidal Voronoi Tessellations (CVT) with square or cube-shaped cells. There exist many established concepts for the construction of Voronoi diagrams in multidimensional Euclidean L_2 space. But these algorithms do not trivially translate to vector spaces using other generalized L_p distance metrics. Therefore, we focus this work on an algorithm for the construction of generalized meshed L_∞ Voronoi diagrams in 2D and 3D. Our approach is designed around individual cells, initialized with a meshed representation of the L_∞ level-set, a hypercube. We exploit a characteristic of the L_∞ *max*-norm, such

that the non-trivial bisectors are determined separately for the sites individual orientation axes. Bisector hyperplanes cut the initialization geometry into convex fragments, eventually clipped by evaluating their closest site association (Voronoi criterion). We further extend the metric's definition with a spatially varying orientation field and anisotropic weighting, thus also generalizing weighted L_∞ Voronoi diagrams. This allows for cells to align with given input structures, e.g., when employed for CVT based meshing. An example is shown in Figure 1f where the orientation field aligns to normals and principle curvature directions of a trefoil knot mesh.

Main contributions presented in this paper extend the generation of Voronoi diagrams in several domains: **Dimensionality:** We present examples for 2D and 3D, parts of the concept also translate to higher dimensions. **Orientation:** Sites are not subject to the orientation of one prevalent uniform metric space but can be decoupled and individually oriented. **Weighting:** Our algorithm allows to specify weights for individual sites as in weighted Voronoi diagrams. **Anisotropy:** Site-individual weight vectors that correspond to the site's main axes also allow for anisotropic cell extents.

2. Definitions

An introduction to terms and principles, used in upcoming sections.

L_p **norms**   are defined for finite-dimensional spaces $\mathbb{R}^n$ where the length of a vector $\mathbf{x}$ and a real number p is given as

$$||\mathbf{x}||_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}. \tag{1}$$

For $p = 2$ it simply is the Euclidean distance, whereas the limit of this norm with $p \to \infty$ is the Chebyshev or *max*-norm

$$||\mathbf{x}||_\infty = \lim_{p \to \infty} \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}} = \max_i |x_i|. \tag{2}$$

With $p \to \infty$ the norm then corresponds to the maximum absolute extent over the available dimensions in $\mathbb{R}^n$.

L_p **Voronoi diagrams**   specify the tessellation of a given domain with cells. This segments space into regions of closest proximity, relative to a given set of generating points, called sites. The cell C_i of a site S_i is herein defined by

$$C_i = \left\{\mathbf{r} \in \mathbb{R}^n \mid ||\mathbf{r} - S_i||_p \leq ||\mathbf{r} - S_j||_p \; \forall \, i \neq j\right\}. \tag{3}$$

L_∞ **Cells**   follow by the definition of Equation (3) with $p = \infty$. But to generalize the concept, we extend the definition of a site S_i: A matrix $\mathbf{M}_i$ specifies a non-axis-aligned site orientation and scalar λ_i a site's individual weight. More generally we can also use a $2n$-dimensional vector Λ_i to define anisotropic weights for the individual signed directions. The extended definition for the length of a vector between a point $\mathbf{r}$ and site S_i is then given by

$$||\mathbf{r} - S_i||_{\widetilde{\infty}} = \max\left(\text{diag}(\Lambda_i)^{-1} \cdot \begin{bmatrix} \mathbf{M}_i^T \\ -\mathbf{M}_i^T \end{bmatrix} \cdot (\mathbf{r} - S_i)\right). \tag{4}$$

This evaluates the dot-product of the vector with a $2n \times n$ matrix (here in block notation) and an elementwise division by the $2n$ entries of the Λ_i vector. To actually account for individual weights

of the signed axes, the norm results directly as the maximum of the $2n$ entries and not their absolutes. Setups where all $\mathbf{M}_i = \mathbf{I}$ and $\Lambda_i = [1, \ldots, 1]$ correspond to default L_∞ diagrams with axis-aligned and equally weighted sites. Such default cells initialize as shown on the left in Figure 2. Vectors $\pm\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$ are signed columns of the orientation matrix: The shown pyramidal segmentation clusters points around a site that have their maximum distance along the same dimension, then called *max*-dimension. Therefore, we refer to vectors associated with the individual pyramids as *max*-vectors. Exemplary level-set and initialization for a site with anisotropic weights are shown on the right of Figure 2.

3. Related Work

Euclidean L_2 norm: Fortune's sweepline algorithm [For87] is a robust concept to construct Voronoi diagrams in 2D using the L_2 norm. The Bowyer-Watson algorithm [Bow81, Wat81] formulates an approach for the computation of Delaunay graphs on a given set of points; a Voronoi diagram trivially follows as the graphs dual. While this algorithm generalizes for higher dimensions, it is based on the Euclidean L_2 norm. QHull [BDH96, Sci20] allows for computing Voronoi diagrams in up-to 9D, but is limited to the L_2 norm and uniformly weighted cells. While these approaches in principle generalize to other norms [Ma00], they have been nearly exclusively considered for the L_2 norm.

Beyond L_2:   Only very few practical algorithms for Voronoi diagrams under the L_∞ norm have been considered. Rendering L_2 or generalized L_p diagrams by sampling is trivial even in higher dimensions as it is solely an extension of the z-Buffer approach, thus also suitable for fast GPU-implementation [HIKL*99]. Common tools for computing actual geometry and topology of L_∞ Voronoi diagrams or their Delaunay graph, respectively, are still quite limited. The work of Dey [Dey15] extends CGAL [CGAL22] with the possibility to compute L_∞ diagrams of points and line segments. However, this algorithm is also limited to 2D and uniformly oriented (axis-aligned) and weighted sites only. The algorithm proposed by Eder & Held [EH19] considers weighted 2D sites and boxes with axis-aligned orientations. Liu et al. [LPL11] thoroughly analyze the construction and output complexity of 2D L_1 and L_∞ Voronoi diagrams based on a k-nearest-neighbor graph. Boissonnat et al. [BNN10] explore Voronoi diagrams beyond metrics in general but in context of Bregman divergences; a measure of difference that is neither necessarily symmetric nor satisfies the triangle inequality.

Applications: In Centroidal Voronoi Tessellations (CVT) [LWL*09] a diagram's site positions coincide with centroids of their associated cells. As this property correlates with uniform site distribution, CVTs and their Delaunay dual are commonly used in (re)meshing applications [CSAD04, ADVDI05, YLL*09]. While L_2 Lloyd iterations [Llo82] are known to converge on CVTs [DEJ06], their dual graphs are *only* triangle- and tetrahedral-meshes. However, many applications tend to favor meshes predominantly featuring quadrilateral or hexahedral elements. This can be approached by exchanging the L_2 norm for the generalized L_p norm with a large p or its limit with $p = \infty$. Hausner [Hau01] proposed an image stylization technique for simulating mosaic tilings with content aligned L_∞ Voronoi cells. The approach is sampling
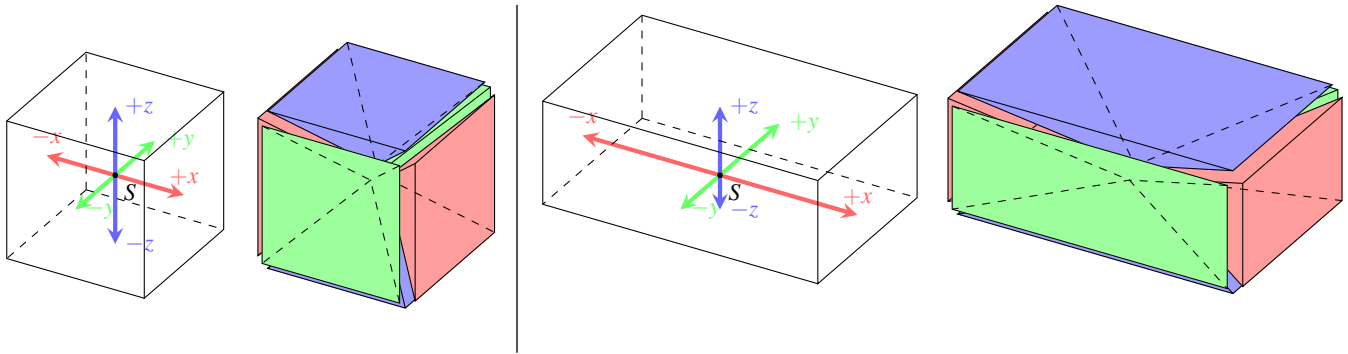
Figure 2: *In 3D, a cell's geometry is initialized as six quad-based pyramids, forming a cube. They align to the site's orientation and meet with their apices at site S. The right example illustrates an anisotropically scaled initialization for a weight vector* $\Lambda_S = \left[2, \frac{3}{2}, 1, 2, 1, \frac{1}{2}\right]^T$.

based and operates solely on pixels in the 2D image space. Mouton and Béchet [MB12] propose a concept for L_∞ Voronoi diagram based quad-meshing. However, one of the strongest arguments for quad/hex over tri/tet meshing is the ability of primitives to align to certain features of the input domain. This is not possible in the approach of Mouton and Béchet as their sites are all uniformly axis aligned. Lévy and Liu [LL10] formulated a Lloyd-Newton approach on L_2 diagrams, minimizing an L_p energy term (using $p = 8$) for individually oriented cells to generate input-aligned quad-dominant surface and hex-dominant volume meshes. Individually input-aligned cells are also featured in the concept of At-Most-Hexa Meshes [BTL22], which utilizes a flood-fill based labeling process to approximate L_∞ Lloyd relaxations, eventually limited in resolution by available GPU memory. Other recent works also focus on GPU based construction of meshless (L_2) Voronoi diagrams [RSLL18] or analogously (L_2) Power diagrams [BAR*21].

4. Concept

This section provides a brief introduction to the challenges one has to face when constructing L_p Voronoi diagrams where $p = \infty$. After an overview of our concept, we derive the geometric constraints for the diagram, eventually constructed in Section 5.

4.1. Problem Statement

From an L_2 standpoint, the construction of a Voronoi diagram (respectively its cells) is rather simple [AKL13]. Adjacent cells are separated by bisectors, which are hyperplanes centered on, and oriented orthogonal to the direct connection between two neighboring sites. Conversely, two sites are neighbors, and their cells adjacent, if they share a common bisector. Neighborhood is also defined by the Delaunay criterion, solely based on site positions. Vertices of the L_2 Voronoi cells follow as the circumcenters of Delaunay simplices. As illustrated in Figure 3i, the diagram's topology follows as the Delaunay graph's dual.

In the L_∞ scenario, the same definitions of neighborhood and adjacency apply, but the construction is no longer trivial: Bisectors are not necessarily planar; cell vertices do not unambiguously follow as circumcenters from a Delaunay-like triangulation; and adjacency not only depends on distances but with our extended metric's definition also on site orientations and weights.
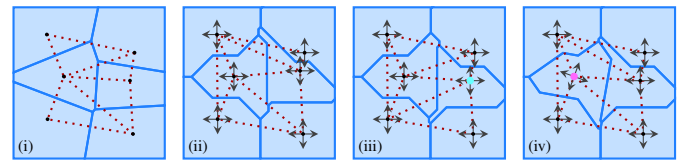
Figure 3: *L_2 (i) and L_∞ (ii,iii,iv) Voronoi diagrams, where site positions and orientations both shape the adjacency graph (dotted). In (iii) the ● site moved downwards. In (iv) the ● site is rotated.*

4.2. L_∞ Voronoi Cells

From the definition of Voronoi cells in Equation (3) and our extended distance norm in Equation (4) we can now infer two things:

1. The level set of this distance function is a hypercube, respectively a square in 2D or a cube in 3D. There are hyperplanes around a site, separating regions of the prevalent *max*-dimension according to the distance definition. The level set and separated *max*-dimensions are visualized in Figure 2.

2. The bisector separating two adjacent cells, is defined as the locus of points with equal distance to both cell's sites. It is formally expressed for sites A and B as

$$d(A, B) = \left\{ \mathbf{r} \in \mathbb{R}^n \mid ||\mathbf{r} - A||_{\widetilde{\infty}} = ||\mathbf{r} - B||_{\widetilde{\infty}} \right\}. \quad (5)$$

The bisector between two cells is only linear (or planar) within regions where the prevalent *max*-dimensions of both cells are constant. Within cells there are also points where the *max*-dimension is actually indefinite, as the distance to the site is identical for two different dimensions. This corresponds to the triangular faces separating the *max*-pyramids in 3D and is visualized in Figure 4 with dashed lines. At points where the indefinite *max*-dimension also equals the distance to another site, the bisector has a kink.

The examples in Figure 4 show two sites A and B with their non-convex L_∞ cells. With uniform site orientations $\mathbf{M}_A = \mathbf{M}_B$ in Figure 4i, the bisector between the cells still features symmetric properties. The orientation $\mathbf{M}_B$ of site B is rotated in Figure 4ii, causing the bisector to feature more than just two kinks. In Figure 4iii site A is moved closer to B, illustrating a case where cells now even split into separated regions. For non-equal site weights $2\lambda_A = \lambda_B$ as in Figure 4iv one cell may even fully enclose another cell.
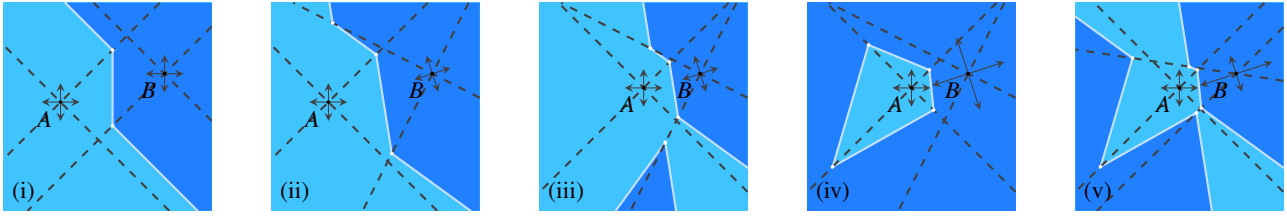
Figure 4: *L_∞ Voronoi cells, where (i) $\mathbf{M}_A = \mathbf{M}_B$; (ii) $\mathbf{M}_A \neq \mathbf{M}_B$; (iii) A is moved, initiating the fragmentation of B's cell into separate regions; (iv) B's weight λ_B is doubled, now enclosing A's cell; (v) Λ_B is scaled anisotropically only for one dimension.*

5. Cell Geometry

As introduced in Section 2, we can interpret the 3D space around a site as six separate regions, where a prevalent *max-dimension* is constant. This understanding lets us formulate the upcoming algorithm and is also used in this first step to evaluate neighborhood relations. Notes for a numerically robust realization are further detailed in Section 6.3.

5.1. Algorithm Outline

As Figure 3 shows, the construction of L_∞ diagrams is challenging due to the complexity of possible influences on a cell. Thus, we designed our algorithm with the focus on individual cells and their local neighborhood. It is outlined with the following main steps:

- **Initialization:** Each cell starts as a simple base mesh and determines possible neighbor sites.
- **Separation:** Bisector planes between valid neighbor configurations virtually cut the base geometry into separate parts.
- **Extraction:** What parts eventually compose the final cell are determined by evaluating the Voronoi criterion (Equation (3)), allowing to extract the non-convex boundary of the resulting cell.

5.2. Initialization and Adjacency

Base Mesh: As visualized in Figure 2, cells are initialized with a meshed representation of their scaled level-set, a hypercube. Regions of constant *max-dimensions* around a site are six quad-based pyramids, located with their apices on the site's position and rotated to its orientation accordingly. In theory, these pyramids are of infinite height, but as basis for our meshing algorithm we chose a finite height. With a known site distribution density, the height is scaled inversely to that. Another heuristic would be the *max-distance* to a closest neighbor site in all signed directions, or if there is none the distance to the domain limits. As detailed further in Appendix A, these heuristics provide a simple and sufficient base for most regular arrangements, but can be extended with a safety-radius factor to also perform robustly on random input. Thus, the geometry of all cells is now set to be initialized as six scaled pyramids forming the initialization cubes. For simplified 2D illustrations, this analogously translates to four *max-triangles* around a site.

Neighborhood: An individual cell's geometry is eventually defined by the boundaries to its direct adjacent neighbors. For L_2 cells, adjacency follows from the Delaunay criterion, as three points define a unique circle (2D) or four points a unique sphere (3D).

There is no trivial analogous criterion in the L_∞ case as three points may not define a unique square (2D), respectively four points not defining a unique cube (3D). This holds true even without the generalization of individual site orientations but assuming uniformly axis-aligned sites [Dey15]. As visualized in Figure 5, we employ the cell's initialization geometry as a proximity heuristic such that volumetric overlap accounts for a nearest-neighbor connection. In this example, all initialization cubes overlap and the neighborhood is fully connected. Sites wrongfully incorporated in a nearest-neighborhood, i.e., cells which are eventually not really adjacent, do not impair the algorithm's outcome.
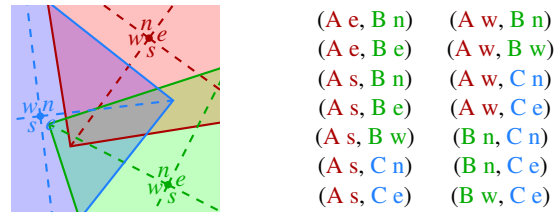

(A e, B n)	(A w, B n)
(A e, B e)	(A w, B w)
(A s, B n)	(A w, C n)
(A s, B e)	(A w, C e)
(A s, B w)	(B n, C n)
(A s, C n)	(B n, C e)
(A s, C e)	(B w, C e)

Figure 5: *Sites A, B and C with initialization geometry on the left. We enumerate the max-vectors and associated pyramids of each site with n, e, s, w. Tuples on the right list intersecting max-triangles (pyramids in 3D), for which bisector planes will be determined.*
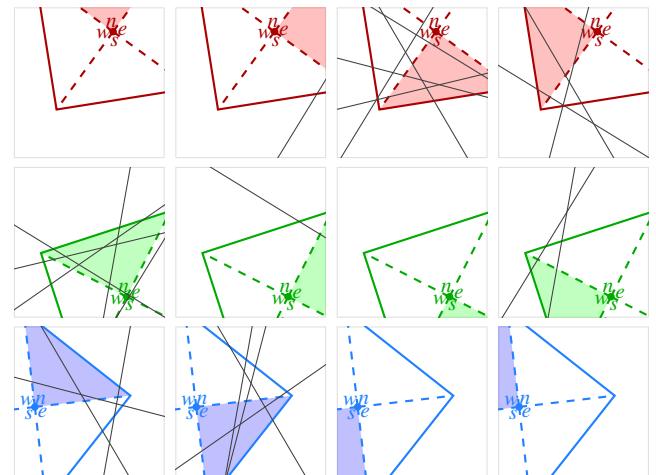

Figure 6: *The planes listed in Figure 5 only cut their associated max-pyramid. Therefore, some pyramids are not cut at all and only clipped by the extents of the meshed domain.*

5.3. Bisector Plane

Each bisector plane corresponds to a separation of pairs of site-*max*-vector combinations as listed on the right of Figure 5. We will now infer a bisector plane as visualized in Figure 7 for two sites A and B. The associated *max*-vectors $\mathbf{a}$ and $\mathbf{b}$ correspond to signed columns of their orientation matrices. Plane P_{AB} is implicitly defined from the locus of points $\mathbf{r}$ with equal *max*-distance to A and B and has the form $0 = (\mathbf{r} - \mathbf{o}_{AB}) \cdot \mathbf{n}_{AB}$.
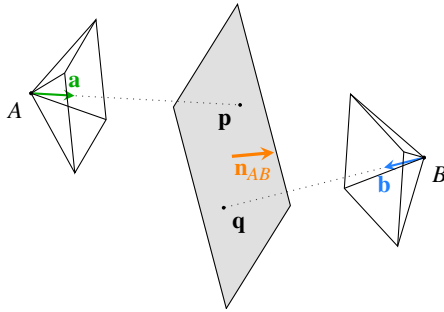

Figure 7: *The (infinite) bisector plane P_{AB} is visualized with the gray rectangle and normal $\mathbf{n}_{AB}$. Pyramids of A and B actually intersect but are scaled down, to avoid visual clutter.*

As shown in Figure 7, we can use any of the two points $\mathbf{p}, \mathbf{q} \in P_{AB}$ as origin $\mathbf{o}_{AB}$. Point $\mathbf{p}$ is the projection of A onto the plane with $\mathbf{p} = A + \mathbf{a}s$. Conversely we know that $s = (\mathbf{p} - B) \cdot \mathbf{b}$. Solving for s gives the following definition of $\mathbf{p}$, where $\mathbf{q}$ follows analogously

$$\mathbf{p} = A + \mathbf{a}\frac{(A-B) \cdot \mathbf{b}}{1 - \mathbf{a} \cdot \mathbf{b}} \qquad \mathbf{q} = B + \mathbf{b}\frac{(B-A) \cdot \mathbf{a}}{1 - \mathbf{a} \cdot \mathbf{b}}.$$

Further can we construct a third point $\mathbf{r}$ on the line between A and B, lying in the plane, thus $(\mathbf{r} - A) \cdot \mathbf{a} = (\mathbf{r} - B) \cdot \mathbf{b}$. With these three points we can derive the normal, resulting in

$$\mathbf{n}_{AB} = \frac{\mathbf{a} - \mathbf{b}}{\|\mathbf{a} - \mathbf{b}\|}.$$

Exceptions: Bisector planes not intersecting the meshing domain are discarded, as well as cases where $\mathbf{a} \cdot \mathbf{b} = 1$ as such planes lie at an infinite distance. However, if also $\mathbf{a} \perp A - B \perp \mathbf{b}$, then there are ambiguous regions of points with identical distances to both sites. In Figure 8i the gray areas are not uniquely associable with one cell. Established conventions [Dey15, EH19] resolve this with double-assignment, random or lexicographical ordering [PL01]: I.e., assign both regions to both cells, or assign the *left* region to the first site and the *right* region to the second. As shown in Figure 8ii, an imposed assignment avoids artificial segmentation of equidistant regions. Nevertheless, we opted for the solution shown in Figure 8iii, as its three-dimensional analog results in more intuitive solutions than pseudo-ordered assignments. This segmentation represents a fallback from the *max* to the second largest dimension, where the bisector plane is then simply defined by

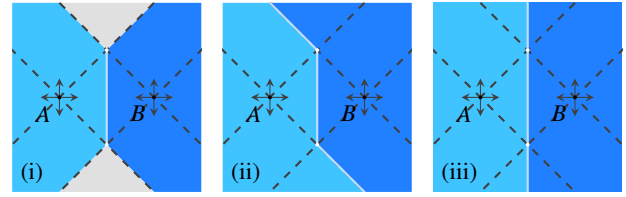$$\mathbf{o}_{AB} = \frac{A+B}{2}, \qquad \mathbf{n}_{AB} = \frac{A-B}{\|A-B\|}.$$

Figure 8: *Gray regions (i) are equidistant to A and B, thus not uniquely associable to either cell. Imposed assignments (ii) resolve this ambiguity with pseudo ordering. Our employed convention (iii) is a fallback from the max to the second largest dimension.*

5.4. Orientation and Weighting

The level-set of the L_2 metric is a hypersphere, thus invariant to orientation changes. Usually L_p Voronoi diagrams with $p \neq 2$ assume all sites to be oriented in coherence with the space's coordinate axes, as underlying distance measures are based on the global L_p metric. We get past this limitation by introducing rotation matrices $\mathbf{M}_i$ as individual orientations for sites S_i. Further is a Voronoi diagram generalized by introducing individual site weights. For L_2 this simply corresponds to altered radii of their hypersphere level sets. The generalized L_∞ distance in Equation (4) for a point $\mathbf{r}$ and site S_i incorporates the orientation matrix and anisotropic weights vector Λ_i. An exemplary outcome of this is shown in Figure 4iv, where the increased weight of site B is visualized with scaled basis vectors and caused the complete enclosure of A's cell.

The new bisector plane is then defined by any of the points $\mathbf{p}, \mathbf{q}$ as origin $\mathbf{o}_{AB}$ and plane normal $\mathbf{n}_{AB}$ with

$$\mathbf{p} = A + \mathbf{a}\frac{(A-B) \cdot \mathbf{b} \cdot \lambda_A}{\lambda_B - \mathbf{a} \cdot \mathbf{b} \cdot \lambda_A} \qquad \mathbf{n}_{AB} = \frac{\frac{\mathbf{a}}{\lambda_A} - \frac{\mathbf{b}}{\lambda_B}}{\left\|\frac{\mathbf{a}}{\lambda_A} - \frac{\mathbf{b}}{\lambda_B}\right\|}.$$

$$\mathbf{q} = B + \mathbf{b}\frac{(B-A) \cdot \mathbf{a} \cdot \lambda_B}{\lambda_A - \mathbf{a} \cdot \mathbf{b} \cdot \lambda_B}$$

Prior, we would discard planes when $\mathbf{a} \cdot \mathbf{b} = 1$ as the denominator in the plane origin term is 0. However, with $\lambda_A \neq \lambda_B$, this no longer holds true and the plane origin is no longer at an infinite distance, thus not discarded. For the exception mentioned before where $\mathbf{a} \perp A - B \perp \mathbf{b}$, the fallback normal definition is the same, but for the origin we have to incorporate the weights as

$$\mathbf{o}_{AB} = \frac{A \cdot \lambda_B + B \cdot \lambda_A}{\lambda_A + \lambda_B}.$$

This concept easily generalizes further as a site is not limited to one weight, applied uniformly in all dimensions. λs can be specified freely for all six $\pm\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$ extents in a Λ vector. For the example in Figure 4v, only one dimension of B is scaled anisotropically. Site individual orientations and decoupled Λ weight vectors as defined in Equation (4) allow for the most generalized anisotropic L_∞ Voronoi diagram possible.

One scalar λ_i for all extents of a site S_i affects all its bisector planes equally. For individual weights in a Λ_i vector, this is not the case and has to be accounted for in the transitions between the prevalent *max*-dimension sectors. In Figure 4v with anisotropically scaled weights, these sectors (separated by dashed lines) widen or narrow, respectively. We reflect this in our geometric concept by scaling

the initialization pyramid-cube accordingly as shown on the right in Figure 2. This guarantees that bisector planes of adjacent pyramids intersect the separating face at the same points.

Dimensionality The general L_p norm in Equation (1) and analogously our generalization in Equation (4) are defined in $\mathbb{R}^n$. Intuition suggests that the cut & clip principle of our algorithm (detailed in upcoming Section 5.6) also translates analogously to higher dimensions with masking based on the Voronoi criterion and a certain *max*-dimension. Equidistant points **p** and **q** with respect to A and B and normal $\mathbf{n}_{AB}$ for the definition of bisector planes also follow the general definition in $\mathbb{R}^n$. The initialization is generalized with hypercubes, then to be intersected with hyperplanes. As a proper realization in $\mathbb{R}^n$ with $n > 3$ is left open for future work, the claim for applicability in higher dimensions is only theoretical at this point.

5.5. Valid Bisector Planes

As described before, bisector planes are only valid in certain regions between adjacent sites and the actual non-planar bisector assembles from finite portions of different planes. In the next step, the initialization geometry will be virtually cut with the determined bisector planes. Signed $\pm\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$ dimensions give a total of 6 possible *max*-dimension regions around a site (Figure 2). For two sites A and B, the definition above yields 6×6 possible planes. But not all 36 planes are part of the bisector, i.e., the cell geometries of A and B. As shown in Figure 6, some planes do not intersect their associated initialization geometry at all and can be neglected. Then there are also planes which intersect the geometry but are not part of a bisector. It is a hard problem to determine which planes actually *are* part of the bisector, thus of the final cell geometry. *Unneeded* cut operations will not degenerate the resulting geometry.

Filtering We experimented with various geometric properties to determine the validity of certain bisector planes: Distance measures or combinations with site orientation give no reliable criterion to determine if a plane is part of a bisector or not. The example in Figure 4ii illustrates cases where constellations of *max*-vectors give valid bisectors, where they can both face or oppose each other. I.e.: $(+\mathbf{x}_A, -\mathbf{x}_B)$ and $(+\mathbf{y}_A, +\mathbf{y}_B)$ give valid planes, but $(+\mathbf{x}_A, +\mathbf{x}_B)$ and $(+\mathbf{y}_A, -\mathbf{y}_B)$ do not. The positions of **p** and **q**, i.e., where the *rays* defined by **a** and **b** intersect the bisector plane, are also no reliable indicator. In some cases **a** or **b** are scaled with a negative sign so that the intersection points **p** and **q** may lie *behind* A or B but the cut plane is valid, nevertheless.

We found it a useful filter criterion for the validity of planes to check if the corresponding pyramids of their *max*-dimension vectors intersect [GPR02], as listed in Figure 5. This refines the adjacency criterion from Section 5.2 with respect to the sites' individual orientations. Filtered results may contain false-positives (unnecessary planes labeled as necessary) but more importantly does not create false-negatives (necessary planes labeled as unnecessary). This can be guaranteed by the following logic: If the intersection of two *max*-pyramids is empty, they don't have any common point. By the definition of a bisector as a set of common points, non-intersecting pyramids can, therefore, not have a bisector.
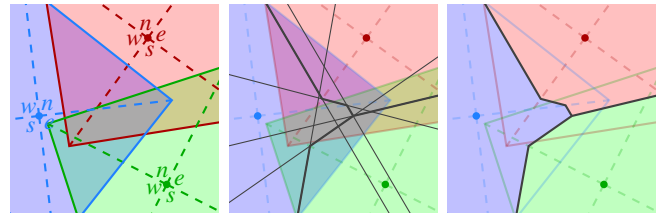

Figure 9: *Initialization geometry (left) split by valid cut planes (center). Final cell geometry (right) results by clipping cell fragments, that violate the Voronoi criterion.*

5.6. Cut & Clip Pyramids

In this step it is determined which planes are part of a bisector. Therefore, all sites' pyramids are virtually cut with their individual (filtered) sets of possible bisector planes. The center of Figure 9 and left of Figure 10 illustrate this cut geometry. Initialization pyramids are convex and cutting them with planes will also result in convex polyhedra only. Due to this convexity guarantee, centroids of these polyhedral fragments will always be inside their hull. This is important, as these polyhedra centroids are used to evaluate the Voronoi criterion from Equation (3) for each polyhedron with respect to their associated cell's site and neighboring sites. Polyhedra where the centroid violates the criterion, i.e., is *closer* to another site, are clipped from the cell's geometry. Appendix A provides a more detailed discussion on the conceptual validity of this approach. The concept is visualized in Figure 9 with cut geometry in the center and clipped results, i.e., the final cell extents, on the right. Figure 10 shows the fragmented initialization geometry of an exemplary isolated 3D cell. Fragments highlighted in yellow mark the polyhedra removed from the cell. The final non-convex L_∞ cell geometry eventually results from the polygons that enclose the remaining non-clipped polyhedra.
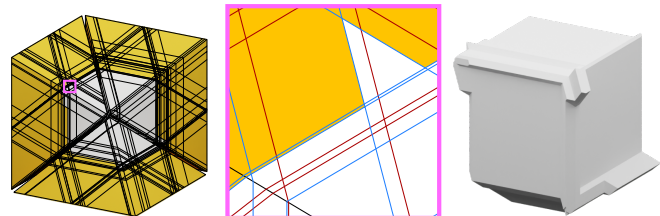

Figure 10: *The cut & clip algorithm on the initialization pyramids. The exposed inner of a cell (left) shows only the $\{-\mathbf{x}, +\mathbf{y}, -\mathbf{z}\}$ pyramids with their cuts. Clipped geometry is highlighted in yellow. Valid and invalid cuts are marked in the zoom-in (center). The eventually extracted geometry for the L_∞ cell is shown on the right.*

5.7. L_∞ Centroids

In upcoming experiments we utilize L_∞ Voronoi diagrams in a Lloyd relaxation. Therefore, we also need to specify what a cell centroid represents in the L_∞ norm. A polytope centroid can be generally specified as the minimizer of an energy function over its volume, in our case a cell C. For the L_p metric, the energy of a centroid $\mathbf{c}_0$ computes as

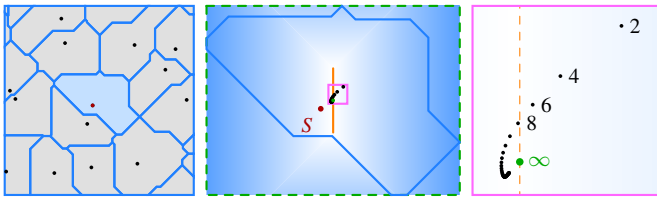$$F_{L_p} = \sqrt[p]{\int_C \|\mathbf{y} - \mathbf{c}_0\|_p^p \, \mathrm{d}\mathbf{y}}. \tag{6}$$

Figure 11: *An isolated non-convex L_∞ cell and its bounding-box (center, dashed). The zoom-in (right) highlights the trace of L_p energy minimizing centroids at $p \in \{2, ..., 96\}$. With increasing p, the centroids approach the cell's bounding-box center at $p = \infty$.*

Figure 12: *S_A and S_B's cell bounding-boxes (dashed) largely overlap, due to the sites close proximity. Contrary to centroid updates in L_2 Lloyd iterations, the L_∞ centroids c_A and c_B (bounding-box centers) are not pushed apart but remain close.*

In the L_2 case, this energy is minimized by the center of mass. With our generalized L_∞ norm in Equation (4), the L_∞ centroid results as the point with the smallest *max*-distance to any point in the cell, as explained below.

With increasing p, distances to farthest cell points dominate the energy term. As p eventually transitions to ∞, the energy to be minimized equals the absolute *max*-distance of the centroid to the cell's extents in any axis-aligned direction. In Figure 11 (middle), the blue gradient visualizes the evaluated cell energy for all points within the cell's bounding-box. For a non-square shaped bounding-box, there is actually not only a single point minimizing the L_∞ energy but a line segment of points with equally minimal energy (orange). Plotted on the right of Figure 11 are the L_p energy minimizing centroids for increasing p. As they approach the center of this minimal L_∞ energy segment, we use this (bounding-box) center point as a proxy for the L_∞ centroid.

6. Experiments and Discussion

This section presents experiments with our L_∞ Voronoi diagrams employed in Lloyd relaxations, a common application for generating CVTs. We further analyze and compare our algorithm's complexity, evaluate and discuss our results.

6.1. Lloyd Relaxation

The goal to employ the L_∞ norm in a Lloyd relaxation is to form CVTs with quad- or hex-like cells due to its distinct square or cube shaped level set. Thus, it found use in applications related to quad- and hex-meshing [Hau01, LL10, BTL22].

Analogously to the energy of a single cell F_{L_p} (Equation (6)) the global energy is computed as the integral over the whole diagram's domain. Lloyd's algorithm now minimizes this global diagram energy by alternating two steps: Compute Voronoi cells for all sites, then relocate sites to the centroids of their cells. With our method we are now able to compute real L_∞ Voronoi cells, respectively centroids, to perform L_∞ Lloyd relaxations.

With the definition of the L_∞ centroids from Section 5.7, scenarios as shown in Figure 12 may arise. Sites in close proximity can form cells of almost triangular shape in such a way that their bounding-boxes largely overlap. Bounding-box centers are located on the *diagonal*, close to the triangular cell's boundary and, therefore, these new centroids are again very close. Unlike the L_2 centroids, when
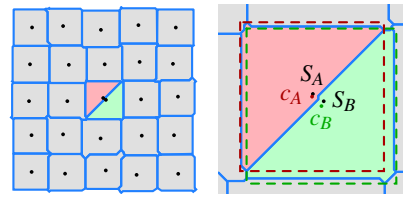
employed in a Lloyd relaxation, the L_∞ centroids may not necessarily repulse each other. Pairs of centroids can move onto the same position, creating a numerical ambiguity in the diagram.

Site-Merging A possible solution for the issue of two sites falling onto the same point could be to simply merge them, i.e., remove one of them in further iterations. Theoretically, repeated merge operations would allow for all cells in a diagram to eventually collapse into one. However, merging close sites with triangular cells creates one quad cell where the centroid then resides in a very stable state, far inside its cell. Thus, we could in practice not construct an initialization to provoke such a successive collapse. While this operation perfectly resolves the anomaly shown in Figure 12 and does not conflict with the energy minimization, it is only a local improvement. This can be seen in the center and right examples in Figure 21: Local energy minima can manifest as non-quad cells in a non-orthogonal alignment but remain unaffected by merge operations as their centroids are not close. The distance threshold for merging two sites was set fix for these examples but with largely varying site weights, thus cell sizes, it should be scaled accordingly. Further, if the number of sites in a diagram is a fix constraint, only removing them during the relaxation is not an option. Instead, sites could be teleported [CSAD04], but the best spawn positions are non-trivial to determine and so far left open for future work.
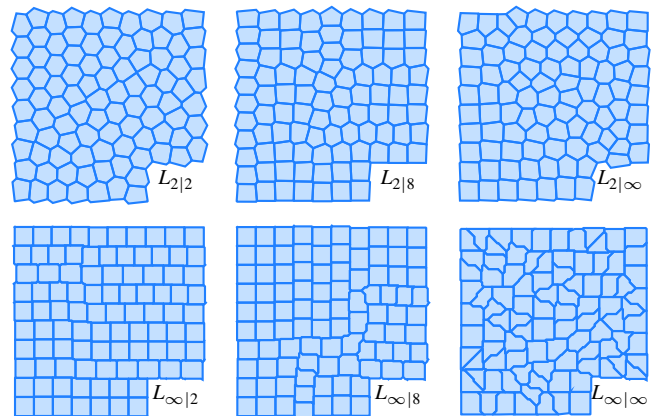

Figure 13: *L_2 (top) and L_∞ (bottom) diagrams after 100 Lloyd iterations, with L_p energy minimizing centroids using $p \in [2, 8, \infty]$. L_p energy plots for each combination are featured in Figure 14.*
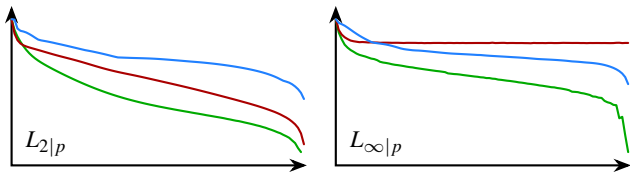
Figure 14: *Normalized plots on a logarithmic **y**-axis of L_2 (left) and L_∞ (right) Voronoi diagrams' energy F_{L_p}, being minimized over 100 Lloyd relaxations (**x**-axis) using $p \in \{2, 8, \infty\}$.*

Convergence Figure 13 compares Lloyd-relaxed Voronoi diagrams after 100 iterations each, using different *p*-combinations for cell and centroid computation. For better readability we refer to mixed L_p computation combinations as $L_{\text{cell}|\text{centroid}}$ relaxations: In $L_{2|\infty}$, for example, the Voronoi diagram is computed with Euclidean L_2 cells and the centroids for the relaxation are determined with the Chebyshev L_∞ norm. Unsurprisingly, standard $L_{2|2}$ diagrams converge with strictly decreasing energy towards hexagonal shaped cells as this is the most dense packing of circles in the plane [CW10]. Analogous to the setup of Lévy and Liu [LL10] the examples also feature $L_{2|p}$ diagrams using standard Euclidean L_2 cells and centroids minimizing F_{L_p} with $p = 8$. These, and our $L_{2|\infty}$ diagrams feature quad-like cells but the overall structure is not as regular and quad-like as anticipated.

Intuition would suggest to see the best quad-cell arrangement in $L_{\infty|\infty}$ diagrams, where cells' bounding-box centers are used as true L_∞ centroids. However, as described above, the bounding-box centroid also allows for constellations of two sites moving to the same position, as their triangle-shaped cells have overlapping and almost identical bounding-boxes. On the other hand, if we combine our L_∞ cells with L_2 and L_8 centroids, the diagrams approach the anticipated quad-like cell layout ($L_{\infty|2}$).

Plots of the global diagrams' energies over the course of the relaxations from Figure 13 are shown in Figure 14. Energy results are normalized to a maximum of 1. With a logarithmic **y**-axis scaling, the behavior of the energy term in different L_p combinations becomes prominently visible. The plots show that a strictly decreasing energy can only be guaranteed with the same norm for cell and centroid computation. In L_2 diagrams (left) the energy for L_p centroids with $p = 2$ steadily decreases as do $p = 8$ and $p = \infty$. With our L_∞ diagrams (right) the $p = \infty$ energy again monotonically decreases. The L_2 centroids reach a stable energy plateau early on. Kinks towards the end of the L_8 energy plot are numeric issues due to the normalization and logarithmic axis-scale.

Traces shown in Figure 15 visualize individual sites' positions over the course of a Lloyd relaxation in 3D, starting on a random initialization. The overlapping bounding-box issue also occurs in three-dimensional diagrams and bounding-box centers pairwise approach a common position. This phenomenon is visualized as nearest-neighbors distances in blue (far) and red (close). Thus, Euclidean L_2 centroids become blue as they increase the distance to their neighbors. On the other hand, Chebyshev L_∞ centroids do not stray apart and tend to cluster. The 3D CVT results in Figure 22 further exemplify this phenomenon.
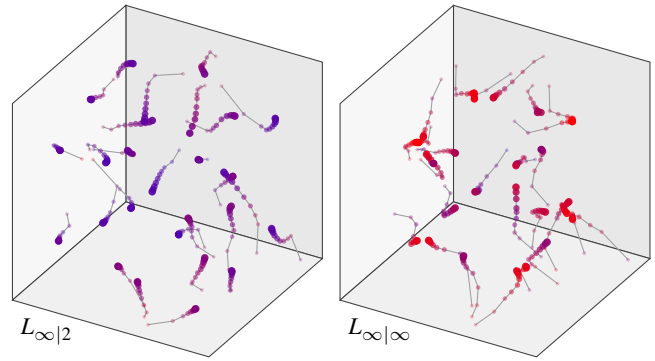

Figure 15: *Sites' traces in Lloyd relaxations with hybrid $L_{\text{cell}|\text{centroid}}$ Voronoi diagrams. Distances to nearest neighbor sites are visualized from far to close. Euclidean centroids strive apart from each other while Chebyshev centroids tend to cluster.*

6.2. Computational Complexity

The mathematically expected complexity for Voronoi diagram construction algorithms of *n* points is well studied in Euclidean L_2 space. Under a constrained *quasi-uniform* point distribution in the unit *d*-ball, Dwyer [Dwy91] proofed the possibility of an algorithm performing in linear time. The common Bowyer-Watson algorithm [Bow81, Wat81] for computing Delaunay graphs has an average complexity of $\mathcal{O}(n \log n)$ but can degenerate to $\mathcal{O}(n^2)$ for certain input configurations [Reb93]. Aurenhammer et al. [AKL13] summarize various proofs for the general construction complexity of L_2 Voronoi diagrams: The problem can be reduced to the sorting of *n* values, thus being bounded by $\mathcal{O}(n \log n)$ [For87]. In L_2 it can be also formulated as the intersection of $n - 1$ halfplanes with a complexity of $\Theta(n \log n)$ [PS85].

For *d*-dimensional L_∞ Voronoi diagram construction techniques, the list of available algorithms for comparison is very limited. Liu et al. [LPL11] formulated their 2D approach based on *k*-nearest-neighbors in a Hanan Grid [Han66] and provide an upper bound of $\mathcal{O}(\min\{k(n-k), (n-k)^2\})$ for *n* points. Our approach is related to this concept of a nearest neighborhood, but not limited to a fixed size *k*. Herein the *proximity* measure for neighbors is the (oriented) *max*-norm distance. Therefore, we employ the initialization geometry pyramids from Figure 2, where two sites qualify as neighbors if one of their pyramids intersect each other, respectively. For a known site distribution, the pyramids' height *h* can be scaled proportionally inverse: I.e., for n^3 points distributed regularly in a cube shaped domain with extent *e*, each cell ideally spans over $\frac{e}{n}$ in each dimension. Without any prior knowledge on the distribution density, a height $h = e$ corresponds to the worst case of $k = n - 1$, which can be avoided using the heuristic described in Section 5.2. Assuming sites are jittered within the extent of their ideal cell, a pyramid height of $h = \frac{2e}{n}$ is sufficient to cover all relevant neighbors. In CVTs, sites approach regular, evenly spaced distributions and the neighborhoods are reduced to an almost ideal size. I.e., with uniform orientations, this corresponds to $k = 8$ in 2D and $k = 26$ in 3D. For the case of non-uniformly oriented sites, *h* needs to be scaled by $\sqrt{d}$ (for *d* dimensions), respectively.
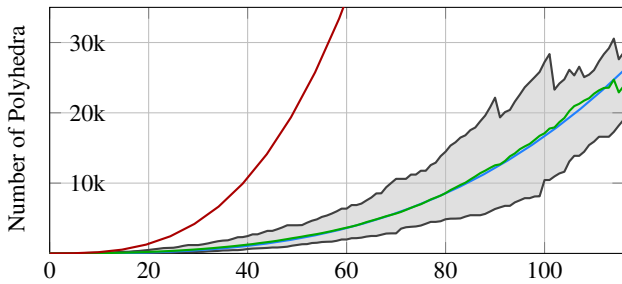
Figure 16: *Number of polyhedra for m hyperplane cuts (horizontal axis). The gray area is enclosed by the overall max. and min. number of polyhedra for this number of cuts. Other lines plot the* mean, cake-number *and a polynomial approximation* $\frac{1}{60}m^3$.

Cutting the initial geometry of each cell with m planes has a worst case complexity, theoretically bound by the mth cake-number [GMJ87]. As shown in Figure 16, the actual complexity is drastically smaller as not every hyperplane cuts the maximum possible amount of polyhedra. On average, we found $\frac{1}{60}m^3$ as a reasonable estimate for the cell-cut operation's complexity. In practice m never exceeded 150 (in CVTs < 100) as it is naturally limited by the constellation of the nearest-neighborhood and can be considered a local property. Thus, in combination with a suitable kNN algorithm, our cell-focused approach operates in $\mathcal{O}(nk\log n)$ time.

Runtime   As execution timings heavily depend on a diagrams initialization specifications, Table 1 lists the average computation time *per* cell. These were measured with an implementation in Python and executed on a CPU core at 4 GHz. Cells are independent of each other, thus allow for parallel computation on a multithreaded CPU. Recent works proposed similar cell-focused Voronoi approaches, parallelized on GPU hardware [RSLL18, BAR*21] as their convex L_2 cells only require clipping operations. Data structures involved in our concept are a bit more intricate and not yet ported to a GPU implementation, although theoretically possible.

Fig.	knn (%)	P (%)	cut (%)	clip (%)	ds. (%)	t (s)	#
20(l)	5.37	0.11	87.67	2.81	4.05	0.06	25
20(c)	4.87	0.10	88.33	2.85	3.85	0.07	25
20(r)	4.17	0.09	88.82	3.51	3.42	0.09	25
21	4.40	0.13	88.41	3.21	3.86	0.10	81
17(l)	0.43	0.00	81.17	5.71	12.68	5.84	27
17(r)	0.72	0.48	82.24	6.56	10.01	6.71	125
1c	0.45	0.29	77.12	6.29	15.84	10.93	343
19(r)	1.29	0.59	80.20	5.11	12.80	8.99	1331

Table 1: *Timings of exemplary results (top: 2D, bottom: 3D). Stats on the right list the average time per cell and number of sites. Center columns break down the individual steps: knn and bisector (P) computation, the cut & clip algorithm and the final dissolve-step.*

As listed in Table 1, the *cut* operations consume the largest portion of computation time, spent on an individual cell. This is not surprising as the cutting is a subdivision iteration on a data structure, maintaining geometry and topology. Computations of bisector planes or evaluation of the clipping condition are, in contrast, simple one-step operations. The jump in computation time between 2D and 3D cells can be clearly attributed to the curse of dimensionality: Simple geometric operations become more expensive, cells have 6 pyramids

to cut instead of 4 triangles, with a lot more bisector planes induced by an also increased number of neighboring sites. The average cell computation time is mostly invariant to changes in the orientation or anisotropy fields. An increased number of sites in the diagram can affect the individual cells with more possible neighbors, thus more bisector cuts. However, with proximity heuristics, this effect can be limited by local neighborhoods of finite size.

6.3. Results

Exemplary results with different parameter configurations are shown in Figure 17. The left example is initialized with random site positions, resulting in quite prominent differences in cell size. The arbitrary orientations show as bisectors of different cells rarely align with any other normals. In the second example, sites are positioned on a regular grid with aligned orientations, both perturbed with noise. Naturally the resulting diagram is much more regular and cells close to hexahedral.

Geometric boundaries of a cell are the locus of points with equal distance to two sites, described by a hyperplane. Therefore, all cells in our diagrams are enclosed by planar polygonal faces. As pointed out in the zoom-in in Figure 18, faces can be non-convex and in certain cell-neighbor constellations may feature cavities.

For simplicity, all results are restricted to a unit-cube domain [YLL*09]. Nevertheless, other general domains do not conflict with our procedure as long as they are also compact and tessellated.
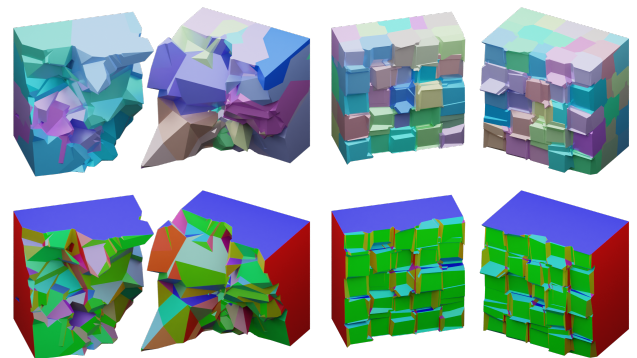

Figure 17: L_∞ *Voronoi results: Positions and orientations at random (left) vs. regular-jittered (right). The face-normal colored examples (bottom) reflect the uniformity of aligned orientations in contrast to the large diversity with the random orientations.*
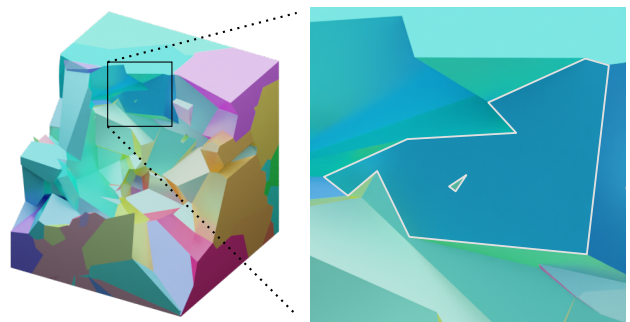

Figure 18: *Polygonal faces of a cell are always planar but not necessarily convex and may feature cavities, induced by other cells.*
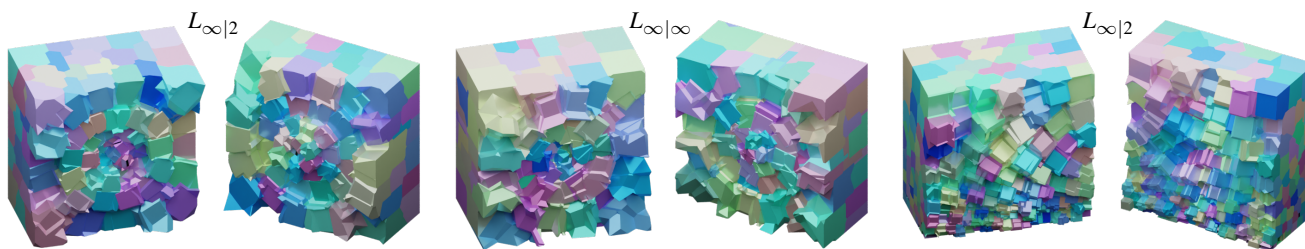
Figure 19: *Examples of anisotropic 3D CVTs. The left and center follow an orientation and weighting field as shown in the bottom row of Figure 21. The result on the right has an increasing cell size from bottom up and follows a smooth three-dimensional orientation field.*
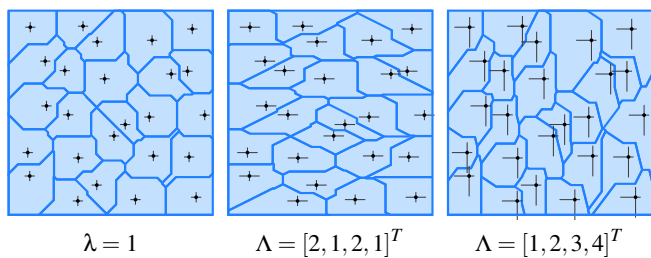

Figure 20: *Effects of weighted sites. Site positions are identical in all diagrams, the listed weight vectors are applied to all sites.*
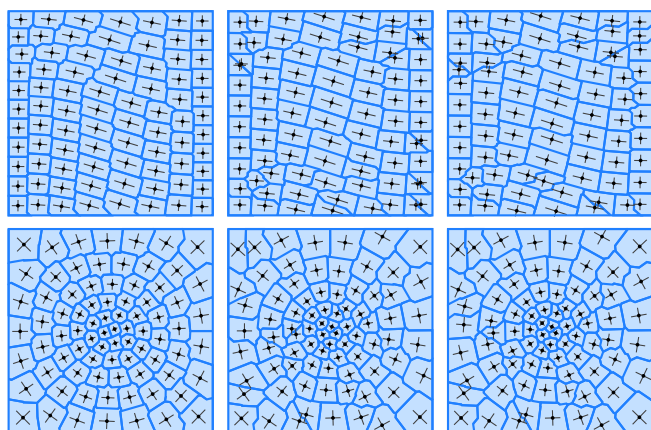

Figure 21: *2D examples of anisotropic weighted CVTs. Top: Wavy orientation field, increasing **x**-weights in the center. Bottom: Circular orientations and weights increase with distance to the center. The left column shows results relaxed with $L_{\infty|2}$, the center and right with $L_{\infty|\infty}$. In the right column site-merging was enabled.*

Figure 22: *Anisotropic 3D CVTs with different cell / centroid combinations. The isolated $L_{\infty|\infty}$ cells have largely overlapping bounding-boxes (bottom), thus very close centroids (Section 6.1).*

Anisotropy Equation (4) introduced the possibility to specify individual weights for the signed *max*-vectors of each site. The examples in Figure 20 compare the effects of different weight vectors on a given set of sites. Shown on the left is a default L_∞ diagram, where all site directions are equally weighted with $\lambda = 1$. The center diagram exemplifies anisotropically scaled sites, where only $\pm\mathbf{x}$ extents are scaled by 2. In the right example, the *max*-vectors are all scaled differently with $\Lambda = [1,2,3,4]^T$. For these examples the listed weight is applied to all sites in each diagram, respectively. Diagrams with individual site weights are exemplified with the CVTs in Figures 19, 21 and 22. Weight vectors can be specified analogously to an orientation field and queried for each site's position.
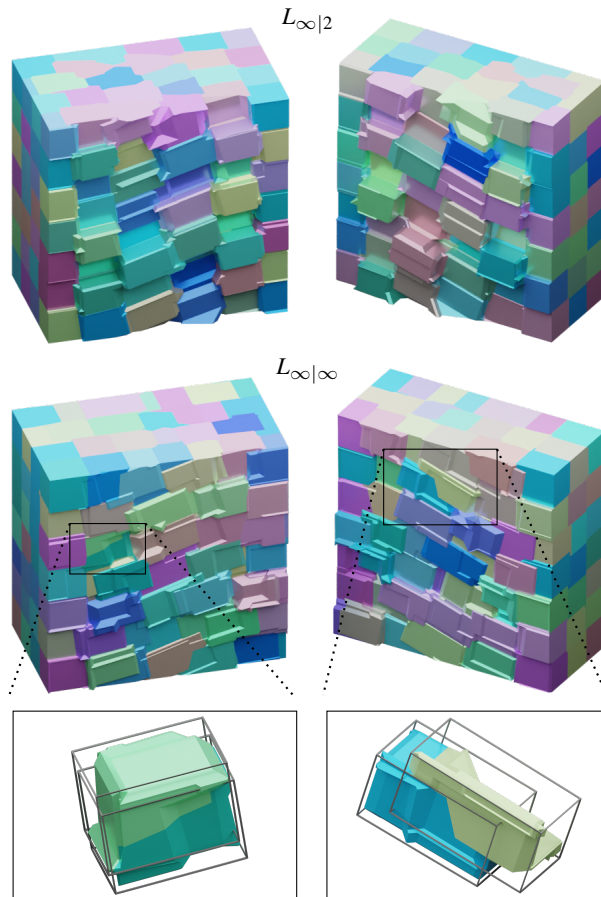
Lloyd CVTs The examples in Figure 21 list 2D CVT results of our generalized L_∞ Voronoi diagrams with two different orientation fields and anisotropic weighting. We can compare relaxed $L_{\infty|2}$ (left) with $L_{\infty|\infty}$ (center, right) examples. Site-merging (Section 6.1) was enabled for the examples on the right. As described before, this only locally resolves situations where close-by sites form triangular cells but the $L_{\infty|\infty}$ CVTs still contain non-regular constellations, i.e., local energy minima. The 3D CVT results in Figure 22 also compare different cell and centroid combinations with equivalent orientation fields and anisotropic weighting as in

the 2D examples. As pointed out with the isolated cells of the $L_{\infty|\infty}$ CVT, close-by sites with largely overlapping bounding-boxes are also valid constellations in 3D diagrams. More examples of anisotropic weighted CVTs are shown in Figure 19. Again, with L_2 centroids the CVT results feature more prominently box-shaped cells than with L_∞ centroids where cells also are at a relaxed state in a rather irregular arrangement.
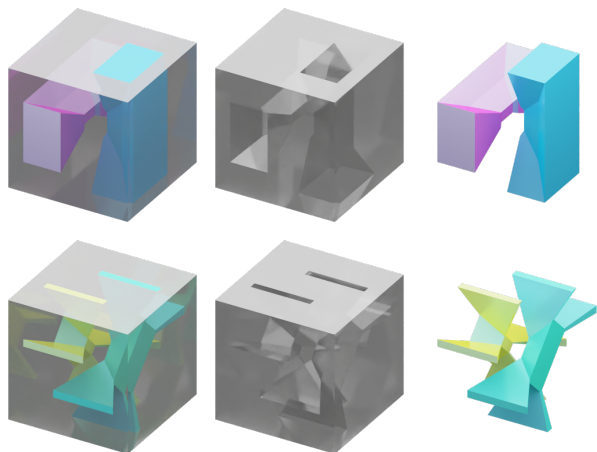

Figure 23: *Anisotropic weighting vectors can provoke cells with a genus larger than 0 (gray). In the bottom row, the two colored cells are additionally rotated, creating even more complex structures.*

Genus As exemplified in 2D in Figure 4, unequal site weights can lead to scenarios of one cell fully enclosing another cell. While this translates analogously to 3D, the three-dimensional space further provides another degree of freedom: In Figure 23, three sites A, B and C are located on the **x** axis at positions $x = \{-0.25, 0, 0.25\}$, respectively. The weighting is set uniformly for B with $\lambda_B = 2$ and with anisotropic vectors for the outer cells as $\Lambda_A = [1, 2, 1, 1, 2, 1]^T$ and $\Lambda_C = [1, 1, 2, 1, 1, 2]^T$. Between A and B, and B and C, the equilibrium of weights, or respectively the lack of it, in certain dimensions leads to cells A and C tunneling through the cell of B. The 45° **x**-axis rotation of A and C in the bottom row example leads to another interesting scenario as they now transcend 5 of B's *max*-dimension regions each. For this visualization, cells are limited by the unit-cube domain. This does not invalidate the example: With six additional sites, equally weighted as B and located on the $\pm\{\mathbf{x}, \mathbf{y}, \mathbf{z}\}$ axes with a distance of 2, the outcome of B's cell with a genus larger than 0 would be identical.

Precision The extraction of cells and their polygonal topology is based on robust integer labels, thus invariant to numeric issues. As mentioned in Section 5.6, geometry is cut only virtually: Recursive intersection with multiple planes would accumulate 3D snap rounding errors [DLL18] due to limited precision in the representation of floating point numbers. Instead, each vertex can be inferred from the intersection of three planes, which allows for very tiny details, as highlighted in Figure 24. This principle also naturally applies for the initialization of the pyramid cube in Figure 2, analogously represented as six proper oriented planes. To approach possible downstream applications, the whole meshed domain can be set to an appropriate scale for export.
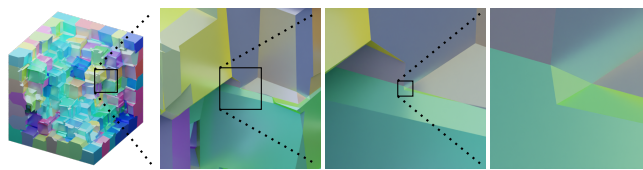

Figure 24: *Zoom-in on a microscopic detail in the cell geometry.*

6.4. Conclusion

In this work we present an algorithm for constructing meshed Voronoi diagrams using a generalized form of the L_∞ norm. Common methods generalize for two, three or more dimensions but are mostly only considered in Euclidean space and the L_2 norm. With the L_∞ norm, the construction of Voronoi diagrams becomes an indefinitely harder problem as bisectors are no longer planar, adjacency also depends on site orientation and cells are no longer only convex, can be of a genus larger than 0 or decompose into separated regions. Previous concepts for the construction of L_∞ Voronoi diagrams are bound to 2D space and assume uniformly oriented and equally weighted sites. Our concept is focused on the construction of cells, shaped by their immediate neighborhood. We determine the non-trivial bisectors, i.e., the meshed cell geometry as portions of hyperplanes, unambiguously defined by pairs of prevalent *max*-dimensions from two neighboring sites. In our cut & clip algorithm, the common Voronoi criterion eventually determines the final geometry of each cell. While the general definitions of our concept theoretically translate to higher dimensions we present and discuss results for 2D and 3D diagrams. The definition for bisector planes is extended with site-individual orientation matrices and the possibility for anisotropic weight vectors. This fully generalizes the concept of common Voronoi diagrams to weighted, anisotropic L_∞ Voronoi diagrams. When employed in Lloyd relaxations to compute CVTs, we found that true L_∞ diagrams with L_∞ centroids do not necessarily converge on only square- or cube-shaped cells. Sites are in a relaxed state while remaining close or even collapse to a single point as their bounding-boxes can largely overlap. This novel insight contrasts many applications that optimize for CVTs with L_∞ approximations. However, we also experimented with hybrid Lloyd relaxations combining L_2 centroids and our L_∞ diagrams. Hybrid relaxations also converged and resulted in CVTs with the anticipated regular grid-like structures. Future improvements, possibly with hardware parallelization, will make this a valuable asset for CVT generation and downstream meshing applications. To the best of our knowledge, the presented algorithm is a first of its kind to compute generalized L_∞ Voronoi diagrams in 2D and 3D.

References

[ADVDI05] ALLIEZ P., DE VERDIÈRE É. C., DEVILLERS O., ISENBURG M.: Centroidal Voronoi diagrams for isotropic surface remeshing. *Graphical models 67*, 3 (2005), 204–231.

[AKL13] AURENHAMMER F., KLEIN R., LEE D.-T.: *Voronoi diagrams and Delaunay triangulations.* World Scientific Publishing Company, 2013.

[BAR*21] BASSELIN J., ALONSO L., RAY N., SOKOLOV D., LEFEB-VRE S., LÉVY B.: Restricted power diagrams on the GPU. In *Computer Graphics Forum* (2021), vol. 40.1, Wiley Online Library, pp. 1–12.

[BDH96] BARBER C. B., DOBKIN D. P., HUHDANPAA H.: The quick-hull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS) 22*, 4 (1996), 469–483.

[BNN10] BOISSONNAT J.-D., NIELSEN F., NOCK R.: Bregman Voronoi Diagrams. *Discrete & Computational Geometry 44*, 2 (2010), 281–307.

[Bow81] BOWYER A.: Computing Dirichlet tessellations. *The computer journal 24*, 2 (1981), 162–166.

[BTKA10] BOCK M., TYAGI A. K., KREFT J.-U., ALT W.: Generalized Voronoi tessellation as a model of two-dimensional cell tissue dynamics. *Bulletin of mathematical biology 72*, 7 (2010), 1696–1731.

[BTL22] BUKENBERGER D. R., TARINI M., LENSCH H. P. A.: At-Most-Hexa Meshes. *Computer Graphics Forum 41*, 1 (2022), 7–28.

[CGAL22] COMPUTATIONAL, GEOMETRY, ALGORITHMS, LIBRARY: *CGAL User and Reference Manual*, 5.4 ed. CGAL Editorial Board, 2022.

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. In *ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), SIGGRAPH '04, Association for Computing Machinery, p. 905–914.

[CW10] CHANG H.-C., WANG L.-C.: A Simple Proof of Thue's Theorem on Circle Packing. *arXiv preprint arXiv:1009.4322* (2010).

[DEJ06] DU Q., EMELIANENKO M., JU L.: Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations. *SIAM journal on numerical analysis 44*, 1 (2006), 102–119.

[Dey15] DEY S. K.: *Voronoi diagrams in the max-norm: algorithms, implementation, and applications*. PhD thesis, Università della Svizzera italiana, 2015.

[DLL18] DEVILLERS O., LAZARD S., LENHART W.: *3D Snap Rounding*. Research Report RR-9149, Inria Nancy - Grand Est, Feb. 2018.

[Dwy91] DWYER R. A.: Higher-dimensional Voronoi diagrams in linear expected time. *Discrete & Computational Geometry 6*, 3 (1991), 343–367.

[EH19] EDER G., HELD M.: Weighted Voronoi diagrams in the maximum norm. *International Journal of Computational Geometry & Applications 29*, 03 (2019), 239–250.

[For87] FORTUNE S.: A sweepline algorithm for Voronoi diagrams. *Algorithmica 2*, 1 (1987), 153–174.

[GMJ87] GORDON B., MCCAWLEY JR J.: *Challenging mathematical problems with elementary solutions*, vol. 1. Courier Corporation, 1987. Originally published in 1954 by Akiva M. Yaglom and Isaak M. Yaglom.

[GPR02] GANOVELLI F., PONCHIO F., ROCCHINI C.: Fast tetrahedron-tetrahedron overlap algorithm. *Journal of Graphics Tools 7*, 2 (2002), 17–25.

[Han66] HANAN M.: On Steiner's problem with rectilinear distance. *SIAM Journal on Applied mathematics 14*, 2 (1966), 255–265.

[Hau01] HAUSNER A.: Simulating decorative mosaics. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM, pp. 573–580.

[HIKL*99] HOFF III K. E., KEYSER J., LIN M., MANOCHA D., CULVER T.: Fast computation of generalized Voronoi diagrams using graphics hardware. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 277–286.

[LL10] LÉVY B., LIU Y.: L_p Centroidal Voronoi Tessellation and Its Applications. In *ACM SIGGRAPH 2010 Papers* (New York, NY, USA, 2010), SIGGRAPH '10, Association for Computing Machinery.

[Llo82] LLOYD S.: Least squares quantization in PCM. *IEEE transactions on information theory 28*, 2 (1982), 129–137.

[LPL11] LIU C.-H., PAPADOPOULOU E., LEE D.-T.: An output-sensitive approach for the L_1 / L_∞ k-nearest-neighbor Voronoi diagram. In *European Symposium on Algorithms* (2011), Springer, pp. 70–81.

[LWL*09] LIU Y., WANG W., LÉVY B., SUN F., YAN D.-M., LU L., YANG C.: On Centroidal Voronoi Tessellation—Energy Smoothness and Fast Computation. *ACM Trans. Graph. 28*, 4 (sep 2009).

[Ma00] MA L.: *Bisectors and Voronoi Diagrams for Convex Distance Functions*. PhD thesis, FernUniversität Hagen, 2000.

[MB12] MOUTON T., BÉCHET E.: Lloyd relaxation using analytical Voronoi diagram in the L_∞ norm and its application to quad optimization. *Proceedings of the 21st International Meshing Roundtable* (2012).

[Now15] NOWAK A.: Application of Voronoi diagrams in contemporary architecture and town planning. *Challenges of Modern Technology 6*, 2 (2015).

[PL01] PAPADOPOULOU E., LEE D.: The L_∞ Voronoi diagram of segments and VLSI applications. *International Journal of Computational Geometry & Applications 11*, 5 (2001), 503–528.

[PS85] PREPARATA F. P., SHAMOS M. I.: *Computational geometry: an introduction*. Springer Science & Business Media, 1985.

[Reb93] REBAY S.: Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer-Watson algorithm. *Journal of computational physics 106*, 1 (1993), 125–138.

[RSLL18] RAY N., SOKOLOV D., LEFEBVRE S., LÉVY B.: Meshless Voronoi on the GPU. *ACM Trans. Graph. 37*, 6 (dec 2018).

[Sci20] SCIPY 1.0 CONTRIBUTORS: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods 17* (2020), 261–272.

[VKK*03] VARADHAN G., KRISHNAN S., KIM Y. J., DIGGAVI S., MANOCHA D.: Efficient max-norm distance computation and reliable voxelization. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (Goslar, DEU, 2003), SGP '03, Eurographics Association, p. 116–126.

[Wat81] WATSON D. F.: Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. *The computer journal 24*, 2 (1981), 167–172.

[WTLY13] WU H.-Y., TAKAHASHI S., LIN C.-C., YEN H.-C.: Voronoi-based label placement for metro maps. In *International Conference on Information Visualisation* (2013), IEEE, pp. 96–101.

[YLL*09] YAN D.-M., LÉVY B., LIU Y., SUN F., WANG W.: Isotropic Remeshing with Fast and Exact Computation of Restricted Voronoi Diagram. In *Computer Graphics Forum* (2009), vol. 28.5, pp. 1445–1454.

Appendix A: Conceptual Validity

This appendix aims to substantiate heuristics and algorithms utilized in our procedure, proving their practical applicability.

Pyramid Initialization

Constructing a Voronoi diagram for a given domain is the segmentation of space into uniquely identifiable cells. Our cells emerge from resolving overlaps with other cells and, by design, only reduce in size during the diagram generation. Therefore, a main objective for the initialization is that all points of the domain are already contained within at least one cell.

Portions of cells reaching over the domain boundaries are simply clipped. The cut & clip section of our algorithm determines, which parts of a cell-cell-overlap actually are associated with at cell. As described in Section 6.2, cells scaled to extent over the full domain
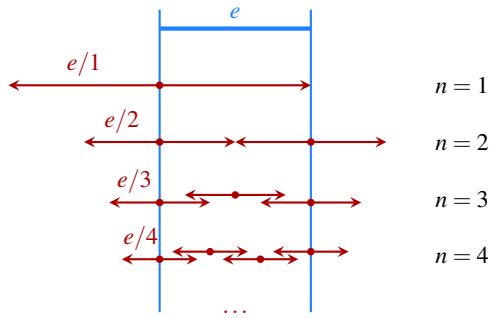
Figure 25: *A 1D example to illustrate the pyramid heights scale based on the domain extent e and number of sites n. With $h = \frac{e}{n}$ all points of the domain are covered by at least one cell.*

Figure 26: *Native initialization heuristics (left, center) would fail to produce the result (right). In practice, a multiplicative safety-radius factor 2 allowed to robustly resolve any random input.*

would trivially satisfy the objective of all domain points being contained in cells but would also cause every cell to overlap with all others. The number of overlaps to be resolved would then grow in $\mathcal{O}(n^2)$. Following heuristics aim to optimize the initialization and reduce the amount of overlaps, based on the observation that these are only of relevance for the local neighborhood around a cell.

Figure 25 illustrates a 1D example for our heuristic to scale pyramid heights on a domain with extent e inversely to the site population density $\delta = \frac{n}{e}$, where n is the number of sites (per dimension). Under the assumption of a perfect regular site positioning, i.e., the maximum distance between each other, one can trivially comprehend: The height of a pyramid (equal to the extent of a cell in every dimension) can be set as $h = \frac{1}{\delta} = \frac{e}{n}$ for every point of the domain to be contained in at least one cell.

Without rotation, this would also hold for the 2D and 3D case as cells initialize with a hypercube. However, with orientations one also need to cover the corner points of the squared or cubed domain. Therefore, h is to be scaled with $\sqrt{d}$ for d dimensions, which corresponds to the maximum distance between any two points in the domain, i.e., the diagonal's length. This formulation of h, of course, only applies if a rather regular site distribution can be assumed. For jittered positions, h needs to be scaled by a factor dependent on the jitter amplitude such that cells can still be guaranteed to overlap.

When there is no assumption to be made on the given site distribution, another fallback solution could be to query the local neighborhood of a site. If h is set to the *max*-distance of a closest neighbor site, it can be guaranteed for both cells to overlap. This has to be determined individually for the individual signed *max*-dimensions and either scale each *max*-pyramid individually or chose h as the maximum over all minimal neighbor distances. If there is no nearest neighbor for a certain signed direction, the cell needs to extend at least up to the domain boundary such that again all points within the domain are covered.

Limitations These heuristics only describe bland concepts and need to be adapted to individual application scenarios. For example in Figure 26 sites are arranged to particularly provoke failures of our initialization schemes. In the left image, each cell is scaled to extent to its *closest neighbor* site but still not all parts of the domain are covered by a cell. In the center, the blue cell extends to the *farthest* neighbor cell over all dimensions and the green cell
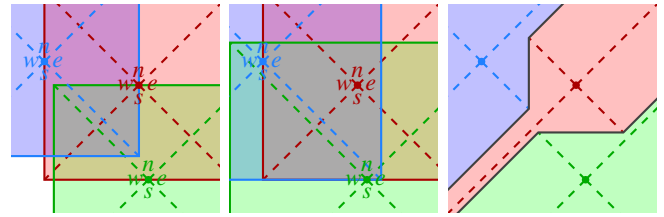
has no neighbor in w direction so it extends to the domain boundary. Now the whole domain is covered by at least one cell but this initialization is still not sufficient to create the correct result (right).

As described before, a $h = e$ scaling covers all space with all cells, thus guarantees to produce a correct diagram but in quadratic time. The heuristics aim to reduce computation costs by limiting cell overlaps to a local neighborhood but potentially sacrificing robustness. In practice, we utilized the proposed methods to derive base scales for individual cells but also applied a safety-radius with a factor of 2. With CVTs however, this factor can also be lowered to be < 1, as sites align in a very regular arrangement.

Cut & Clip Algorithm

In the following we argue why the cut & clip algorithm is a valid procedure to construct L_∞ diagrams. Under the assumption of a proper initialization it is guaranteed that the intersection of neighboring site's base meshes is not empty. A bisector hyperplane P_{AB} between sites A and B specifies a halfspace, separating points closer to A from points closer to B. This analogously holds for the bisectors P_{AC} and P_{BC} introduces by a third site C. Intersecting the bisectors, i.e., superimposing the halfspaces does not corrupt their significance for the individual sites. Bisectors are only relevant for associated sites, e.g., A's cell is not affected by the bisector P_{BC}.

The init. meshes consist of triangles (2D) or quad-based pyramids (3D), i.e., convex primitives. Splitting them with a straight cut (line or plane) will again result in convex primitives. **I.** With this guarantee to be convex, primitive's centroids will never lie outside but always on their inside. **II.** By construction, no primitive transcends a bisector, i.e., has parts on both sides of any halfspace.

With these two observations we can safely formulate criteria for a convex fragment, by only considering its centroid. I.e., we can determine on which side of a bisector a fragment lies be evaluating the Voronoi criterion for its centroid. If the centroid of a fragment is closer to a site A than to any other, then it is part of A's cell. If there is another site B, closer to the fragment centroid than site A, not a single point within the fragment can be part of A's cell.

At this point it is crucial that this evaluation is strictly limited to the scope of one site: Rejected fragments are not automatically assigned to another, closer, site's cell. This cell-individual evaluation conveniently also lends itself for parallelism.