Algorithms and Criteria for Volumetric Centroidal Voronoi Tessellations

Li Wang

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : Mathématiques et Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Li WANG

Thèse dirigée par **Edmond BOYER**, INRIA

préparée au sein du **Laboratoire Laboratoire Jean Kuntzmann** dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Algorithmes et Critères pour les Tessellations Volumétriques de Voronoi Centroïdales

Algorithms and Criteria for Volumetric Centroidal Voronoi Tessellations

Thèse soutenue publiquement le **27 janvier 2017**,
devant le jury composé de :

Monsieur EDMOND BOYER
DIRECTEUR DE RECHERCHE, INRIA DELEGATION ALPES, Directeur de thèse
Monsieur FRANCK HETROY-WHEELER
MAITRE DE CONFERENCES, GRENOBLE INP, Examinateur
Monsieur PIERRE ALLIEZ
DIRECTEUR DE RECHERCHE, INRIA CENTRE S. ANTIPOLIS - MEDITERRANEE, Rapporteur
Monsieur BRUNO LEVY
DIRECTEUR DE RECHERCHE, INRIA CENTRE NANCY GRAND EST, Rapporteur
Madame STEFANIE HAHMANN
PROFESSEUR, GRENOBLE INP, Président
Monsieur SEBASTIEN VALETTE
CHARGE DE RECHERCHE, CNRS DELEGATION RHONE AUVERGNE, Examinateur

Abstract

This thesis addresses the problem of computing volumetric tessellations of three-dimensional shapes, *i.e.*, given a three-dimensional shape that is usually represented by its boundary surface, how to optimally subdivide the interior of the surface into smaller shapes, called *cells*, according to several criteria related to accuracy, uniformity and regularity. We consider centroidal Voronoi tessellations, which are uniform and regular volumetric tessellations.

A *centroidal Voronoi tessellation (CVT)* of a shape can be viewed as an optimal subdivision in the sense that the cells' centers of mass, called *centroids*, are regularly distributed inside the shape. CVTs have been used in computer vision and graphics because of their properties of uniformity and regularity that are immune to shape variations. However, problems such as how to evaluate the regularity of a CVT and how to build a CVT from different representations of shapes remain.

As one contribution of this thesis, we propose regularity criteria based on the normalized second order moments of the cells. These regularity criteria allow evaluating volumetric tessellations and specially comparing the regularity of different Tessellations without the assumption that their shape and number of sites should be the same. Meanwhile, we propose a hierarchical approach based on a subdivision scheme that preserves cell regularity and the local optimality of CVTs. Experimental results show that our approach performs more efficiently and builds more regular CVTs according to the regularity criteria than state-of-the-art methods.

Another contribution is a novel CVT algorithm for implicit shapes and an extensive comparison between the Marching Cubes, the Delaunay refinement technique and our algorithm. The keys of our algorithm are to use convex hulls and local improvements to build accurate boundary cells. We present a comparison of these three algorithms with different criteria including accuracy, regularity and complexity on a large number of different data. The results show that our algorithm builds more accurate and regular volumetric tessellations than the other approaches.

We also explore applications such as a shape animation framework based on CVTs that generates plausible animations with real dynamics.

Keywords. Volumetric tessellation • Centroidal Voronoi tessellation • Regularity

Résumé

Cette thèse traite du problème du calcul d'une tessellation volumique d'une forme tridimensionnelle, c'est-à-dire, étant donnée une forme tridimensionnelle qui est habituellement représentée par sa surface frontière, comment subdiviser de manière optimale l'intérieur de la surface en formes plus petites, appelées *cellules*, selon plusieurs critères concernant la précision, l'uniformité et la régularité. Nous considérons les tessellations de Voronoï centroidales qui sont des tessellations volumiques uniformes et régulières.

Une *tessellation de Voronoi centroïdale (CVT)* d'une forme peut être considérée comme une subdivision optimale au sens où les centres de masse, appelés *centroides* des cellules, sont répartis de manière optimale à l'intérieur de la forme. CVTs ont été utilisés en vision par ordinateur et en infographie en raison de leurs propriétés d'uniformité et de régularité qui sont indépendantes des variations de la forme. Cependant, des problèmes restent ouverts, comme l'évaluation de la régularité d'une CVT ou la construction d'une CVT à partir de formes représentées de diférentes manières.

Une contribution de cette thèse est que nous proposons des critères de régularité basées sur les moments de second ordre normalisés des cellules. Ces critères de régularité permettent d'évaluer les tessellations volumiques, et surtout de comparer la régularité des différentes Tessellations sans l'hypothèse que leur forme et leur nombre de sites devraient être les mêmes. Nous proposons également une approche hiérarchique basée sur un schéma de subdivision qui préserve la régularité des cellules et l'optimalité locale des CVTs. Les résultats expérimentaux montrent que notre approche est plus efficace et construit des CVTs plus régulières que les méthodes de l'état de l'art selon les critères de régularité.

Une autre contribution est une nouvelle algorithme de calcul de CVT pour les formes implicites et une comparaison approfondie entre le Marching Cubes, la technique du raffinement de Delaunay et notre algorithme. La clé de notre algorithme est d'utiliser des enveloppes convexes et une amélioration locale pour construire des cellules au bord avec précision. Nous présentons une comparaison des trois algorithmes avec des critères différents, comme la précision, la régularité et la complexité sur un grand nombre de données différentes. Les résultats montrent que notre méthode construit les tessellations volumiques les plus précises et les plus régulières.

Nous explorons aussi des applications comme, par exemple, une chaîne de traitement d'animation des formes basées sur les CVTs qui génère des animations plausibles à partir de dynamique réelle.

Mots-clés. Tessellation volumique • Tessellation de Voronoi centroidale • Régularité

Contents

List of Figures

List of Tables

Symbols and Abbreviations

$\mathbb{R}$	real coordinate space
$\mathcal{M}$	triangle mesh
$\mathcal{S}$	shape surface
$\mathcal{V}$	shape volume
$\mathcal{X}$	point set
Ω	three-dimensional domain
Ω_i	cell of a volumetric tessellation, i corresponds to the index of the i-th site if the tessellation is Voronoi
$\overline{G}$	average value of a set of dimensionless second moments
π_i^w	power function for the i-th site of a power diagram with respect to the weight w_i
$\widetilde{G}$	median value of a set of dimensionless second moments
B_{ij}	bisector of the i-th and the j-th Voronoi cells
B_{ij}^w	bisector of the i-th and the j-th additively weighted Voronoi cells
d_i	Euclidean distance function for the i-th site of a Voronoi tessellation
d_i^g	general distance function for the i-th site of Voronoi tessellation
d_i^w	weighted distance function for the i-th site of an additively weighted Voronoi tessellation with respect to the weight w_i
E	CVT energy function
f	scalar function
$G(P)$	dimensionless second moment of a polytope P
G_2	lower bound of dimensionless second moment in two dimensions

G_3	lower bound of dimensionless second moment in three dimensions
G_{max}	max value of a set of dimensionless second moments
G_{rmse}	root-mean-square error of a set of dimensionless second moments
P	polytope
x_i	i-th site of a Voronoi tessellation
CDT	constrained Delaunay triangulation
CVT	centroidal Voronoi tessellation
MC	Marching Cubes
RDT	restricted Delaunay triangulation

Chapter 1

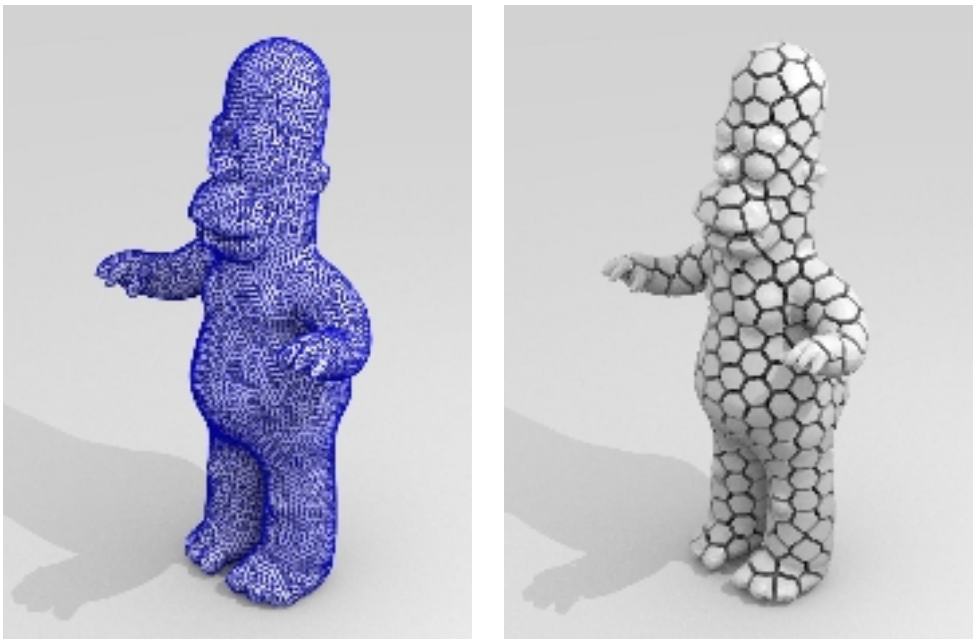
Introduction

Contents

1.1 Context

A shape is usually represented by its boundary surface and is largely used in computer vision and graphics. However, the information offered by a shape surface is not always sufficient for several applications [Raviv et al., 2010] [Allain et al., 2016]. In order to cover more information, volumetric data that represents the interior of a shape surface, *i.e.* the *shape volume*, is used. The aim of *a volumetric tessellation* is to build a volumetric quantization of a shape volume by filling it with small cells as shown in Figure 1.1. A *good* volumetric tessellation [Wang et al., 2016b] consists of uniform and regular cells with a good approximation to the boundary surface. The analysis of the goodness of a volumetric tessellation and the construction of a good volumetric tessellation remain a challenge.

A volumetric tessellation can be computed using many approaches that can be roughly divided into two categories according to the algorithm they use: Eulerian approaches and Lagrangian approaches. Eulerian approaches consider a grid discretizing the observation domain that contains the shape. The grid is usually fixed and composed of the same sized cells that are identified inside or outside the shape. The volumetric tessellation can be further obtained by computing the intersection between the grid and the shape. These approaches are fast. However, a large waste of memory occurs when the observation domain is relatively large



(a) (b)

Figure 1.1 – (a): A surface. (b): A volumetric tessellation.

compared to the shape and irregular cells are often generated on the boundary. Instead of discretizing the observation domain containing the shape, Lagrangian approaches discretize directly the shape that reduces the complexity. These approaches usually include an optimization phase that provides the goodness control of the generated volumetric tessellations.

Centroidal Voronoi tessellation (CVT) is a Lagrangian approach that is widely used in many applications due to its good properties. An example is shown in Figure 1.2. The CVT algorithm distributes points, called *sites*, inside the shape and computes the intersection between the Voronoi tessellation of the sites and the shape. Then the site positions are updated by minimizing the so-called CVT energy function that measures the quantification error. The final volumetric tessellation consists of the cells centered around the sites. A CVT corresponds to a local minimum of its energy function. However, finding the optimal CVT that corresponds to the global minimum of its energy function remains a challenge since the energy function is usually non-linear and non-convex.

This thesis focuses on the construction of CVTs and their analysis including an evaluation and comparison with other existing approaches.

Figure 1.2 – An animation based on tracking and morphing on CVTs.

It attempts to answer the following questions: 1. How can the regularity be defined for a volumetric tessellation ? 2. How to generate a CVT as close to the optimal as possible ? 3. How to compare different approaches to compute volumetric tessellations ? 4. What are possible new applications of regular volumetric tessellations ? The answers are detailed in Chapters 3, 4, 5 and 6, respectively.

1.2 Contributions

The main contributions of this thesis are detailed into Chapters 3, 4, 5 and 6. Before entering these chapters, the state of the art for volumetric tessellation approaches is reviewed in Chapter 2. Finally, the conclusions and future work are presented in Chapter 7.

Chapter 3

In this chapter, we build on a theoretical work of Conway and Sloane [1982] and propose regularity criteria for volumetric tessellations. These criteria are based on the normalized second order moments of polyhedra. We show that the regularity criteria are linked to the CVT energy function but are dimensionless and therefore enable global evaluations as well as comparisons. We also propose an application of one of these criteria by considering it as the stopping criterion for CVT computation. This work has been published in [Wang et al., 2016a];

Chapter 4

In this chapter, we propose a hierarchical approach that provides CVTs with more regularity than state-of-the-art methods. Our strategy is based

on a subdivision scheme that preserves cell regularity and the local optimality of CVTs on unbounded domains. This scheme tends to propagate cell regularity through hierarchy levels when applied to bounded domains. We demonstrate the efficiency of this framework with an in-depth evaluation that includes sensitivity analysis, comparisons with previous work and analyses of the convergence speed and computation time. This work has been published in [Wang et al., 2016a];

Chapter 5

In this chapter, we propose a novel approach that builds CVTs from implicit forms. These tessellations provide volumetric and surface representations with strong regularities in addition to provably more accurate approximations of the implicit forms considered. In order to compare with other existing approaches, we present an extensive evaluation that analyzes various properties of the main approaches for implicit to explicit volumetric tessellations: Marching Cubes, Delaunay refinement and CVTs, including accuracy and shape quality of the resulting shape mesh. This work has been published in [Wang et al., 2016b].

Chapter 6

In this chapter, we propose a novel polyhedra clipping algorithm to compute clipped Voronoi tessellations. This algorithm reduces the three-dimensional clipping problem to a two-dimensional triangle-triangle intersection problem. We demonstrate the efficiency and robustness of our algorithm with a wild range of experiences. This work has been published in [Wang et al., 2016a]. We also propose a novel system for animation generation. This system first generates CVTs from a stream of three-dimensional observations acquired with a video-based capture system, then produces animation by combining video-based shape motion and mechanical effects on the generated CVTs. This work has been made available in [Allain et al., 2016].

List of Publications

- Li Wang, Franck Hétroy-Wheeler, Edmond Boyer. A Hierarchical Approach for Regular Centroidal Voronoi Tessellations. In *Computer Graphics Forum 2016* (Vol. 35, No. 1, pp. 152-165).
- Li Wang, Franck Hétroy-Wheeler, Edmond Boyer. On Volumetric Shape Reconstruction from Implicit Forms. In *ECCV 2016 - European*

Conference on Computer Vision, Oct 2016, Amsterdam, The Netherlands.

- Benjamin Allain, Li Wang, Jean-Sébastien Franco, Franck Hétroy-Wheeler, Edmond Boyer. Shape Animation with Combined Captured and Simulated Dynamics. In *arXiv:1601.01232*, ArXiv. 2016.

Chapter 2

Volumetric Tessellation

Contents

2.1 Introduction

A *volumetric tessellation* of a shape volume $\mathcal{V}$ consists of one or more smaller shapes that fill $\mathcal{V}$ with no overlap and no gap. It is also called tilling, partition or subdivision of $\mathcal{V}$. These smaller shapes of the volumetric tessellation are called *cells* and noted the $\{\Omega_i\}$. A shape is usually represented by its boundary that is a surface $\mathcal{S}$. It has to be mentioned that volumetric tessellations are different from surface tessellations that are restricted to $\mathcal{S}$. Volumetric tessellations can be generalized to any dimensions. In this thesis, we focus on tessellations in three dimensions.

Over the last decade, the construction of volumetric tessellations has been largely studied and used for applications in both computer vision and computer graphics. Many methods have been proposed for different applications. Depending on the input, the methods can be roughly divided into two categories. When the input is an implicit representation that identifies a shape volume $\mathcal{V}$ as being a region within an observation domain Ω, the output tessellation is an approximation of $\mathcal{V}$ [Lorensen and Cline, 1987] [Jamin et al., 2015]. Such implicit representation is typically given as a scalar function $f : \Omega \to \mathbb{R}$ that takes different values inside

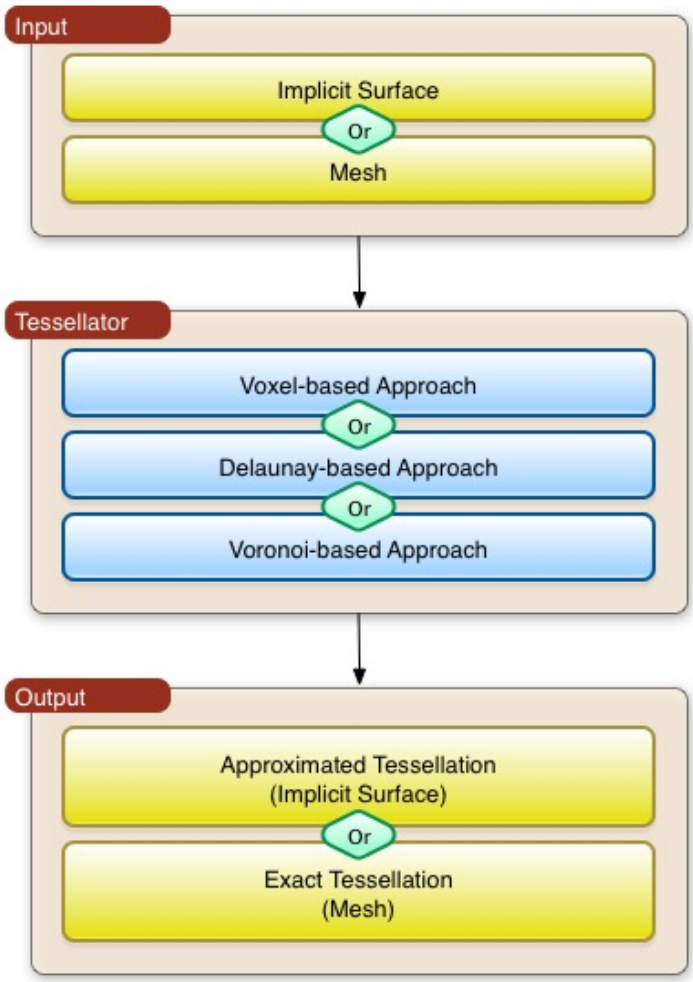
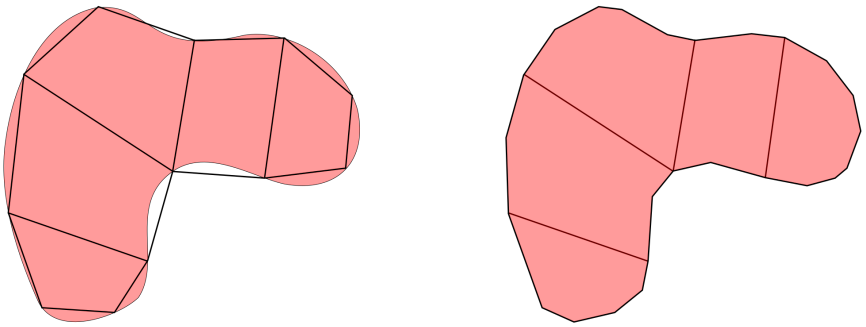


Figure 2.1 – An overview of volumetric tessellations.

and outside $\mathcal{V}$, for instance an implicit function, an indicator function or a distance function. This implicit representation can be characterized by an implicit surface that is usually obtained for a real object through surface reconstruction methods from point clouds [Berger et al., 2014]. The conversion from implicit surfaces to explicit volumetric tessellations is also called polyhedrization or mesh generation [Jamin et al., 2015]. When the input is an explicit representation, usually a polygonal mesh $\mathcal{M}$, the output tessellation is exact with its associated polygonal surface that is identical to $\mathcal{M}$ [Yan et al., 2013] [Si, 2015]. Without specification, $\mathcal{M}$ refers to a triangle mesh in this thesis. Figure 2.2 gives an illustration of the difference between the two cases in two dimensions.

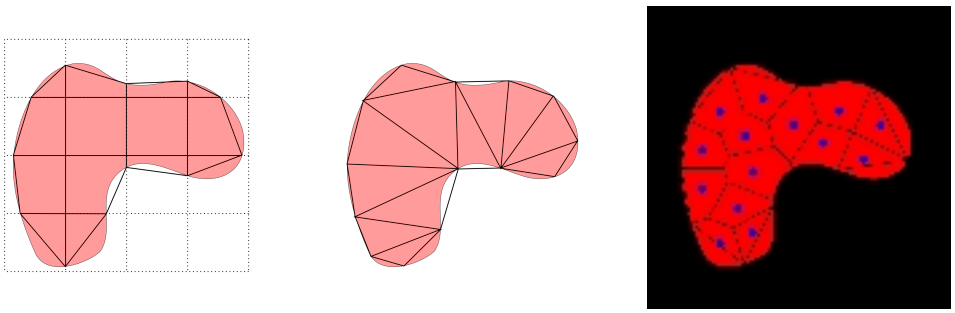


(a) (b)

Figure 2.2 – Two-dimensional volumetric tessellations for different input shape (in red). (a): Approximated case when the input is an implicit surface. (b): Exact case when the input is a mesh.

Depending on the algorithms they used, the methods can be roughly categorized into voxel-based, Delaunay-based and Voronoi-based approaches. Voxel-based approaches include the Marching Cubes method [Lorensen and Cline, 1987] and its extensions. They use a fixed grid that discretizes the observation domain Ω containing the shape volume $\mathcal{V}$ into cells that are usually identical such as cubes or tetrahedra for instance. Then the intersection between $\mathcal{V}$ and the cells is computed. Delaunay-based approaches, such as [Jamin et al., 2015] and [Si, 2015], build a three-dimensional Delaunay triangulation, also called tetrahedralization, with the sampling points on the shape surface $\mathcal{S}$. Voronoi-based approaches compute the intersection between $\mathcal{V}$ and the Voronoi diagram, dual of the Delaunay triangulation, of points $\{x_i\}$, called sites, that are sampled inside V. Figure 2.3 visualizes the difference between these three approaches with the same input in two dimensions.

Figure 2.1 shows the overview of the volumetric tessellation process. Following this overview, Voxel-based, Delaunay-based and Voronoi-based approaches are reviewed in details in Section 2.2, 2.3 and 2.4, respectively.



(a) (b) (c)

Figure 2.3 – Two-dimensional volumetric tessellations with different approaches for the same input shape (in red). (a): Voxel-based. (b): Delaunay-based. (c) Voronoi-based, with its associated sites in blue.

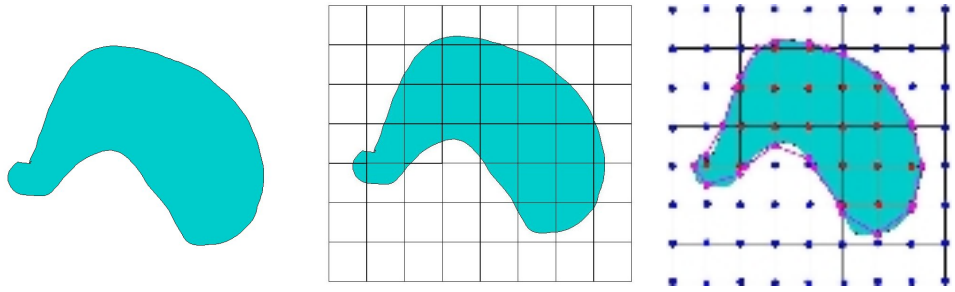
2.2 Voxel-based Approaches

Voxel-based approaches for implicit surfaces include the well-known Marching Cubes (MC) algorithm and its extensions. It has to be mentioned that the MC has originally been proposed for the isosurface extraction problem. Most of the improvements on the MC are designed for requirements at the surface level, for instance surface simplification. However, since the essential idea of the MC is to use a volumetric grid to discretize the domain that contains the input shape, it belongs to the set of volumetric tessellation approaches. Besides, the volumetric output generated by the MC have been used in many applications. For example, Wu et al. [2015] considered the volumetric output as the input for deep learning of shape recognition, and Raviv et al. [2010] used it to find volumetric heat kernel signatures of shapes.

The process of voxel-based approaches consists of two main steps as follows:

1. Voxelization: use a grid to discretize the domain containing the input shape.
2. Clipping: compute the intersection between the grid and the input shape.

Figure 2.4 visualizes these steps using an example in two dimensions. The grid used for voxelization is usually fixed and constructed with cubes sharing the same size. However, many extensions using grids with cells of different shapes, such as multi-resolution or tetrahedra, have been designed for the purpose of complexity simplification or disambiguation.



(a) (b) (c)

Figure 2.4 – Overview of voxel-based approaches in two dimensions. (a): Input shape. (b): Voxelization of the domain containing the shape with cubic cells. (c) Approximated clipping in pink. The images are from [Anderson].

These extensions are reviewed in Section 2.2.1. When the input shape is a mesh, the clipping step boils down to a polyhedron intersection problem between the mesh and the cells of the grid. Otherwise, it is the typical approximated clipping of the original MC algorithm. Voxel-based approaches are widely used because of their time efficiency. However, the generated cells in the boundary can be very irregular and non-uniform. We detail the clipping algorithm that uses cubic grid since it is more interesting in terms of regularity than other grids in Section 2.2.2. Redundancy, correctness and consistency problems arise when the tessellation is an approximation of an implicit input. Although many solutions have been proposed, these problems remain a challenge. Improvements are reviewed in Section 2.2.3.

2.2.1 Voxelization

The voxelization in the original MC algorithm from Lorensen and Cline [1987] adopts a fixed grid with the same sized cells which are cubes. This is because cube is one of the simplest space-filling polyhedron and is easy to construct. However, it causes not only ambiguity problems but also a waste of memory, especially when the size of cubes is chosen to be too small compared to the input volume. Many extensions have been proposed to solve these problems. Depending on the shape of cells, these methods can be divided into two categories: multi-resolution cells and non-rectangular cells.

In order to reduce the memory complexity and to generate triangles with adaptive size on the shape surface, Shu et al. [1995] considered an

adaptive MC that subdivides the cubes according to the shape surface smoothness. This method may cause cracks in the generated surface. Although a crack-patching method is applied for filling the cracks with polygons of the same shape, loss of finer features may happen when subdividing the cubes. An octree structure is used in [Shekhar et al., 1996] that replaces small cells with a larger one instead of subdividing. In order to patch the cracks, edges of the smaller cells are forced to coincide with those of their larger neighbors. This crack-patching method does not generate new triangles. Weber et al. [2003] extended the MC to rectilinear grid with multi-resolution. The cracks among the cells are then filled with polyhedra.

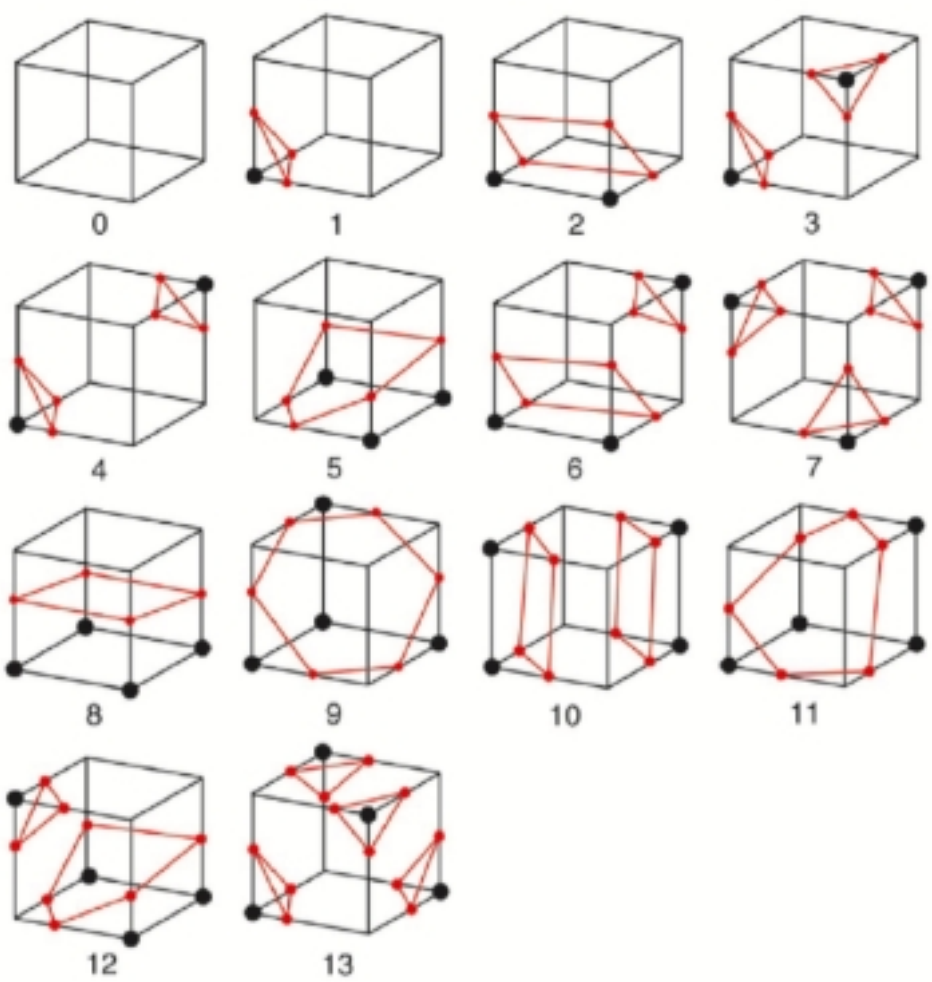
Instead of using cubes as cells, tetrahedra are firstly considered in an algorithm proposed in [Akio and Koide, 1991] that is called Marching Tetrahedra. It subdivides each cube cell into four tetrahedra. The advantage of using tetrahedra is that it simplifies the lookup table and eliminates the ambiguities of the facetization. However, the complexity increases and the orientation of each tetrahedron has to be defined. Marching Tetrahedra has been widely used in many applications and also been applied to rectilinear grid and multi-resolution [Elvins, 1992]. Other cell types such as hexahedron [Carr et al., 2003], octahedron [Carr et al., 2003] [Takahashi et al., 2004] and other irregular shapes [Newman and Yi, 2006] have been considered for the purpose of complexity reduction.

These extensions are mainly proposed for improving the generated surface quality without considering the regularity of the generated volumetric tessellation. In the following, we consider the voxelization with a fixed grid with the cubes of the same size because it generates more regular tessellations compared to the others.

2.2.2 Clipping

Depending on the type of input shapes, the clipping algorithm is either approximated or exact. In the exact case where the input shape is a mesh, the problem becomes a polyhedron intersection problem between the mesh and the cells. Sutherland's clipping algorithm [Sutherland and Hodgman, 1974] can be used. In Chapter 6, we introduce a novel, efficient and robust clipping algorithm. This work has been published in [Wang et al., 2016a].

When the input shape is an implicit surface, the standard clipping algorithm is the MC algorithm. The standard MC algorithm [Lorensen and Cline, 1987] first identifies the boundary cubes that intersect the implicit surface by marking the vertices inside the shape. There are 256



[Lopes and Brodlie, 2003]

Figure 2.5 – The 14 basic intersection topologies. Vertices are marked in black. The intersections between the implicit surface and the edges of cubes and their connections are in red.

(2^8) possibilities for a cube since a cube has eight vertices that can be either marked or unmarked. However, if rotational, reflective and mirror symmetries are considered, the intersection topologies can be reduced to only 14, as shown in Figure 2.5. For each topology, the approximated surface is generated by connecting the intersections between the implicit surface and the edges of cube. This information is stored in a look-up table that makes the process very fast.

The intersection between the implicit surface and the edges can be estimated using an interpolation method. Depending on the information available on the vertices, different methods can be applied. When only implicit function values are available, linear interpolation is widely used because it is fast and simple. The false position method gives better results by locating the intersection iteratively. When both implicit function values and their derivatives are available, Hermite interpolation is proposed to give an accurate intersection [Fuhrmann et al., 2015]. When no additional information than inside or outside is offered, the bisection method can be used which can iteratively locates the intersection within a precision defined by user.

It has been pointed out that there are ambiguities in the standard MC algorithm [Nielson and Hamann, 1991] [Natarajan, 1994]. In Figure 2.5, there are face ambiguities in cases 3, 6, 7, 10, 12 and 13 and interior ambiguities in cases 4, 6, 7, 10, 12 and 13. Chernyaev [1995] identified that there are 33 different cases where two of them can be removed because of reflective and mirror symmetries. Many methods have been proposed for disambiguation that are reviewed in Section 2.2.3.

2.2.3 Improvements

There are two main issues in approximated volumetric tessellations using the standard MC algorithm that are topological inconsistency and non-manifoldness. The topological inconsistency issue comes from the ambiguities in the standard MC cases as firstly noted by Dürst [1988]. The asymptotic decider method [Nielson and Hamann, 1991] has been proposed for solving the face ambiguity using bilinear interpolation of ambiguous faces' vertices. Then the lookup table of the standard MC algorithm has been extended to 33 cases [Chernyaev, 1995] by adding subcases for disambiguation. An interior discriminant has also been proposed for the internal ambiguity subcases by detecting bilinear interpolations over any plane inside the ambiguous cubes. Nielson [2003] and Lopes and Brodlie [2003] proposed trilinear interpolation methods that provide an el-

egant solution to both face and internal ambiguities. The implementation for [Chernyaev, 1995] is available in [Lewiner et al., 2003].

Manifoldness can be lost if the facetization generates non-manifold edges. Nielson [2003] and Lopes and Brodlie [2003] proposed to generate additional points on the boundary and inside cube for preserving both topological consistency and manifoldness. Etiene et al. [2012] mentioned that the implementation of [Lewiner et al., 2003] may fail to produce manifold surface. The method has been then improved by Custodio et al. [2013]. Recently, a novel method using quadratic equations is proposed for generating both topological consistent and manifold surface in [Grosso, 2016]. For more detail reviews on this topic, please see [Newman and Yi, 2006].

2.3 Delaunay-based Approaches

The main step of Delaunay-based approaches is the Delaunay triangulation that fills a shape volume $\mathcal{V}$ with three-dimensional triangles, *i.e.* tetrahedra. The computation of a triangulation of a point set can be defined as a process of associating the points by forming triangles. Given a finite set of points $\mathcal{X}$, a triangulation of $\mathcal{X}$ is a simplicial complex that tessellates the convex hull of $\mathcal{X}$ and whose vertices belong to $\mathcal{X}$. Several triangulations may be constructed from the same set of points. Among them, the Delaunay triangulation is the most interesting one since it possesses good properties. The Delaunay triangulation and its dual, the Voronoi diagram, are widely studied and used in many applications of different areas. After some extensions such as the constrained Delaunay and the restricted Delaunay triangulations have been proposed, it has been used for volumetric tessellations of shapes, also called volumetric mesh generation.

The remainder of this section is organized as follows: We first introduce the background on the Delaunay triangulation of a point set including its algorithms and extensions in Section 2.3.1. Then the Delaunay-based volumetric tessellations in the approximated case and in the exact case are reviewed in Section 2.3.2 and Section 2.3.3, respectively.

2.3.1 Background on Delaunay Triangulations

The Delaunay triangulation was first proposed by Delaunay [1934]. Given a finite point set $\mathcal{X}$, the Delaunay triangulation of $\mathcal{X}$ is the triangulation such that each triangle has a circumsphere which does not contain

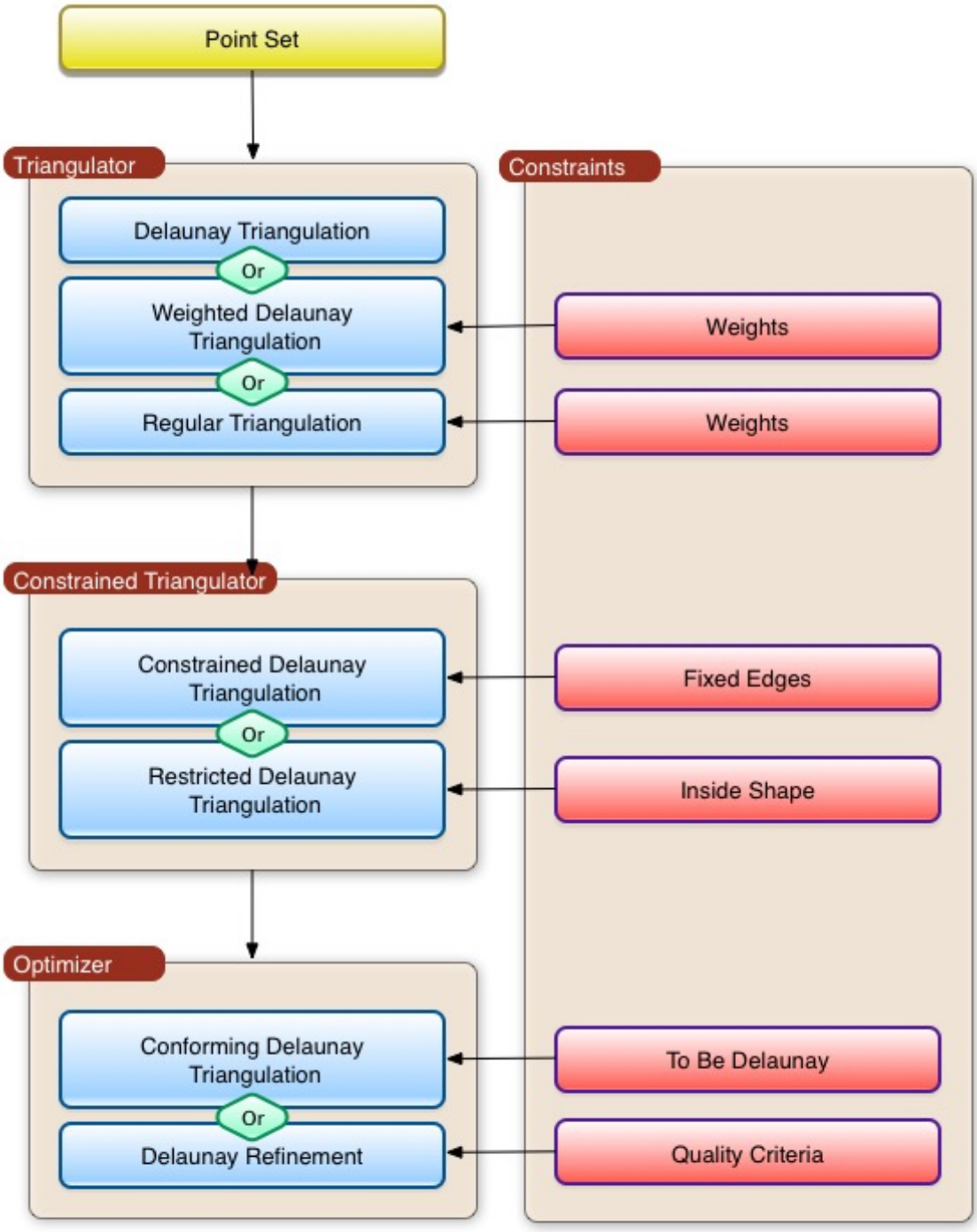


Figure 2.6 – The pipeline of general Delaunay triangulation computation.

the other points. Among all the triangulations of $\mathcal{X}$, the Delaunay triangulation maximizes the smallest angle of the triangles in the triangulation. This means that for the maximum minimum angle criterion, the Delaunay triangulation is the *best* one of all the triangulations of $\mathcal{X}$. If all the points in $\mathcal{X}$ lie on the same line, there is no Delaunay triangulation. If four or more points are cocyclic, the Delaunay triangulation is not unique.

A Delaunay triangulation can also be defined by higher dimension embedding. Given a finite point set $\mathcal{X} = \{x_i\}$ in an m-dimensional Euclidean space $\mathbb{E}^m$, *the parabolic lifting map* [Brown, 1979] transforms the Delaunay triangulation of $\mathcal{X}$ into a convex hull in $\mathbb{E}^{m+1}$. Each $x_i \in \mathcal{X}$ corresponds to $x_i' = (x_i, \|x_i\|^2) \in \mathcal{X}'$ in $\mathbb{E}^{m+1}$ and the Delaunay triangulation of $\mathcal{X}$ is the projection on the m-dimensional plane of the convex hull of $\mathcal{X}'$ in $\mathbb{E}^m$.

Because of its nice properties [Fortune, 1992] [Loera et al., 2010], the Delaunay triangulation is used for graph construction [Chew, 1986], network optimization [Mitchell, 2000] *etc*. Over the last decades, many extensions of the Delaunay triangulation have been proposed for requirements of different applications. We use the pipeline of the general Delaunay triangulation computation to gather the extensions in a systematic way. Figure 2.6 shows the pipeline that consists of a triangulator, a constrained triangulator and an optimizer. The triangulator takes a finite point set $\mathcal{X}$ as input and computes the Delaunay triangulation of $\mathcal{X}$. When $\mathcal{X}$ consists of points with weights, the weighted Delaunay triangulation or the regular triangulation methods can be computed. Then the triangulator passes the triangulation of $\mathcal{X}$ to the constrained triangulator which adjusts the input triangulation according to the given constraints. The constrained Delaunay triangulation method forces some fixed segments or polygons to appear in the triangulation and the restricted Delaunay triangulation method allows removing all the triangles that are outside the given shape. Since the above process cannot guarantee the triangulation to be Delaunay, the conforming Delaunay triangulation method in the optimizer splits the non-Delaunay edges by inserting additional points (Steiner points) and reconstitutes the triangulation until it satisfies the Delaunay criterion. The Delaunay refinement method subdivides the triangulation in order to improve its quality.

In Section 2.3.1.1, we review the algorithms for basic Delaunay triangulation of point sets. The main extensions of the Delaunay triangulation are listed in Section 2.3.1.2.

2.3.1.1 Delaunay Triangulation Algorithms

Algorithms for the three-dimensional Delaunay triangulation can be classified into the following categories: incremental reconstruction, incremental insertion, higher dimensional embedding and divide and conquer. The incremental reconstruction algorithm [Joe, 1991] starts with a single tetrahedron without containing any other of input points. Then input points are inserted successively to the existing Delaunay triangulation and uses local transformations known as the flip algorithm is used to build a new Delaunay triangulation. In order to accelerate the algorithm, the input points can be sorted before or stored in a uniform grid [Joe, 1991]. Unlike the incremental reconstruction algorithm that inserts points outside the existing triangulation, the incremental insertion algorithm inserts points inside it. In order to build a new triangulation, one strategy is to delete the tetrahedra whose circumsphere contains the inserted point and then to rebuild a triangulation to fill the hole [Bowyer, 1981] [Watson, 1981]. Another strategy is to use local transformations [Facello, 1995]. As mentioned earlier, the Delaunay triangulation can be defined by the projection of the convex hull in higher dimension. The input points are firstly embedded into four dimensions by the lifting map. Then their convex hull and its projection in three dimensions are computed to obtain the Delaunay triangulation [O'Rourke and Goodman, 2004]. A divide and conquer algorithm [Cignoni et al., 1998a] divides the input points into small partitions using splitting planes. Then the Delaunay triangulation of each partition is computed. The last step is to merge them to the final Delaunay triangulation. The parallel versions of some of the above algorithms are proposed in [Kohout and Kolingerová, 2003] [Kohout et al., 2005] [Lo, 2012].

2.3.1.2 Extensions of Delaunay Triangulations

In this section, we review the main extensions of the Delaunay triangulation. As mentioned earlier, the Delaunay triangulation can be defined in several ways, such as by the requirement that the circumspheres of all tetrahedra in triangulation do not contain other vertices or by the projection of convex hull in higher dimensions using the parabolic lifting map. In order to define the Delaunay triangulation and some of its extensions such as the weighted Delaunay triangulation or the regular triangulation in a consistent way, we use the notion of *Voronoi diagram*, also called *Voronoi tessellation*, which is the dual of the Delaunay triangulation.

Given a finite set of n points $\mathcal{X} = \{x_i\}_{i=1}^{n}$ in three-dimensional Eu-

clidean space $\mathbb{E}^3$, we define a *distance function* d_i, also called *cost function*, for each x_i as follows:

$$\begin{aligned} d_i : \mathbb{E}^3 &\longrightarrow \mathbb{R} \\ x &\longmapsto \|x - x_i\|^2, \end{aligned} \tag{2.1}$$

where $\|\cdot\|$ is the Euclidean distance. The *Voronoi region* Ω_i, also called *Voronoi cell*, of x_i is the set of points that are closest to x_i than to any other points of $\mathcal{X}$. That is:

$$\Omega_i = \{x \in \mathbb{E}^3 \,|\, d_i(x) \leq d_j(x),\ \forall j \neq i\}. \tag{2.2}$$

x_i is called the *site* of Ω_i. The *bisector* B_{ij} of two Voronoi cells Ω_i and Ω_j is the set of points that are shared by these two cells. That is

$$B_{ij} = \{x \in \mathbb{E}^3 \,|\, \|x - x_i\| = \|x - x_j\|\}. \tag{2.3}$$

The Delaunay triangulation of $\mathcal{X}$ can be constructed from the Voronoi diagram by connecting two sites whose corresponding Voronoi cells share a bisector. We use analogous definitions to define weighted Delaunay triangulation and regular triangulation in the following paragraphs.

Weighted Delaunay Triangulation

Given a finite set of n points $\mathcal{X} = \{x_i\}_{i=1}^n \in \mathbb{E}^3$ with weights $\{w_i\}_{i=1}^n \in \mathbb{R}$, we define a distance function d_i^w for x_i as follows:

$$\begin{aligned} d_i^w : \mathbb{E}^3 &\longrightarrow \mathbb{R} \\ x &\longmapsto \|x - x_i\| - w_i, \end{aligned} \tag{2.4}$$

where $\|\cdot\|$ is the Euclidean distance. Using this distance function, we can define an analog to the Voronoi diagram that is called *the additively weighted Voronoi diagram* with its cells defined as follows:

$$\Omega_i^w = \{x \in \mathbb{E}^3 \,|\, d_i^w(x) \leq d_j^w(x),\ \forall j \neq i\}. \tag{2.5}$$

The bisector B_{ij}^w of two additively weighted Voronoi cells is defined as follows:

$$B_{ij}^w = \{x \in \mathbb{E}^3 \,|\, \|x - x_i\| - w_i = \|x - x_j\| - w_j\}. \tag{2.6}$$

Recall that the weighted Delaunay triangulation of $\mathcal{X}$ can be obtained from the additively weighted Voronoi diagram by connecting two sites whose corresponding cells share a bisector.

Regular Triangulation

Given a finite set of n points $\mathcal{X} = \{x_i\}_{i=1}^n \in \mathbb{E}^3$ with weights $\{w_i\}_{i=1}^n \in \mathbb{R}$, we define a distance function π_i^w for x_i as follows:

$$\begin{aligned} \pi_i^w : \mathbb{E}^3 &\longrightarrow \mathbb{R} \\ x &\longmapsto \|x - x_i\|^2 - w_i, \end{aligned} \tag{2.7}$$

where $\|\cdot\|$ is the Euclidean distance. This function is also called *power function* and is used to define the *power diagram*. Then the regular triangulation of $\mathcal{X}$ can be defined in the same way as the Delaunay triangulation.

It has to be mentioned that although both the weighted Delaunay triangulation and the regular triangulation use weighted point sets, they use different distance functions. Intuitively, the bisectors of the regular triangulation are planes. However, the bisectors of the weighted Delaunay triangulation may be radical planes.

Constrained Delaunay Triangulation

A constrained Delaunay triangulation (CDT) [Chew, 1989] is an extension of the Delaunay triangulation that enforces certain geometric constraints into the triangulation such as segments and polygons in three dimensions. Usually, the tetrahedra that contain the constraints do not satisfy the Delaunay triangulation criterion, thus, a CDT is not necessarily Delaunay. The computation of CDTs is the main step of the exact Delaunay-based volumetric tessellation. The input mesh is considered as a set of constraints in the tessellation. CDT algorithms are reviewed in detail in Section 2.3.3.1.

Restricted Delaunay Triangulation

A restricted Delaunay triangulation (RDT) is an extension of the Delaunay triangulation that forces the triangulation to stay inside the input shape. In the exact case, only the tetrahedra inside the input mesh are kept. In the approximated case, the center of the circumsphere of the tetrahedron can be used for checking if the tetrahedron is inside the input implicit surface [Jamin et al., 2015]. The computation of RDTs is the main step of the approximated Delaunay-based volumetric tessellation and is reviewed in detail in Section 2.3.2.2.

Conforming Delaunay Triangulation

A conforming Delaunay triangulation is an extension of the CDT that subdivides the constraints by inserting additional points (Steiner points). These points allow to keep the constraints in the triangulation while satisfying the Delaunay criterion.

Delaunay Refinement

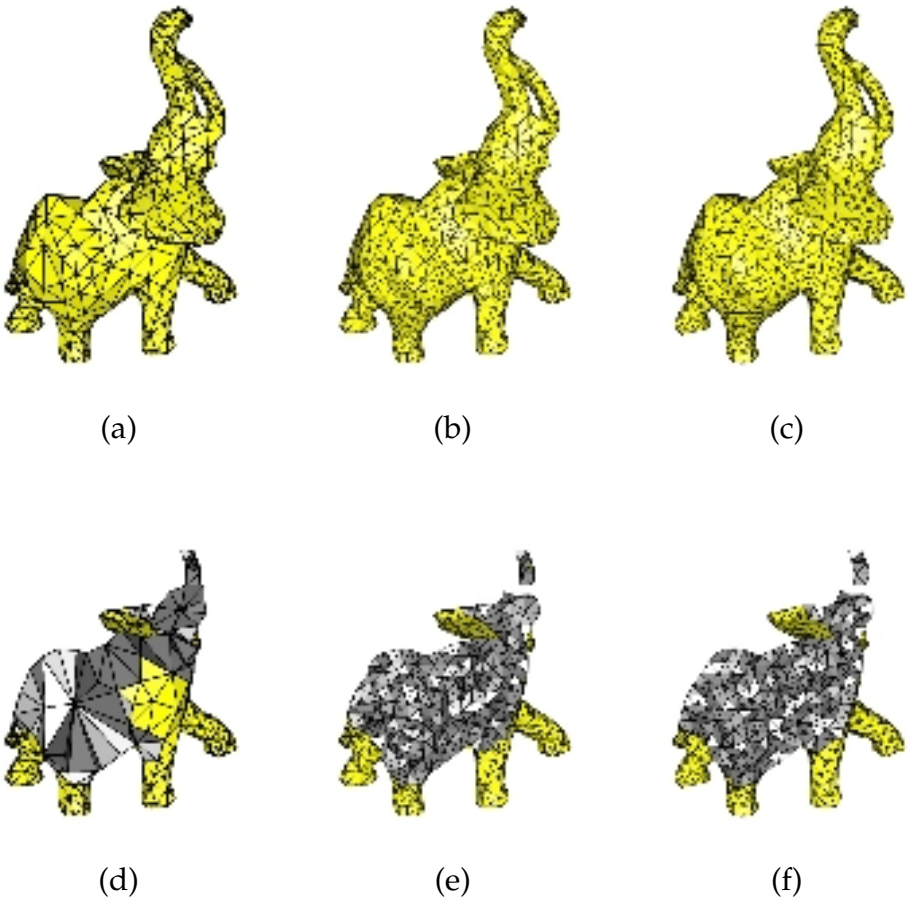
The Delaunay refinement is an optimization process for the purpose of improving the tetrahedra quality by subdividing the edges, facets or cells in tessellations. However, it also has several drawbacks. The process may produce badly shaped tetrahedra and may have non-terminating loops because of sharp features. Different versions of the Delaunay refinement are proposed for both the approximated and the exact cases. See the details in Section 2.3.2.3 and 2.3.3.2.

2.3.2 Approximated Case

The computation of an approximated Delaunay-based volumetric tessellation considers a shape volume $\mathcal{V}$ as input and builds a RDT inside $\mathcal{V}$. In this case, the input $\mathcal{V}$ is usually represented by an implicit surface $\mathcal{S}$ representing its boundary and the constructed RDT is supposed to approximate $\mathcal{V}$. The computation consists of the following three steps:

1. Initialization: sample a finite set of points $\mathcal{X}$ lying on $\mathcal{S}$. If sharp features are required to be preserved, the points lying on the sharp edges are sampled and inserted into $\mathcal{X}$.
2. RDT: build the RDT of $\mathcal{X}$ inside $\mathcal{V}$.
3. Optimization: subdivide edges, faces or cells of the RDT by inserting additional points (Steiner points) into $\mathcal{X}$ or change the position of the existing points in $\mathcal{X}$, then rebuild the RDT of the updated $\mathcal{X}$ until certain user-defined mesh quality criteria are satisfied.

Figure 2.7 gives an illustration of the results after different steps. As we can see, after step 1 and 2, a RDT with badly shaped tetrahedra is constructed (see Figure 2.7 (a) and (d)). (b) and (e) in Figure 2.7 show that the triangulation is subdivided by inserting additional points and the new constructed tetrahedra are better shaped. This subdivision process is called the Delaunay refinement and the additional points are called Steiner points. The quality of the triangulation can be further improved by updating the position of the points using variational methods (see Figure



(a) (b) (c)

(d) (e) (f)

Figure 2.7 – (a, d): The RDT of an implicit surface representing an elephant. (b, e): The result of (a) after Delaunay refinement. (c, f): The result of (b) after a Lloyd optimization.

2.7 (c) and (f)). These three steps are reviewed in detail in Section 2.3.2.1, 2.3.2.2 and 2.3.2.3, respectively.

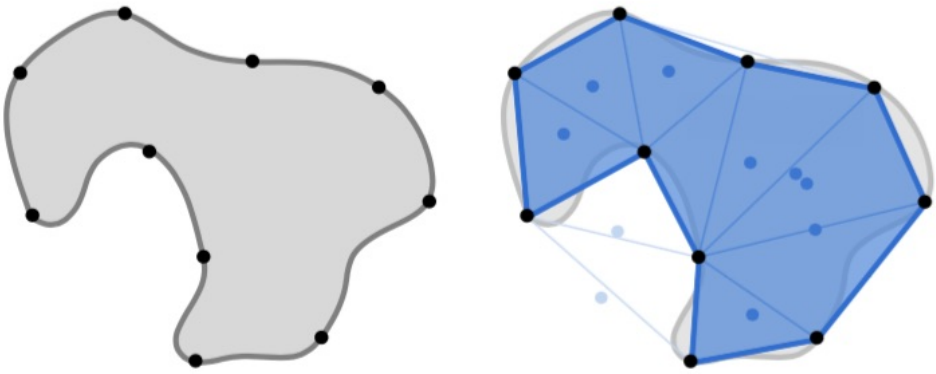
2.3.2.1 Initialization and Sharp Features

In order to build a RDT inside the input shape volume $\mathcal{V}$, a finite set of points $\mathcal{X}$ is sampled on the implicit surface $\mathcal{S}$ during the initialization step. Several methods can be used for obtaining these initial points such as random ray shooting for instance. $\mathcal{X}$ may be also provided by the user. It is worth mentioning that the uniform distribution and high density of $\mathcal{X}$ can improve the quality of the RDT so that the tetrahedra are more regular and the triangulation approximates better to $\mathcal{V}$. However, uniform

sampling on $\mathcal{S}$ can be difficult when $\mathcal{S}$ is complex. Instead of generating an uniform sampling in the intialization step, the quality of regularity and approximation of the RDT can be improved in the optimization step. During the computation of the Delaunay refinement, the Steiner points are inserted inside $\mathcal{V}$ or on $\mathcal{S}$ according to certain criteria of regularity and approximation. With the Delaunay refinement, the initialization step can be simplified to only sample a small set of points on $\mathcal{S}$.

Sharp features can be easily integrated into the Delaunay-based approach during the initialization step. The protecting-balls approach [Cheng et al., 2010] is proposed to ensure that the given sharp features appear in the final tessellation. The approach discretizes the sharp features into a set of weighted points that are called protecting balls. These weighted points are then inserted into $\mathcal{X}$ and the weighted Delaunay triangulation is used instead of the Delaunay triangulation in the whole triangulation process. The points except for the protecting balls have zero weight so that the sharp edges can be forced into the final triangulation.

2.3.2.2 Restricted Delaunay Triangulation



(a) (b)

Figure 2.8 – (a): The input shape with the sampling points on its implicit surface. (b): The RDT. The center of the circumcircle of the triangles is in blue. The images are from [CGAL].

The RDT is the main step of the approximated Delaunay-based volumetric tessellation. It consists of building a Delaunay Triangulation and restricting this Delaunay triangulation to the input shape volume $\mathcal{V}$. In order to remove the tetrahedra outside $\mathcal{V}$, the center of the circumspheres

are used. For each tetrahedron of the Delaunay triangulation, if the center of its circumsphere is outside $\mathcal{V}$, the tetrahedron is considered as outside $\mathcal{V}$ and has to be removed. Otherwise, the tetrahedron is kept. Figure 2.8 gives a two-dimensional illustration of the RDT. It is important to note that as a result of the process of removing tetrahedra, it is difficult to guarantee the topological consistency and the manifoldness of the final tessellation and its surface, even with a dense sampling [Oudot, 2008].

2.3.2.3 Optimization

The optimization step aims at building a *good* tessellation that approximates well the input shape volume $\mathcal{V}$ and is composed of regular tetrahedra. As mentioned earlier, it is difficult to obtain a uniform and dense sampling on the implicit surface $\mathcal{S}$ during the initialization step. With only the sampling points lying on $\mathcal{S}$, the RDT contains badly shaped tetrahedra (see Figure 2.7 (a) and (d)). The Delaunay refinement subdivides facets and cells of the triangulation by inserting Steiner points inside $\mathcal{V}$ and on $\mathcal{S}$. Variational methods and local optimization allow removing degenerate tetrahedra called *slivers*. Slivers are flat tetrahedra without large radius-edge ratio.

Delaunay Refinement

Given a RDT, the process of Delaunay refinement [Oudot et al., 2005] consists of the following steps: find the badly shaped surface facets and cells according to the user-defined criteria, insert Steiner points, update the RDT and repeating the above steps until all the faces and cells meet the criteria. The criteria are composed of facet criteria and cell criteria. Facet criteria are used for controlling the size, shape and approximation error of the surface facet. A *Steiner point* for removing a bad surface facet is defined as the center of the circumsphere of this facet whose center lies on the input surface. The cell criteria are used for controlling the size and shape of the cells. In order to remove a badly shaped cell, a *Steiner point* defined as the center of the circumsphere of this tetrahedron is inserted. With certain constraints on the criteria, the process of Delaunay refinement can be guaranteed to terminate after inserting a finite set of Steiner points [Chew, 1993] [Ruppert, 1995] [Shewchuk, 1998b]. However, the Delaunay refinement is insensitive to the slivers.

Variational Methods

As mentioned earlier, the Delaunay refinement may produce slivers. In order to remove the slivers, variational methods including the Lloyd relaxation [Du et al., 1999] [Du and Wang, 2003] and the optimal Delaunay triangulation relaxation [Chen and Xu, 2004] [Alliez et al., 2005] are proposed. The main idea of these methods is to define an energy function that quantifies the tessellation and to minimize this function by updating the position of the vertices. In the Lloyd relaxation, the function is defined on the dual Voronoi cells. In the optimal Delaunay triangulation relaxation, the function is defined directly on the tetrahedra. The minimization of the energy function allows the vertices to be uniformly distributed and thus to build a RDT with regular tetrahedra.

Local Optimization

Local optimization methods include vertex perturbation [Li, 2000] [Tournois et al., 2009] and sliver exudation [Cheng et al., 2000]. Unlike the variational methods that optimize the tessellation globally, local optimization aims to find the slivers and to remove them locally. The vertex perturbation method changes the position of vertices of the slivers with a perturbation until the connectivity of the Delaunay triangulation is updated. On the contrary, the sliver exudation method assigns weights to the vertices of the slivers and uses the weights to change the connectivity of the weighted Delaunay triangulation.

2.3.3 Exact Case

In the exact case, the input shape volume $\mathcal{V}$ is bounded by a triangle mesh $\mathcal{M}$. The vertices of $\mathcal{M}$ are considered as a set of points used for building a Delaunay triangulation and the edges and facets of $\mathcal{M}$ are considered as the constraints for CDT. The process consists of the following steps: building the DT of the vertices of $\mathcal{M}$, forcing the edges and facets of $\mathcal{M}$ into the DT, removing the tetrahedra outside of $\mathcal{V}$ and optimizing the quality of the tessellation. Figure 2.9 gives an illustration of the process.

2.3.3.1 Constrained Delaunay Triangulation

The three-dimensional CDT is the main step of the exact Delaunay-based volumetric tessellation. In two dimensions, the edge constraints can always be forced into the Delaunay triangulation using an edge flipping algorithm. However, the three-dimensional constrained Delaunay

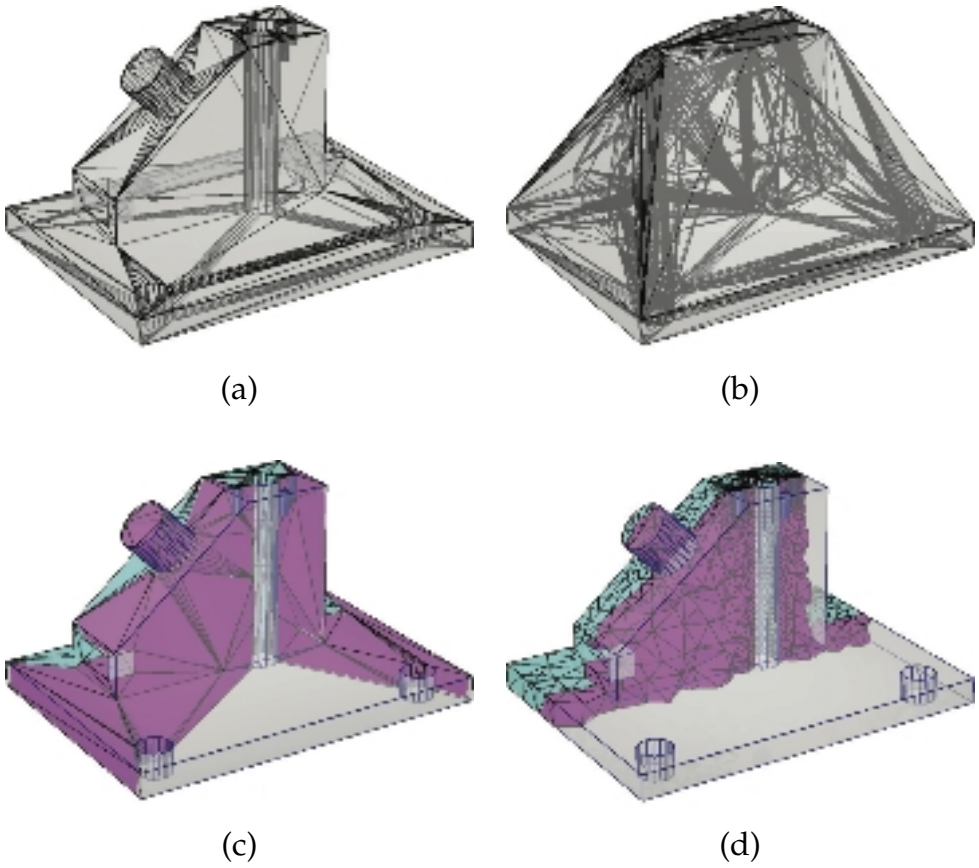


Figure 2.9 – Overview of the exact Delaunay-based volumetric tessellation. (a): Input shape represented by a mesh. (b): The Delaunay triangulation of the vertices of the input mesh. (c): The CDT with the constraints defined by the edges and the facets of the mesh. (d): Result after the Delaunay refinement. The images are from [Si, 2015].

triangulation does not always exist, except under the condition that all constraints are Delaunay [Shewchuk, 1998a]. Otherwise, Steiner points need to be inserted. The three-dimensional CDT consists of inserting the edges and the facets of the input mesh $\mathcal{M}$. Si and Gärtner [2005] introduced an algorithm to insert the edges while adding Steiner points to ensure the existence of CDT. The facets can be inserted using the flip algorithm [Shewchuk, 2003] or the cavity retetrahedralization algorithm [Si and Gärtner, 2011]. The latter is shown to be more robust in [Si and Shewchuk, 2014].

2.3.3.2 Optimization

The optimization step aims at generating well shaped tetrahedra as in the approximated case. Shewchuk and Si [2014] proposed a new version of the refinement algorithm for CDT. With certain modifications of splitting rules in the typical Delaunay refinement, the constrained Delaunay refinement can recover the sharp features and remove the slivers as well.

2.4 Voronoi-based Approaches

The Voronoi-based approach adopts a Lagrangian strategy that tessellates the shape volume $\mathcal{V}$ directly instead of a region Ω containing $\mathcal{V}$. The interest is to reduce the complexity when modeling large scenes. The approach builds a centroidal Voronoi tessellation (CVT) of $\mathcal{V}$ that is composed of uniform and regular cells. This can be an important feature for many applications.

The computation of a CVT consists of the following three steps:

1. Initialization: sample a a user-defined number of points $\mathcal{X}$ inside $\mathcal{V}$. These points are called sites.
2. Clipping: compute the intersection between the Voronoi tessellation of $\mathcal{X}$ and $\mathcal{V}$. The intersection is called clipped Voronoi tessellation.
3. Optimization: update the position of the sites until meeting user-defined criteria.

Figure 2.10 gives an illustration of the computation in two dimensions. The three steps are reviewed in detail in Section 2.4.2, 2.4.3 and 2.4.4, respectively. Background on Voronoi tessellation is first reviewed in Section 2.4.1 before entering into the details of the steps.

2.4.1 Background on Voronoi Tessellation

2.4.1.1 Voronoi Tessellation

Given a finite set of n points $\mathcal{X} = \{x_i\}_{i=1}^n$, called *sites*, in a m-dimensional Euclidean space $\mathbb{E}^m$, the *Voronoi cell* or *Voronoi region* Ω_i [Aurenhammer, 1991] [Fortune, 1992] [Okabe et al., 2009] of x_i is defined as follows:

$$\Omega_i = \{x \in \mathbb{E}^m \parallel \|x - x_i\| \leq \|x - x_j\|, \forall j \neq i\}, \tag{2.8}$$

where $\|\cdot\|$ is the Euclidean distance. The partition of $\mathbb{E}^m$ into Voronoi cells is called a *Voronoi tessellation*.

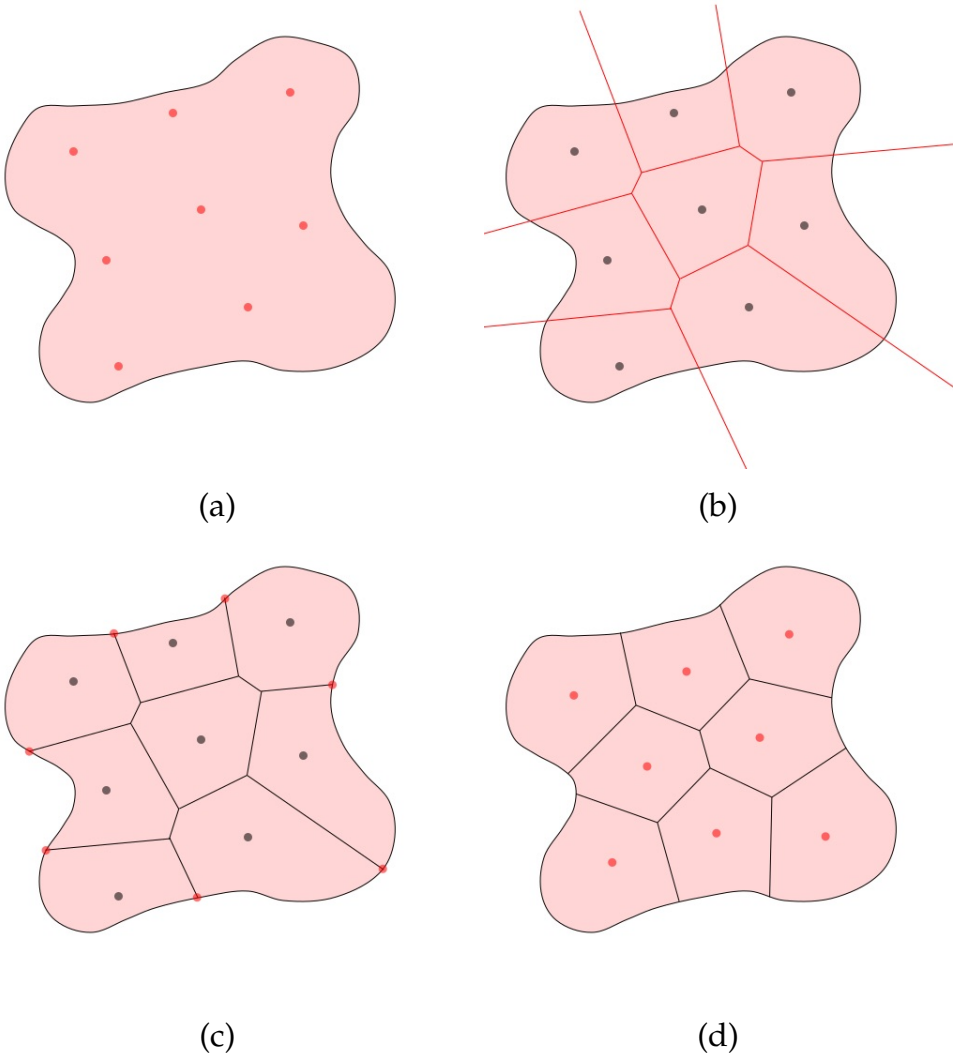


Figure 2.10 – Overview of the computation of a Voronoi-based tessellation in two dimensions. (a): Initialization: sample the sites inside the input shape. (b): Voronoi tessellation of the sites. (c): Clipping: compute the clipped Voronoi tessellation. (d): Optimization: update the position of the sites by minimizing the CVT energy function.

The Voronoi cell Ω_i of x_i is the set of the points whose distance to x_i is smaller than (or equal to) their distance to the other sites in $\mathcal{X}$. This is the solution to a semi-discrete (from continuous source to discrete target) optimal transportation problem [Lévy, 2015]. With a general distance metric, the general Voronoi cell Ω_i^g of x_i can be defined as follows:

$$\Omega_i^g = \{x \in \mathbb{E}^m \parallel d_i^g(x) \leq d_j^g(x), \forall j \neq i\}, \tag{2.9}$$

where d_i^g is the general distance to x_i. Extensions of Voronoi tessellation such as the weighted Voronoi tessellation [Okabe et al., 2009], the power diagram [Aurenhammer, 1987] and the L_p Voronoi tessellation [Lévy and Liu, 2010] can be considered as variations of the distance metric. In this thesis, we focus on the most widely used Voronoi tessellation with the Euclidean distance.

2.4.1.2 Centroidal Voronoi Tessellation

Voronoi cells intersecting the boundary of the shape are not closed. However, in many applications, only the intersection of the Voronoi cells with an input shape volume $\mathcal{V}$ are required. A *clipped Voronoi tessellation* [Yan et al., 2013] is the intersection between the Voronoi tessellation and $\mathcal{V}$. A *clipped Voronoi cell* is thus defined as:

$$\Omega_i = \{x \in \mathcal{V} \parallel \|x - x_i\| \leq \|x - x_j\|, \forall j \neq i\}, \tag{2.10}$$

A *centroidal Voronoi tessellation* [Du et al., 1999] is a special type of clipped Voronoi tessellation where the site of each Voronoi cell is also its center of mass. Let the clipped Voronoi cell Ω_i be endowed with a density function ρ such that $\rho(x) > 0 \, \forall x \in \mathcal{V}$. The center of mass $\hat{x}_i$, also called the *centroid*, of Ω_i is defined as follows:

$$\hat{x}_i = \frac{\int_{\Omega_i} \rho(x) x \, \mathrm{d}\sigma}{\int_{\Omega_i} \rho(x) \, \mathrm{d}\sigma}, \tag{2.11}$$

where $\mathrm{d}\sigma$ is the area differential.

CVTs are used to discretize two-dimensional or three-dimensional regions. In that respect, CVTs are optimal quantizers that minimize a distortion or quantization error $E : \mathbb{E}^{nm} \rightarrow \mathbb{R}$ defined as:

$$E(X) = \sum_{i=1}^{n} F_i(X) = \sum_{i=1}^{n} \int_{\Omega_i} \rho(x) \|x - \hat{x}_i\|^2 \, \mathrm{d}\sigma. \tag{2.12}$$

CVTs correspond to local optima of the above function E, also called CVT energy function [Du et al., 1999]. By definition, an *optimal CVT* achieves the global minimum of this function. Yet finding an optimal CVT appears difficult since the energy function is usually non linear and non convex [Liu et al., 2009] [Lu et al., 2012]. The function E measures the quantization error of a Voronoi tessellation and expresses, to some extent, the compactness of the cells [Liu et al., 2009]. However, it does not quantify how regular a tessellation is since it depends on the dimensions of the original region as well as the number of cells considered. Some regularity criteria are proposed in Chapter 3.

2.4.2 Initialization

The initial position of the sites has a strong influence on the convergence speed and on the result quality. Different methods have been considered in the literature.

Random Sampling

The idea is to sample the initial site locations randomly inside the input shape. This simple and fast method is widely used. However, neither the speed of convergence nor the quality of the result can be guaranteed.

Other sampling methods can be used to improve these criteria, such as farthest point sampling or Ward's method [Moriguchi and Sugihara, 2006].

Greedy Edge-collapsing

Moriguchi and Sugihara [2006] proposed a method which applies a greedy edge-collapsing decimation on the input shape and uses the decimated mesh vertices as the initial site positions. As pointed out by Quinn et al. [2012], this method can be time-consuming, and the sites may not be regularly positioned if the shape is not described by a regular mesh. Consequently, the quality of the resulting CVT can be even worse than using random sampling.

Hammersley Sampling

Quinn et al. [2012] suggested to use Hammersley sequences to generate the initial site positions. Hammersley sequences have correlated positions, which means that the probability of a site being at some position

depends on the positions of its neighbors. Unfortunately, the Hammersley sequence generation algorithm as described in [Quinn et al., 2012] can only place the sites in a square in two dimensions or a cube in three dimensions. As a result, the number of sites in the tessellation is difficult to control in any other cases.

Our Approach

A hierarchical approach will be detailed in Chapter 4.

2.4.3 Clipping

2.4.3.1 Exact Case

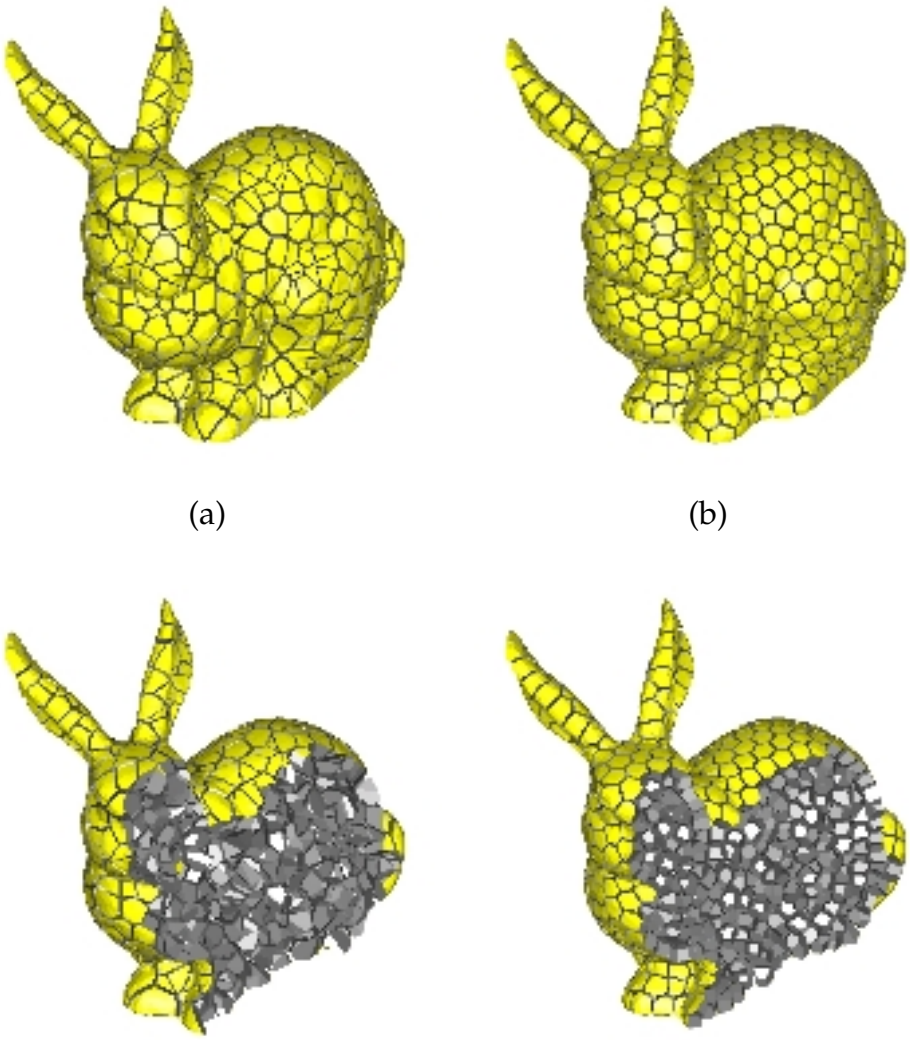
Yan et al. [2013] proposed an algorithm to compute the clipped Voronoi tessellation of three-dimensional shapes described by tetrahedral meshes. This algorithm consists of two main steps: detection of boundary sites by computing surface restricted Voronoi tessellation [Edelsbrunner and Shah, 1994] [Yan et al., 2009] and computation of the intersection between the Voronoi cells of boundary sites and the input tetrahedral mesh using Sutherland's clipping algorithm [Sutherland and Hodgman, 1974]. This method expresses the clipping problem as a three-dimensional volume intersection problem but also requires a tetrahedral mesh as input. When the input shape is given as a closed triangle mesh, a three-dimensional constraint Delaunay triangulation must be computed first [Shewchuk, 1998a] [Shewchuk, 2008]. This is a complex problem which has many degenerate cases and usually requires additional (Steiner) points to ensure the existence of a solution. The complexity highly depends on the quality of the input surface triangle mesh [Erickson, 2001]. Recently, Lévy [2014] proposed another efficient method based on iterative convex clipping.

In Chapter 6, we propose a novel algorithm that exploits a two-dimensional constrained Delaunay triangulation to determine triangles on the input mesh that intersect a given Voronoi cell without the need of a tetrahedral mesh inside the shape. This work has been published in [Wang et al., 2016b].

2.4.3.2 Approximated Case

In Chapter 5, we propose the first clipping algorithm to build the clipped Voronoi tessellation of implicit shapes.

2.4.4 Optimization



(a) (b)

(c) (d)

Figure 2.11 – (a, c): Voronoi tessellation of sites randomly placed inside the shape. (b, d): The CVT after the optimization.

The optimization step aims at building a tessellation with uniform and regular cells by minimizing the CVT energy function. Figure 2.11 gives an illustration of Voronoi tessellations with and without optimization.

Most of the strategies update the site positions using the Lloyd's gradient descent method [Lloyd, 1982]. At each iteration, this method

moves the current sites to the centroid positions of the corresponding clipped Voronoi cells. This is the continuous equivalent to the k-means clustering algorithm in the discrete case. It has been proved that this leads the CVT energy function to reach a local minimum [Du et al., 1999]. Convergence speed can anyway be slow since the site positions may oscillate around a local minimum.

To speed up convergence, Du and Emelianenko [2006] proposed a Lloyd-Newton method that is equivalent to minimizing the sum distances between this sites and the centroids of the corresponding Voronoi cells. Unfortunately, the resulting tessellation is not always a proper CVT since it is not necessarily a local minimum of the CVT energy function. In an influential work, Liu et al. [2009] proved that the CVT energy function has C^2 continuity almost everywhere, except for some non-convex parts of the shape. According to this property, quasi-Newton methods can be used to minimize the CVT energy function. This leads to fast and effective updates in practice.

Another strategy worth mentioning here is the stochastic approach of Lu et al. [2012]. In this iterative approach, the site positions are perturbed once a local minimum of the energy function is reached and the algorithm is then launched again. The global minimum can theoretically be reached after an infinite number of iterations. In practice, convergence is still slow, as shown in the experimental results in [Wang et al., 2016a].

Chapter 3

Regularity of a tessellation

Contents

3.1 Introduction

A regular volumetric tessellation in the usual sense consists of uniform and regular cells as defined in the next paragraph. Over the last decades, regular volumetric tessellations have been required in many applications of computer vision, computer graphics and other domains. Hausner [2001] proposed a method based on centroidal Voronoi regions for simulating decorative tile mosaics. Ringler et al. [2008] and Ju et al. [2011] applied centroidal Voronoi tessellation on climate modeling. Raviv et al. [2010] proposed a volumetric heat kernel signature using a quasi-regular volumetric tessellation obtained by computing the Marching Cubes algorithm. Regular volumetric tessellations have been also used as input data for three-dimensional tracking [Allain et al., 2015] [Huang et al., 2016], animation [Allain et al., 2016], shape detection [Wu et al., 2015], etc. Some of these works are shown in Figure 3.1.

A *polytope* is a geometric shape enclosed by hyperplans and is a generalized analog in higher dimensions of a polygon that is two-dimensional and a polyhedron that is three-dimensional. A polytope whose primitive elements are all symmetric is regular. More formally, a regular polygon

Hausner [2001]

Ringler et al. [2008]

Raviv et al. [2010]

Allain et al. [2015]

Figure 3.1 – Applications of regular volumetric tessellations.

is a polygon that is equiangular and equilateral. A regular polytope [Coxeter, 1968] of dimensions higher than or equal to three is defined recursively as a polytope with regular faces and regular vertex figures (see Figure 3.2) where a vertex figure at a vertex is a polytope obtained by joining the midpoints of edges of this vertex. Figure 3.3 shows examples of regular polygons and regular polyhedra.

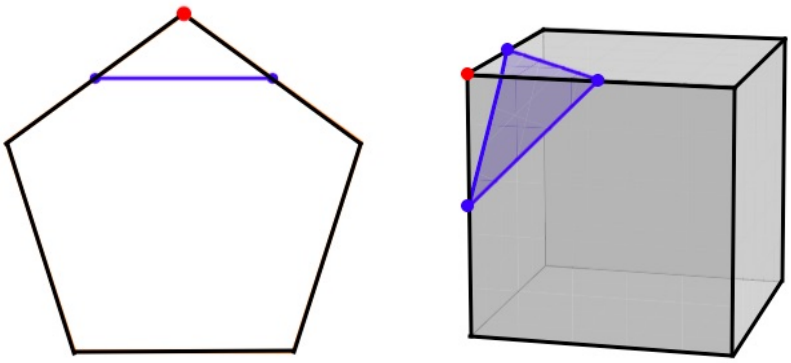


Figure 3.2 – Examples of the vertex figure (colored in blue) at a vertex (colored in red) of a pentagon and a cube.

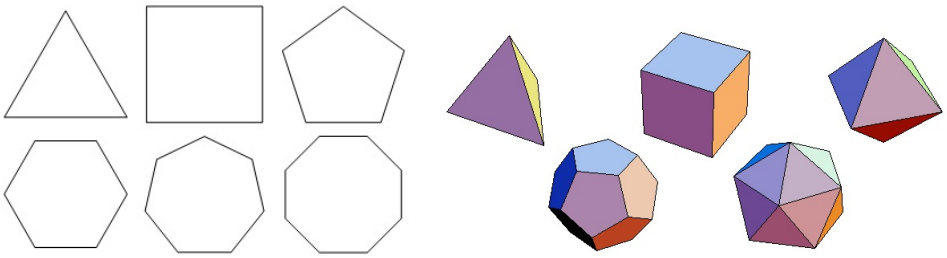


Figure 3.3 – Examples of regular polygons and regular polyhedra. The images are from [STUDYBLUE].

Since the strong symmetry of regular polytopes interests both artists and scientists, their properties have been well investigated. Some regularity measures based on area, perimeter, angle, edge, inscribed and circumscribed circles or spheres for convex polygons and polyhedra have been studied [Coxeter, 1938] [Zunic and Rosin, 2004] [Schulte, 2004] [Chalmeta et al., 2013] and methods for regular polygons detection have been proposed [Shaw et al., 2004] [Barnes et al., 2010] [Chalmeta et al., 2013]. However, these regularity criteria can not be used for comparing different

types of polytopes. Conway and Sloane [1982] proposed a normalized second order moment as a regularity criterion for polytopes which allows evaluating the regularity of any polytope and comparing them.

Nevertheless, the regularity of a tessellation attracts less attention. A measure is desirable to assess the regularity of a tessellation. This is true also when comparing different volumetric tessellations of different shapes. To the best of our knowledge, such a regularity measure has not yet been proposed. In this chapter, we build on the theoretical work of Conway and Sloane [1982] and propose regularity criteria based on the normalized second order moments of the cells. We show that these regularity criteria are linked to the CVT energy function but are dimensionless and therefore enable global evaluations as well as comparisons.

The remainder of this chapter is organized as follows. In Section 3.2, we review related work on dimensionless second moment of polytopes. We propose our regularity criteria for volumetric tessellation in Section 3.3. Section 3.4 discusses the theoretical relation between CVT and our regularity criteria.

3.2 Dimensionless Second Moment

The second moment of a polytope P, also called variance, measures how far the points inside P are spread out relating to its centroid. It can be defined as follows:

$$I(P) = \int_P \|x - \hat{x}\|^2 \, \mathrm{d}x, \tag{3.1}$$

with $\hat{x}$ the centroid of P.

If the volume of a polytope is fixed, a small second moment indicates that its points tend to be close to its centroid. It is easy to imagine that the two-dimensional polygon which possesses the smallest second moment tends to a circle, and in three dimensions, the polyhedron tends to a sphere. The second moment of a polytope can be used as a regularity criterion. However, it depends on the volume of the polytope.

Quantization has been used for studying the properties of tessellations over the last decades [Gersho, 1979] [Conway and Sloane, 1982]. Quantization is the process which maps a continuous or a large discrete set into a relatively small discrete set [Figueiredo, 2008] [Gersho and Gray, 2012]. Let a finite set of points $\{x_1, ..., x_n\}$ in m-dimensional Euclidean space $\mathbb{R}^m$ be a codebook. An m-dimensional quantizer maps each point $x \in \mathbb{R}^m$ into its closest codepoint x_i. If the density function of the source is $p(x)$,

the mean-squared error per symbol of this quantizer is defined as follows:

$$E(m,n,p,\{x_i\}) = \frac{1}{m}\sum_{i=1}^{n}\int_{\Omega_i}\|x-x_i\|^2 p(x)\mathrm{d}x, \tag{3.2}$$

where Ω_i is the Voronoi cell or the Voronoi region [Aurenhammer, 1991] [Fortune, 1992] [Okabe et al., 2009] of x_i.

Given m, n and $p(x)$, let us note the lower bound of E over all choices of $\{x_i\}$ as follows:

$$E(m,n,p) = \inf_{\{x_i\}} E(m,n,p,\{x_i\}).$$

Under different but quite general assumptions, Gersho [1979], Yamada et al. [1980], Bucklew [1981], Bucklew and Wise [1982] and Zador [1982] showed that

$$\lim_{n\to+\infty} n^{2/m}E(m,n,p) = G_m\left(\int_{\mathbb{R}^m} p(x)^{m/(m+2)}\mathrm{d}x\right)^{(m+2)/m}, \tag{3.3}$$

where G_m depends only on m.

Gersho [1979] conjectured that, for a sufficiently large n, any optimal quantizer is such that all Voronoi cells $\{\Omega_i\}$ are congruent to some polytope, with the possible exception of regions touching the boundary of the tessellated object. The polytope only depends on the dimension m. For such a quantizer, it can be deduced from 3.2 and 3.3 that, for any polytope P with uniform density,

$$G(P) = \frac{1}{m}\frac{\int_P \|x-\hat{x}\|^2\,\mathrm{d}x}{\left(\int_P \mathrm{d}x\right)^{(m+2)/m}}, \tag{3.4}$$

where $\hat{x}$ is the centroid of P.

$G(P)$ is called the dimensionless second moment of P. It is a measure which depends neither on the dimension m nor on the volume of P, only on its shape. With this property, $G(P)$ can be used as a regularity criterion for any polytope. After some calculations, Conway and Sloane [1982] gave a comparison of $G(P)$ for various three-dimensional polyhedra, see Table 3.1.

3.3 Regularity Criteria

We are interested in the regularity of tessellations. Since any quantizer maps each point x of the input object Ω into its closest codepoint $\{x_i\}$,

P	$G(P)$
tetrahedron	0.1040042 ...
cube*	0.0833333 ...
octahedron	0.0825482 ...
hexagonal prism*	0.0812227 ...
rhombic dodecahedron*	0.0787451 ...
truncated octahedron*	0.0785433 ...
dodecahedron	0.0781285 ...
icosahedron	0.0778185 ...
sphere	0.0769670 ...

Table 3.1 – Comparison of $G(P)$ for various polyhedra in three dimensions [Conway and Sloane, 1982]. P* means a space-filling polyhedron.

each x_i lies at the centroid of the corresponding tessellated cell Ω_i, this tessellation of Ω can be considered as a Voronoi tessellation and $\{x_i\}$ as its sites. It has to be mentioned that the CVT energy is usually used for evaluating the regularity of tessellations [Liu et al., 2009]. However, while this energy accounts for the compactness of the cells [Liu et al., 2009], it is a metric that depends both on the number of cells and on the volume of the shape.

From now on we assume that the density of Ω is uniform and the number of $\{x_i\}$ is large enough to avoid the boundary effect of Ω. Gersho's conjecture was proved in two dimensions [Newman, 1982], the Voronoi cells being regular hexagons in that case. A weaker version of Gersho's conjecture was also proved in three dimensions [Barnes and Sloane, 1983]. It states that among all lattice-based CVTs (i.e., regular CVTs, where sites are located on a regular grid), the body-centered cubic (BCC) lattice is the optimal one. The BCC lattice has its sites displayed on a regular cubic grid, with additional sites at the center of each cube, see Figure 3.4. The Voronoi tessellation of a BCC lattice is called a bitruncated cubic honeycomb, see Figure 3.5. Each of its cells is a truncated octahedra. Thus, Voronoi cells are truncated octahedra for optimal lattice-based CVTs in 3D.

Using Gersho's conjecture in the unbounded case, Conway and Sloane showed that the lower bound of $G(P)$ for any space-filling polytope in two dimensions is that of the hexagon:

$$G_2 = \frac{5}{36\sqrt{3}} = 0.0801875\ldots$$

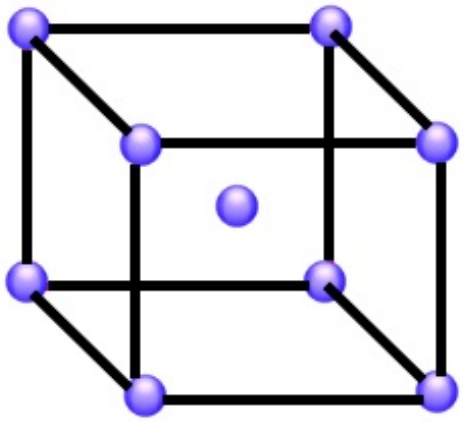


Figure 3.4 – Cell of a body-centered cubic lattice.

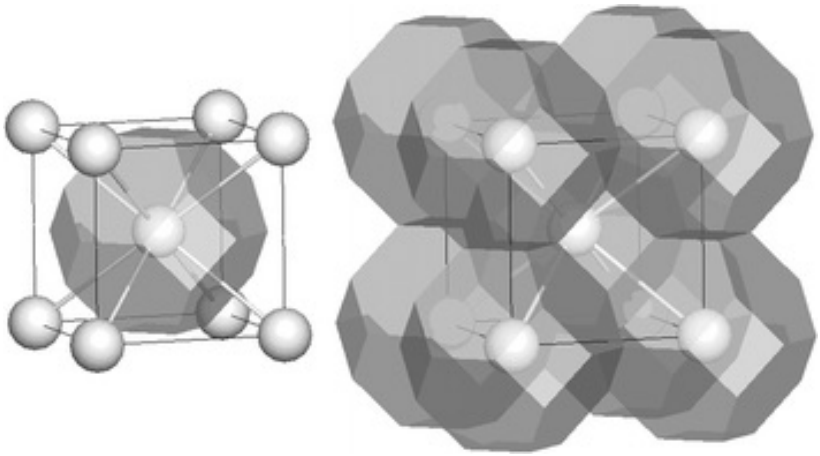


Figure 3.5 – Voronoi tessellation of a body-centered cubic lattice. The images are from [Blatov et al., 2004].

Similarly, in three dimensions, and with unbounded lattices, the optimal lattice-based CVT being the Voronoi tessellation of a BCC lattice, the optimal quantizer is the truncated octahedron with the lower-bound G_3:

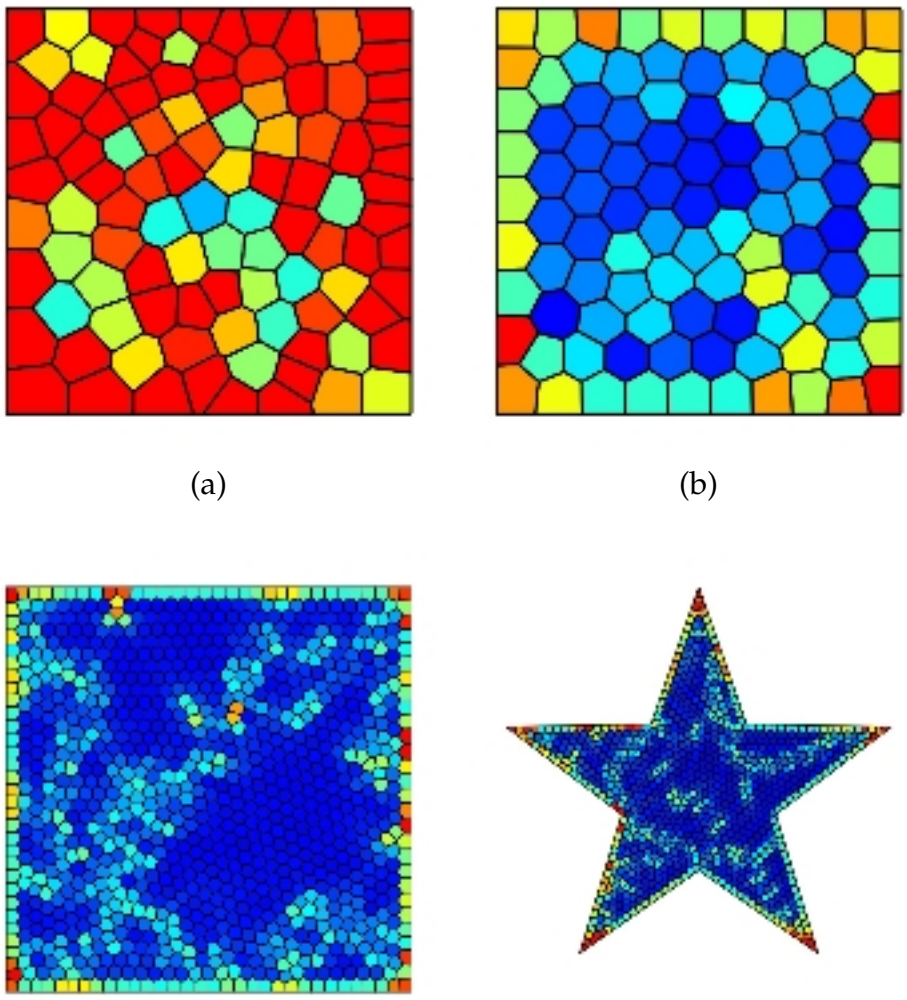
$$G_3 = \frac{19}{192\sqrt[3]{2}} = 0.0785433\ldots$$

Consequently, for a sufficiently large number of sites and with the exception of the boundary regions, an optimally regular m-dimensional tessellation should contain cells Ω_i with values of $G(\Omega_i)$ close to the optimal value G_m. Thus, G is a measure of the regularity of a cell since in the limit, with an infinite number of sites, all cells should reach the value G_m. Note here that we assume a large number of cells and that this reasoning does not apply to the boundary cells, for which the optimal quantizers are not necessarily hexagons (truncated octahedra in 3D) nor necessarily space-filling polytopes. However, under the assumption that the number of boundary cells is substantially lower than interior cells, the distribution of the values of G is a good indicator of the regularity of cells for a given tessellation where the regularity is defined with respect to the dimensionless moment G.

Based on Gersho's conjecture and the work of Conway and Sloane, we proposed different criteria for evaluating the regularity of tessellations. The regularity criteria are defined as follows:

- $G_{max} = \max G(\Omega_i)$: Max value of G.
- $\overline{G} = \frac{1}{n}\sum_{i=1}^{n} G(\Omega_i)$: Average value of G.
- $\widetilde{G} = \underset{i\in\{1\ldots n\}}{\text{median}}\{G(\Omega_i)\}$: Median value of G.
- $G_{rmse} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(G(\Omega_i) - G_m)^2}$: Root-mean-square error (RMSE) of G.

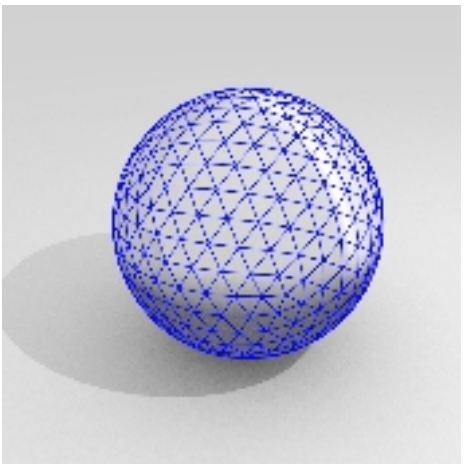
We now provide an application of our regularity criteria. Figure 3.6 shows a qualitative color-coded evaluation of different tessellations in 2D and Table 3.2 shows their corresponding numerical results. As we can see, our regularity criteria allow comparing the tessellations with the same number of cells (see (a) and (b)), with different number of cells (see (b) and (c)) and with different input objects (see (c) and (d)). We can also use the histogram of G to show the regularity distribution of tessellations, see Figure 3.7.



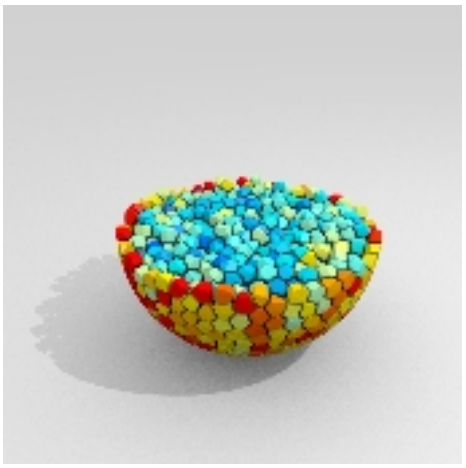
(a) (b)

(c) (d)

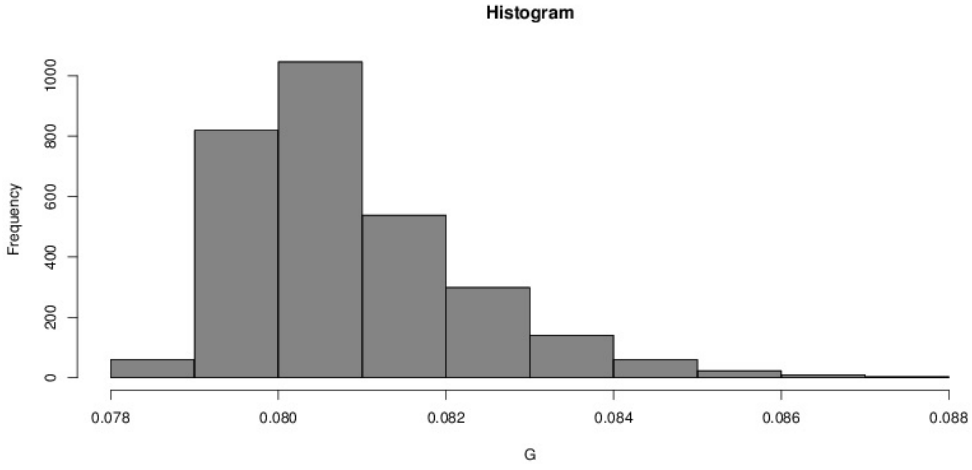
Figure 3.6 – Example of different tessellations in 2D. The cell regularity measure $G_2(\Omega_i)$ is color-coded from blue (regular) to red (far from regular).



(a)



(b)



(c)

Figure 3.7 – Example of a tessellation in 3D with its histogram of cell regularity. (a) Input sphere. (b) A cut of the tessellation. (c) Histogram of its cells regularity.

object	$G_{max} \times 10^{-2}$	$\overline{G} \times 10^{-2}$	$\widetilde{G} \times 10^{-2}$	$G_{rmse} \times 10^{-3}$
(a)	10.772	8.677	8.538	8.066
(b)	8.723	8.185	8.152	2.140
(c)	8.828	8.098	8.058	1.254
(d)	10.904	8.130	8.065	2.527

Table 3.2 – Numerical results corresponding to the tessellations in Figure 3.6.

3.4 Relation to the CVT Energy Function

The CVT energy function expresses the sum of compactness of the cells. It has been used to evaluate CVTs. However, since the CVT energy function depends on the input object and the number of cells, it can only be used to compare CVTs of the same input object and the same number of sites. The computation of a CVT is equivalent to the optimization of its energy function. The stopping of the algorithm is usually controlled by number of iterations because the optimal value of a CVT energy function which is non-convex can not be determined when the number of sites is large. In this section, we propose a formula of the optimal value of a CVT energy function by discovering its relation with $\overline{G}$ and we give an example which uses the optimal value of a CVT energy function to control the stop of a CVT computation instead of the number of iterations.

Given a input object Ω with a density function ρ and a set of sites $X = \{x_i\}_n$, the CVT energy function is defined as follows:

$$E(X) = \sum_{i=1}^{n} \int_{\Omega_i} \rho(x) \|x - x_i\|^2 \, \mathrm{d}\sigma,$$

where $\Omega = \bigcup_{i=1}^{n} \Omega_i$. Under the assumption of a uniform density function ρ and using the definition 3.4, the CVT energy E can be rewritten as:

$$E(X) = \sum_{i=1}^{n} mV(\Omega_i)^{(m+2)/m} G(\Omega_i),$$

where $V(\Omega_i) = \int_{\Omega_i} \mathrm{d}x$. Using Gersho's conjecture with unbounded lattices, optimal CVTs present in that case similar cells with volumes V/n and hence:

$$E(X) \sim mn \left(\frac{V}{n}\right)^{(m+2)/m} \overline{G}(X).$$

Thus optimizing the CVT energy E is equivalent to optimizing the average value $\overline{G}$ of G with infinite lattices. Knowing the theoretical optimal quantizer G_m in that case, we can even deduce that the value of an optimal CVT energy:

$$E_m = mn \left(\frac{V}{n}\right)^{(m+2)/m} G_m. \tag{3.5}$$

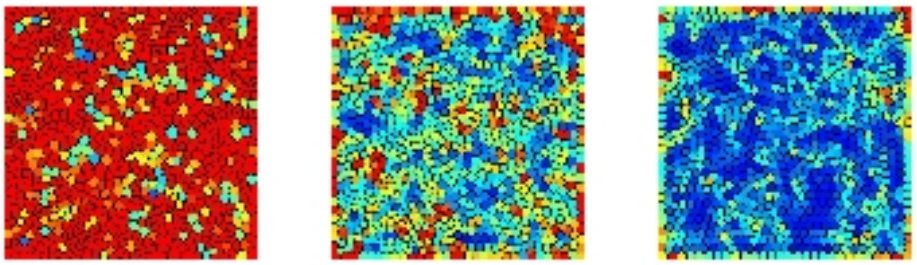
Then we propose a relative error defined as follows:

$$Error(X) = \frac{E(X) - E_m}{E_m}.$$

This error allows evaluating how close a CVT is to its optimal. We propose a novel CVT algorithm using this error measure. See Algorithm 1 and results in Figures 3.8 and 3.9.

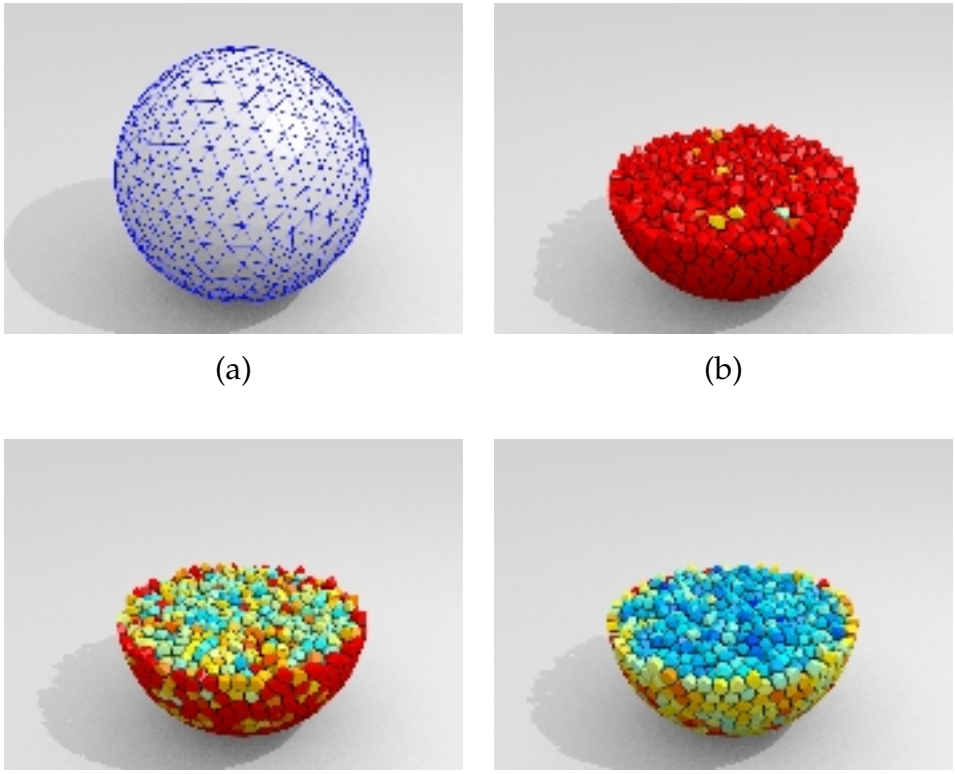
Algorithm 1: CVT algorithm controlled by relative error.

```
Data: object Ω, sites X, relative error e
Result: CVT {Ω_i}
e' := Error(X);
while e' > e do
    X := CVTUpdate(X);
    e' := Error(X);
end
{Ω_i} := ClippedVoronoiTessellation(X);
```



(a) (b) (c)

Figure 3.8 – Results of Algorithm 1 in 2D (square with 1000 sites). (a) $Error(X) = 0.2$. (b) $Error(X) = 0.05$. (c) $Error(X) = 0.02$. The cell regularity measure $G_2(\Omega_i)$ is color-coded from blue (regular) to red (far from regular).



(a) (b)

(c) (d)

Figure 3.9 – Results of Algorithm 1 in 3D (sphere with 5000 sites). (a) Input object. (b) $Error(X) = 0.2$. (c) $Error(X) = 0.05$. (d) $Error(X) = 0.02$. The cell regularity measure $G_3(\Omega_i)$ is color-coded from blue (regular) to red (far from regular).

Chapter 4

A Hierarchical Approach for Regular CVTs

Contents

Centroidal Voronoi Tessellations (CVTs) have been used in many applications because of their strong regularity property. Given the number of sites and a input shape that is usually represented by its boundary surface S (a closed polygonal curve in 2D and a manifold mesh M without boundary in 3D), the CVTs can be characterized by their energy function E. Each one of these CVTs corresponds to a local minimum of E. The CVT corresponding to the global minimum of E is called the optimal CVT and possesses the best regularity among all the CVTs. Over the last decades, some algorithms to compute CVTs with better local minimum have been proposed. They can be categorized in two strategies. One is to find an initialization that leads to a better local minimum, see examples [Moriguchi and Sugihara, 2006] [Quinn et al., 2012]. Another strategy is to improve the CVT optimization step. This is the case of the stochastic approach proposed by Lu et al. [2012]. These algorithms are described in Sections 2.4.2 and 2.4.4.

In this chapter, we propose a hierarchical approach for computing CVTs with better regularity than existing approaches. Our algorithm falls in the first category and is based on a subdivision scheme that preserves cell regularity and the local optimality of CVTs on unbounded domains. This scheme tends to propagate cell regularity through hierarchy levels

when applied to bounded domains. The remainder of this chapter is organized as follows. In Section 4.1, we explain our algorithm in details. We evaluate our algorithm and compare with others in Section 4.2 using not only the CVT energy but also the criteria proposed in Section 3.3. Section 4.3 discusses the advantages and the limitations of our approach and also possible future work.

4.1 Algorithm

4.1.1 Overview

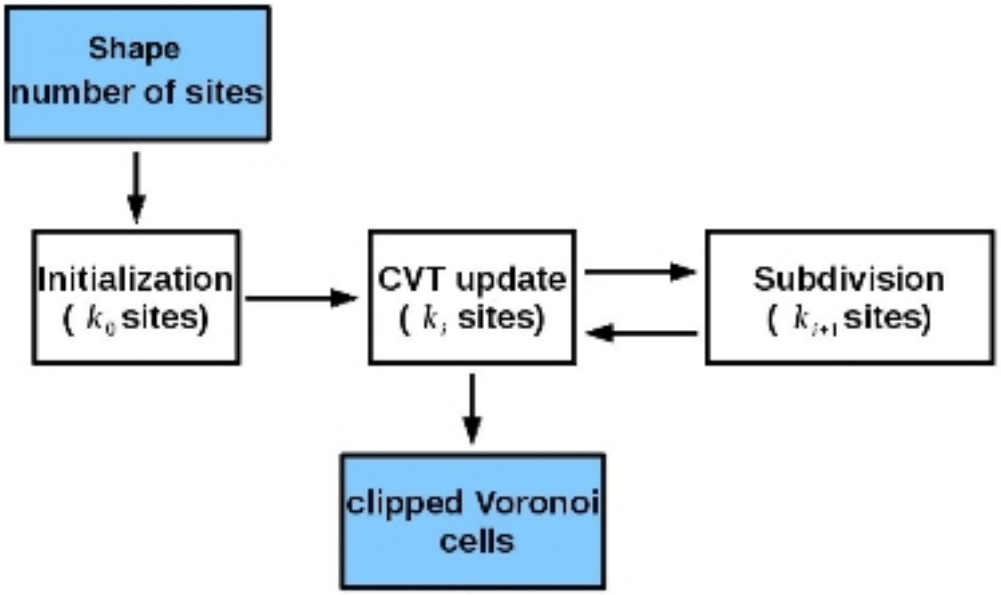


Figure 4.1 – Overview of our hierarchical approach.

We now explain our algorithm to compute CVTs that exploits regularity aspects. Figure 4.1 shows the overview of our hierarchical approach. Our input is a 2D or 3D shape, represented by its boundary: a closed polygonal curve in the first case and a manifold mesh without boundary in the second case. We also ask the user to provide the target number n of sites in the final CVT and the desired number s of subdivisions. From n and s we derive an initial number k_0 of sites, as explained in Section 4.1.2. We elaborate on the choice of these parameters in Section 4.2.

We first create an initial CVT of the shape with k_0 sites. We then subdivide this tessellation, as explained in Section 4.1.2. This generates a

new tessellation with k_1 sites, which is not Centroidal Voronoi. These sites are then moved towards the centroids of their cells to generate a new CVT with k_1 sites. We iterate the subdivision - CVT update process s times, until the desired number $k_s = n$ of sites has been reached.

4.1.2 Subdivision

The idea behind our subdivision scheme is to preserve the local optimality of the CVT. For example in the 2D case, a CVT is locally optimal with respect to our regularity criteria when its sites form an hexagonal lattice (see Figure 4.2 (a)), as explained in Section 3.3. Hence, our goal is to add sites such that the new set of sites keeps forming an hexagonal lattice. In this way, the new CVT will also be locally optimal with respect to regularity in the same area. In non-optimal areas, sites will move and possibly align to form a locally optimal lattice. Thus, iterating the subdivision - CVT update process tends to increase the area where the CVT is optimal for regularity, as shown on Figure 4.4. With a large number of subdivision s, most interior cells are expected to be regular.

Let X be a set of sites of a given CVT. To subdivide this CVT, we compute its dual Delaunay triangulation and add the center of each Delaunay edge to X. As shown on Figures 4.2 and 4.3, this preserves the local optimality of the CVT.

The previous subdivision scheme does not account for the desired number n of sites. To set up the initial number of sites k_0 such that it reaches the value n after s subdivisions, we proceed in the following way.

Let X be an hexagonal lattice, that is a to say an optimal 2D CVT, with k_i sites. Our subdivision scheme generates a new tessellation with $k_{i+1} = k_i + \frac{6k_i}{2} = 4k_i$ sites, since a new site is created on each of the six edges of a cell, an edge is shared by two cells and there are k_i cells. In the optimal 3D case (BCC lattice), the same reasoning shows that $k_{i+1} = k_i + \frac{14k_i}{2} = 8k_i$, since a site is added on each of the 14 faces of a truncated octahedron. The maximum number of iterations to reach n from a small number k_0 of sites in these ideal cases is thus $s = \lfloor \log_4(n) \rfloor$ and $\lfloor \log_8(n) \rfloor$, respectively.

Thus, if $s \geq \lfloor \log_a(n) \rfloor$, with $a = 4$ in the 2D case and $a = 8$ in the 3D case, we change s to $\lfloor \log_a(n) \rfloor$. We then define s numbers $b_1, \ldots, b_s$ such that $b_s = n$ and $b_i = \lfloor b_{i+1}/a \rfloor$, $1 \leq i \leq s-1$. b_i represents the target number of sites after i iterations. We also define $k_0 = \lfloor b_1/a \rfloor$. After the i-th subdivision, we check the new number k_i of sites. In case of an optimal CVT, $k_i = b_i$. Otherwise, $k_i \leq b_i$. If k_i is smaller than b_i, we randomly

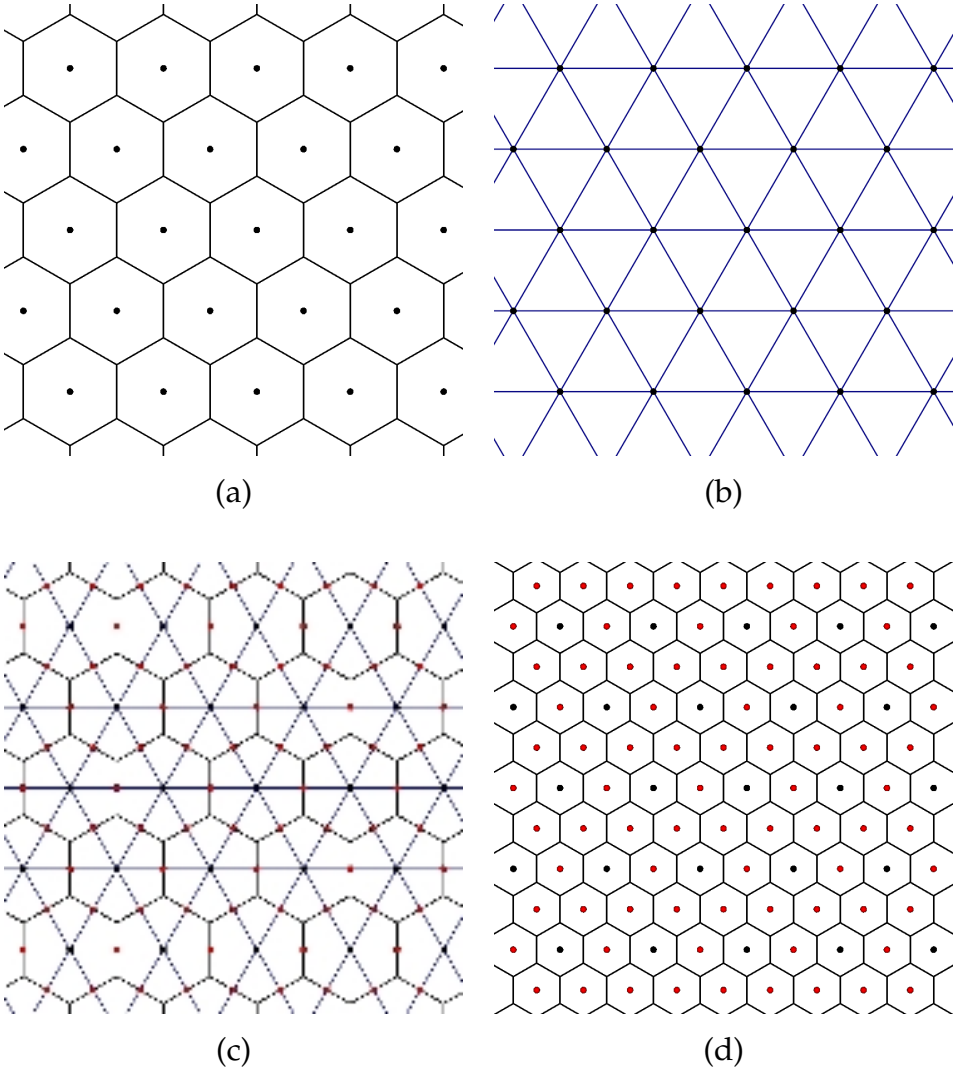
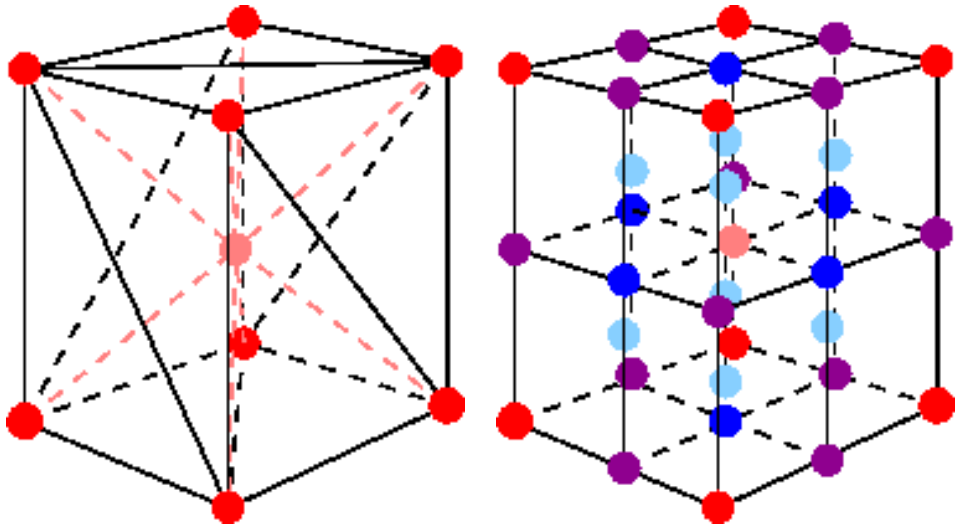


Figure 4.2 – Subdivision scheme (2D case). (a) Locally optimal CVT: the sites form an hexagonal lattice. (b) Delaunay triangulation of the sites. (c) Subdivision: sites are added in the centre of each edge of the Delaunay triangulation (in red). (d) The new set of sites also forms an hexagonal lattice.



(a) (b)

Figure 4.3 – Subdivision scheme (3D case). (a) Delaunay triangulation of the sites, which form a BCC lattice. (b) Subdivision: sites are added in the centre of each edge of the Delaunay triangulation (in blue and purple). The new set of sites also forms a BCC lattice.

sample $b_i - k_i$ new sites inside the *boundary* Voronoi cells. When a new site is inserted into a boundary cell, this cell is then removed from the list of candidate boundary cells for next insertions. This way, the regularity is empirically preserved as much as possible since sites are inserted in different boundary cells, be avoiding many new sites to be neighbours to each other. We thus have b_i sites after the i-th iteration, which are as regularly sampled as possible. This will improve the speed of the CVT update computation, which we describe in the next section.

As an example, Table 4.1 gives the number k_i of sites obtained after each subdivision and the number $b_i - k_i$ of sites added in the boundary cells, for CVTs depicted in Figures 4.4, 4.5 and 4.6.

4.1.3 CVT update

Once a new set of sites is defined, any CVT computation method can be used to move these sites towards the centroid of their Voronoi cells. In practice, we use the L-BFGS quasi-Newton algorithm, since it is known to be one of the fastest methods [Liu et al., 2009]. As explained in the previous section, the sites where the previous CVT was optimal are not moved, thanks to our subdivision scheme. As a consequence, although

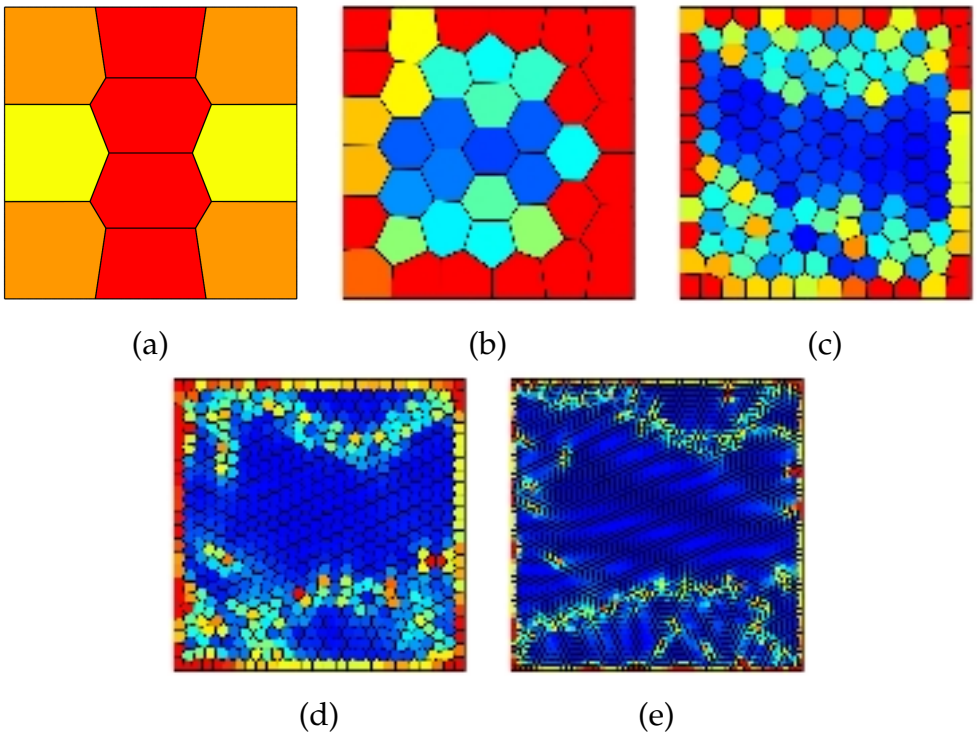


Figure 4.4 – Hierarchical CVT computation. From an initial CVT with $k_0 = 10$ sites (a), successive subdivisions and updates lead to CVTs with $k_1 = 40$, $k_2 = 160$, $k_3 = 640$ and $k_4 = 2560$ sites (from (a) to (e)). The cell regularity measure $G(\Omega_i)$ is colour-coded from blue (regular) to red (far from regular). Note how regular areas grow.

the number of sites has increased, the CVT computation is very fast (see Section 4.2.5 for a discussion).

Once the sites are moved to their new positions and the tessellation is computed, we clip it to the boundary mesh. Our clipping algorithm, detailed in Section 6.1, guarantees that the boundary of the tessellation is a triangulated mesh.

4.1.4 Initialization

Before starting the subdivision - CVT update process, we create a first coarse CVT from the input shape, with a number k_0 of sites. Our aim is to get an initial CVT with as-large-as-possible optimal areas, since our subdivision scheme can only make these areas grow, as explained in Section 4.1.2. We propose two different possible initializations, each of

Shape	n	s	k_0	k_1	$b_1 - k_1$	k_2	$b_2 - k_2$	k_3
			$b_3 - k_3$	k_4	$b_4 - k_4$	k_5	$b_5 - k_5$	
Figure 4.5	1033	4	4	5	11	49	15	233
			25	995	38	/	/	
Figure 4.4	2560	4	10	27	13	145	15	609
			31	2535	25	/	/	
Figure 4.6	10025	5	9	23	16	130	26	568
			58	2373	133	9780	245	

Table 4.1 – Number of sites after each subdivision, and number of sites randomly inserted in boundary cells.

them having different benefits.

A first straightforward idea to initialise the hierarchical CVT computation is to create a CVT using random sampling and a L-BFGS quasi-Newton algorithm to update the positions of the sites. This approach is fast and easy to implement. However, the constructed CVT with k_0 sites may be far regular.

Another idea to create a coarse but regular CVT is to sample the k_0 sites on a optimal lattice (hexagonal lattice in 2D and BCC lattice in 3D) which includes the input shape. The density of the lattice should be chosen so that there are k_0 sites inside the shape. As stated by Quinn et al. [2012], it is very hard to control the number of sites inside the shape because the density depends on both the size and the shape of the input shape. However, tuning the density of the lattice to reach the proper number of sites inside the shape is easier with a small number k_0 of sites than with a large number n sites. When applicable, this leads to an optimal tessellation, except at the boundary of the shape. In addition, most sites do not need to be moved to create a CVT. Thus, this approach is fast and generates regular CVT cells, except on the boundary of the shape.

Both initialization methods are evaluated in the next section. For the random initialization, 10 runs are performed for each test, and the median value is taken as the result.

4.2 Evaluation

We now provide a thorough evaluation of our approach. We first analyse the effect of our hierarchical approach over the regularity of the

generated CVT, by discussing the influence of the two parameters described in Section 4.1: initial number of sites and number of subdivisions. We compare the regularity of 2D and 3D CVTs generated with our hierarchical approach to CVTs computed with previous work (see Sections 2.4.2 and 2.4.4). We also provide some details about computation time. In all figures, CVT cells Ω_i are colour-coded according to the cell regularity measure $G_m(\Omega_i)$ defined in Section 3.3: cells with a high dimensionless second moment are coloured in red, while cells with a low dimensionless second moment are coloured in blue.

4.2.1 Sensitivity to the Initial Number of Sites

As stated in Section 4.1, the idea that drives our hierarchical approach is to first create a large regular area in a coarse tessellation, and then to preserve and if possible widen this area when subdividing. An example is shown in Figure 4.4. The average value and the RMSE of the cell regularity measure $G_2(\Omega_i)$ over all cells Ω_i express that the tessellation becomes more regular over subdivision, see Table 4.2. If n and s are large enough, we expect most of the interior cells of a CVT to be regular, see Figure 4.8 for an example. The max value and the median value do not give very useful information because all the CVTs have some irregular boundary cells and most of the cells are optimal.

CVT	(a)	(b)	(c)	(d)	(e)
Number of sites	10	40	160	640	2560
$G_{2max}(\Omega_i) \times 10^{-2}$	8.894	8.507	8.746	9.038	8.635
$\overline{G}_2(\Omega_i) \times 10^{-2}$	8.458	8.246	8.154	8.099	8.066
$\widetilde{G}_2(\Omega_i) \times 10^{-2}$	8.244	8.231	8.111	8.061	8.041
$G_{2rmse}(\Omega_i) \times 10^{-3}$	5.234	2.556	1.849	1.366	0.883

Table 4.2 – Regularity criteria described in Section 3.3 for CVTs depicted in Figure 4.4.

On the same 2D square example, we create two other CVTs with 1033 and 2560 sites, respectively, using also random sampling initialization and the same number of subdivisions ($s = 4$). We obtain an average cell regularity measure of 8.086×10^{-2} and 8.066×10^{-2} and an RMSE of 1.244×10^{-3} and 0.883×10^{-3}, respectively. This shows that the greater the number n of sites, the smaller the average cell regularity measure.

4.2.2 Sensitivity to the Number of Subdivisions

A CVT with a few big cells is likely to contain less different regular areas than a CVT of the same shape with more, thus smaller, cells. Since our subdivision scheme preserves regular areas, a CVT generated with a large number s of subdivisions is more regular than a CVT with the same number n of sites but generated with a small s, as shown in Table 4.3. As a consequence, we suggest to set s as large as possible.

s	0	1	2	3	4
$G_{2max}(\Omega_i) \times 10^{-2}$	8.820	8.684	8.670	8.892	8.637
$\overline{G}_2(\Omega_i) \times 10^{-2}$	8.079	8.063	8.060	8.056	8.054
$\widetilde{G}_2(\Omega_i) \times 10^{-2}$	8.051	8.041	8.031	8.030	8.030
$G_{2rmse}(\Omega_i) \times 10^{-4}$	9.309	8.146	7.070	6.877	6.875

Table 4.3 – Regularity criteria for CVTs of a square with 10000 sites generated using our hierarchical approach (random sampling initialization) with different number of subdivisions. Thus different numbers of initial sites.

When an optimal lattice sampling is used as initialization, it is preferable to start from a small number of subdivisions, since we only have one large regular area for arbitrary lattice sizes. Actually, $s = 0$ correspond to the optimal lattice sampling, as shown for instance in Figures 4.5 (f) and 4.6 (i). However, as stated before, the larger n, the more difficult it is to build such a lattice with a prescribed number of sites.

4.2.3 Comparison to Previous Work

We test our approach against previously mentioned methods in a simple 2D square. To check which method gives the most regular CVT, we first compute a hexagonal lattice with approximately 1000 sites. As stated above, it is difficult to accurately set the number n of sites. We were able to set $n = 1033$. We then compute CVTs with 1033 sites using the following methods:

- random sampling and L-BFGS update;
- Hammersley sampling [Quinn et al., 2012] and L-BFGS update;
- global Monte-Carlo optimisation [Lu et al., 2012];
- our hierarchical approach with random sampling initialization step (4 subdivisions, which is the maximum possible);

- our hierarchical approach with a lattice sampling initialization step (1 subdivision).

For the global Monte-Carlo optimisation, we have used the parameter values advised in [Lu et al., 2012]. In particular, 200 updates have been performed.

Qualitative results are shown on Figure 4.5. The regularity criteria of the cell regularity measure $G_2(\Omega_i)$ over all cells Ω_i, as well as the CVT energy function values, are given in Table 4.4. Remember that, as explained in Section 3.3, an optimal cell has a dimensionless second moment value of $G_2 = \frac{5}{36\sqrt{3}} = 0.08018\ldots$

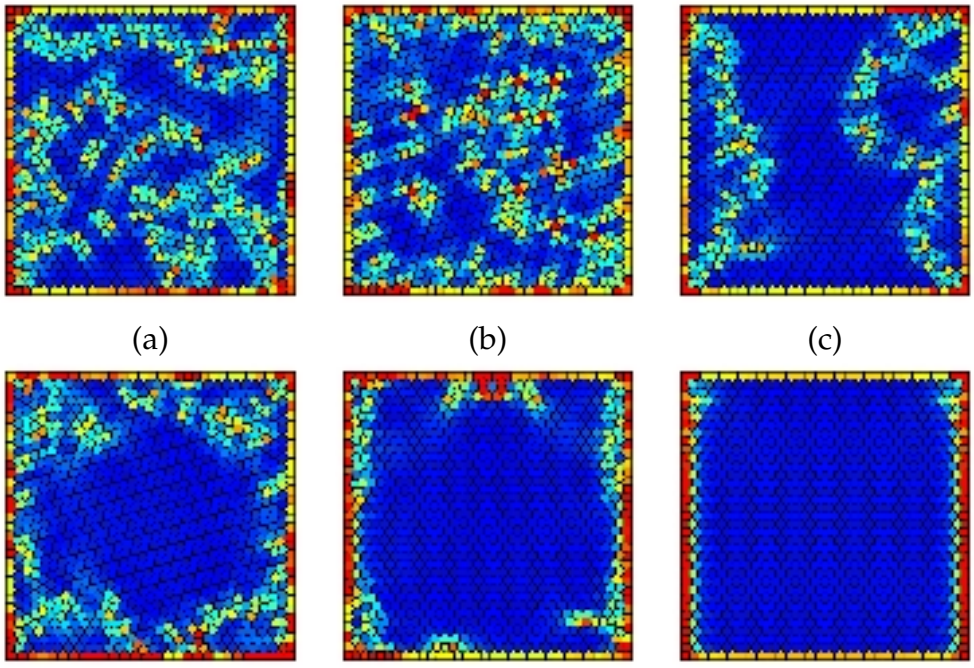


Figure 4.5 – CVTs with 1033 sites. (a) Random sampling + L-BFGS update. (b) Hammersley sampling [Quinn et al., 2012] + L-BFGS update. (c) Global Monte-Carlo [Lu et al., 2012]. (d) Our approach, random sampling initialization. (e) Our approach, lattice sampling initialization. (f) Hexagonal lattice.

The hexagonal lattice is not optimal because of its boundary cells which are not hexagonal. Among other methods, the stochastic approach of Lu et al. [2012] and our hierarchical framework give similar results. The main difference in practice between these two methods is the computation

CVT	(a)	(b)	(c)	(d)	(e)	(f)
$G_{2max}(\Omega_i) \times 10^{-2}$	8.561	8.632	8.534	8.737	8.543	8.772
$\overline{G}_2(\Omega_i) \times 10^{-2}$	8.101	8.107	8.075	8.086	8.077	8.066
$\widetilde{G}_2(\Omega_i) \times 10^{-2}$	8.064	8.067	8.021	8.032	8.020	8.018
$G_{2rmse}(\Omega_i) \times 10^{-3}$	1.160	1.162	1.100	1.131	1.102	1.090
Energy function	25.140	25.160	25.041	25.082	25.045	25.002

Table 4.4 – Regularity criteria measures and CVT energy function value for CVTs depicted in Figure 4.5.

time: the global Monte-Carlo minimisation is about 100 times slower than our approach (207.08 seconds instead of 2.16).

We then test a geometrically more complex shape: a five-branches star. We also increase the number of sites. As in the previous case, we first start by computing an hexagonal lattice with approximately 10000 sites. The lattice contains exactly $n = 10025$ sites. We discard the stochastic approach of Lu et al. [2012] since it is too slow in this case. For the Hammersley sampling, we construct a bounding box of the shape, and try different numbers of sites until we exactly obtain 10025 sites inside the shape. It must be mentioned that several attempts were necessary since the number of sites inside the shape does not necessarily increase when more sites are generated in the bounding box. For the hierarchical approach, we use 5 subdivisions after the random sampling initialization (the maximum possible) and only one after lattice sampling initialization.

CVT	(a)	(c)	(e)	(g)	(i)
$G_{2max}(\Omega_i) \times 10^{-1}$	1.097	1.094	1.094	1.095	1.097
$\overline{G}_2(\Omega_i) \times 10^{-2}$	8.089	8.074	8.068	8.054	8.049
$\widetilde{G}_2(\Omega_i) \times 10^{-2}$	8.053	8.041	8.033	8.021	8.019
$G_{2rmse}(\Omega_i) \times 10^{-3}$	1.192	1.102	1.076	1.012	0.993
Energy function $\times 10^{-1}$	2.234	2.229	2.227	2.222	2.220

Table 4.5 – Regularity criteria measures and CVT energy function value for CVTs depicted in Figure 4.6.

Results are shown on Figure 4.6 and in Table 4.5. These results are in accordance with the results for the square. Our hierarchical approach performs better than the previous initialization methods for all criteria. Moreover, lattice sampling initialization gives better results than random

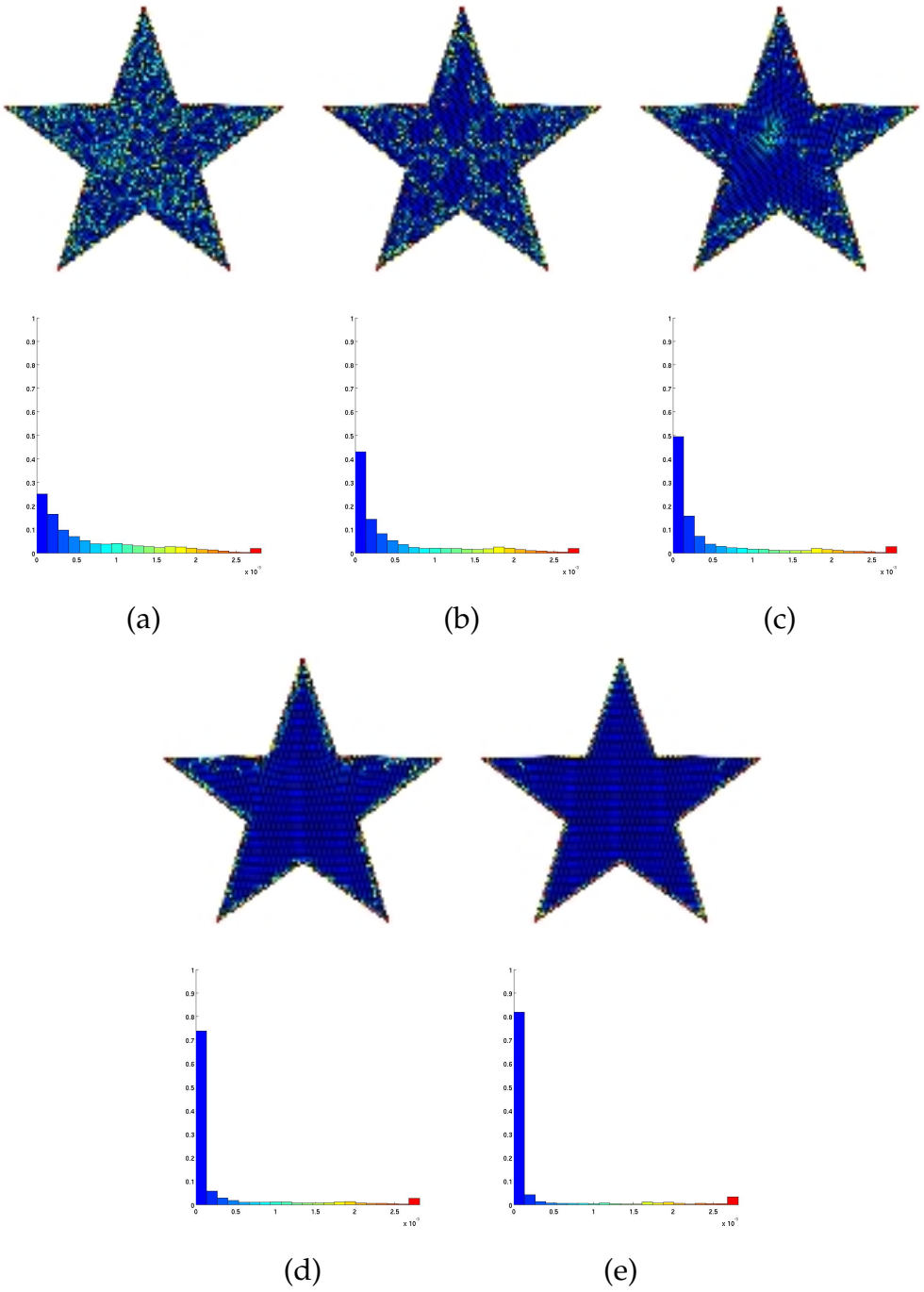


Figure 4.6 – (a,c,e,g,i) CVTs with 10025 sites. (b,d,f,h,j) Corresponding regularity histograms: each bin indicates how many cells share a regularity measure comprised between its boundary values. (a,b) Random sampling + L-BFGS update. (c,d) Hammersley sampling [Quinn et al., 2012] + L-BFGS update. (e,f) Our approach, random sampling initialization. (g,h) Our approach, lattice sampling initialization. (i,j) Hexagonal lattice.

sampling initialization by almost reaching the regularity of a clipped hexagonal lattice.

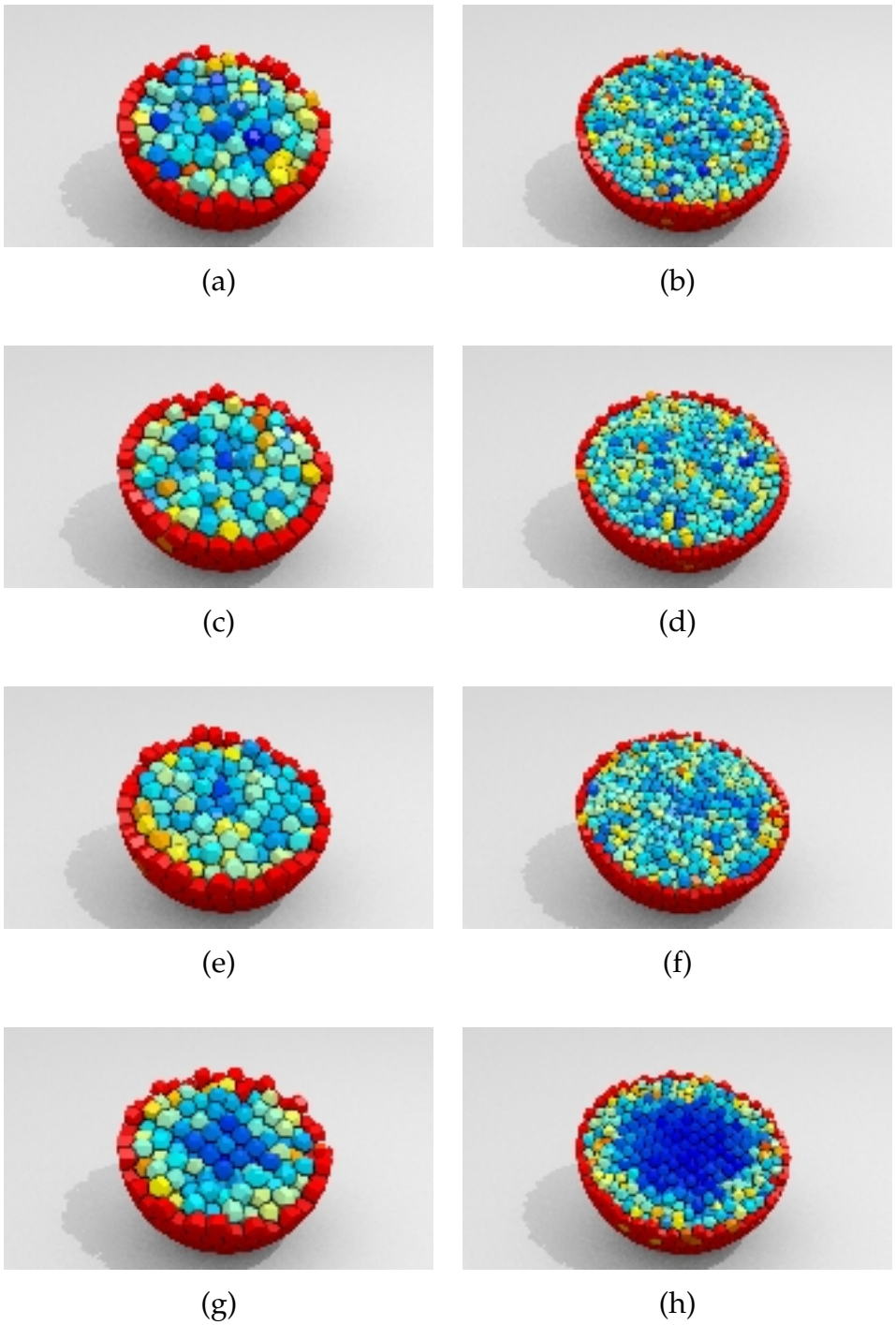
In Figure 4.7 and Table 4.6 we compare CVTs computed in a simple closed 3D ball using random sampling, Hammersley sampling, our hierarchical approach with random sampling initialization and our hierarchical approach with lattice sampling initialization. It was not possible, in this case, to construct a BCC lattice with a prescribed and sufficiently large number of sites. Two different numbers n of sites were tested. We used the maximum number of subdivisions for the hierarchical approach with random sampling initialization: two in the $n = 1000$ sites case and three in the $n = 5000$ sites case.

Remember that in this 3D case, the optimal cell regularity measure value is $G_3 = \frac{19}{192\sqrt[3]{2}} = 0.0785433\ldots$ This example shows that our hierarchical approach performs better than other initialization methods, in case the number n of sites is high enough. In case not, the number of boundary cells is too high with respect to the number of interior cells for the Gersho's conjecture to apply in practice.

CVT	(a)	(c)	(e)	(g)
$G_{3max}(\Omega_i) \times 10^{-2}$	8.436	8.532	8.514	8.479
$\overline{G}_3(\Omega_i) \times 10^{-2}$	8.025	8.025	8.022	8.022
$\widetilde{G}_3(\Omega_i) \times 10^{-2}$	7.979	7.978	7.976	7.976
$G_{3rmse}(\Omega_i) \times 10^{-3}$	1.211	1.211	1.210	1.211
Energy function $\times 10^{-3}$	4.324	4.325	4.324	4.325
CVT	(b)	(d)	(f)	(h)
$G_{3max}(\Omega_i) \times 10^{-2}$	8.430	8.431	8.433	8.435
$\overline{G}_3(\Omega_i) \times 10^{-2}$	7.980	7.976	7.975	7.965
$\widetilde{G}_3(\Omega_i) \times 10^{-2}$	7.947	7.941	7.939	7.928
$G_{3rmse}(\Omega_i) \times 10^{-3}$	1.026	1.026	1.025	1.023
Energy function $\times 10^{-3}$	1.471	1.470	1.470	1.468

Table 4.6 – Regularity criteria measures and CVT energy function value for CVTs depicted in Figure 4.7.

Other 3D CVTs with $50k$, $80k$ and $100k$ sites, computed with a standard random sampling initialization + L-BFGS update, our hierarchical approach using a random sampling initialization and our hierarchical approach with a lattice sampling initialization, are shown in Figure 4.9. It



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

Figure 4.7 – (a,c,e,g) CVTs with 1000 sites in a sphere. (b,d,f,h) CVTs with 5000 sites. (a,b) Random sampling + L-BFGS update. (c,d) Hammersley sampling [Quinn et al., 2012] + L-BFGS update. (e,f) Our approach, random sampling initialization. (g,h) Our approach, lattice sampling initialization.

was not possible to create an Hammersley initialization and a BCC lattice with the correct number of sites inside the shapes in these cases.

4.2.4 Convergence Speed

The main parameter in most CVT computation methods is the number of iterations of the algorithm. A local minimum of the CVT energy function is asymptotically reached, but little is known about how many iterations are necessary before convergence. We have investigated this for the previously described methods, except the global Monte-Carlo minimisation of Lu et al. [2012]. The evolution of both the average of cell regularity measures $\overline{G}_2 = \frac{1}{n}\sum_{i=1}^{n} G_2(\Omega_i)$ and the CVT energy function value with respect to the number of iterations of the CVT L-BFGS update are displayed in Figure 4.10 for the star shape. For our hierarchical approach, this means the number of iterations of the last update (updates for coarser CVTs were done until the usual stopping criterion $\frac{\|g\|}{\|X\|} < 10^{-10}$ is reached, with g the gradient and X the vector of site coordinates, see [Quinn et al., 2012]).

As shown in Figure 4.10, all methods behave the same for both measures. This was expected in our case (see Section 3.4). The hexagonal lattice converges the fastest and to the smallest value. Then our hierarchical approach, combined with a lattice sampling initialization, gives the best results. It is interesting to notice that, because of the non hexagonal boundary cells, no approach reaches the theoretical optimal values ($G_2 = 0.08018\ldots$ and $E_2 = 0.221\ldots$).

4.2.5 Computation Time

Our algorithm is implemented in C++ and we use the CGAL library [CGAL] for 2D constrained and 3D Delaunay triangulations, and the libLBFGS library [Okazaki and Nocedal, 2010] for L-BFGS computation. All computations were performed on an Intel Xeon E5-2643 with 3.30 GHz CPU.

Computation times for our hierarchical approach with random sampling initialization (4 subdivisions) and for a standard method combining a random sampling initialization and L-BFGS updates are shown in Table 4.7. Both methods are computationally equivalent in 2D, but ours is faster in 3D. This can be explained by the fact that in our approach the first CVT

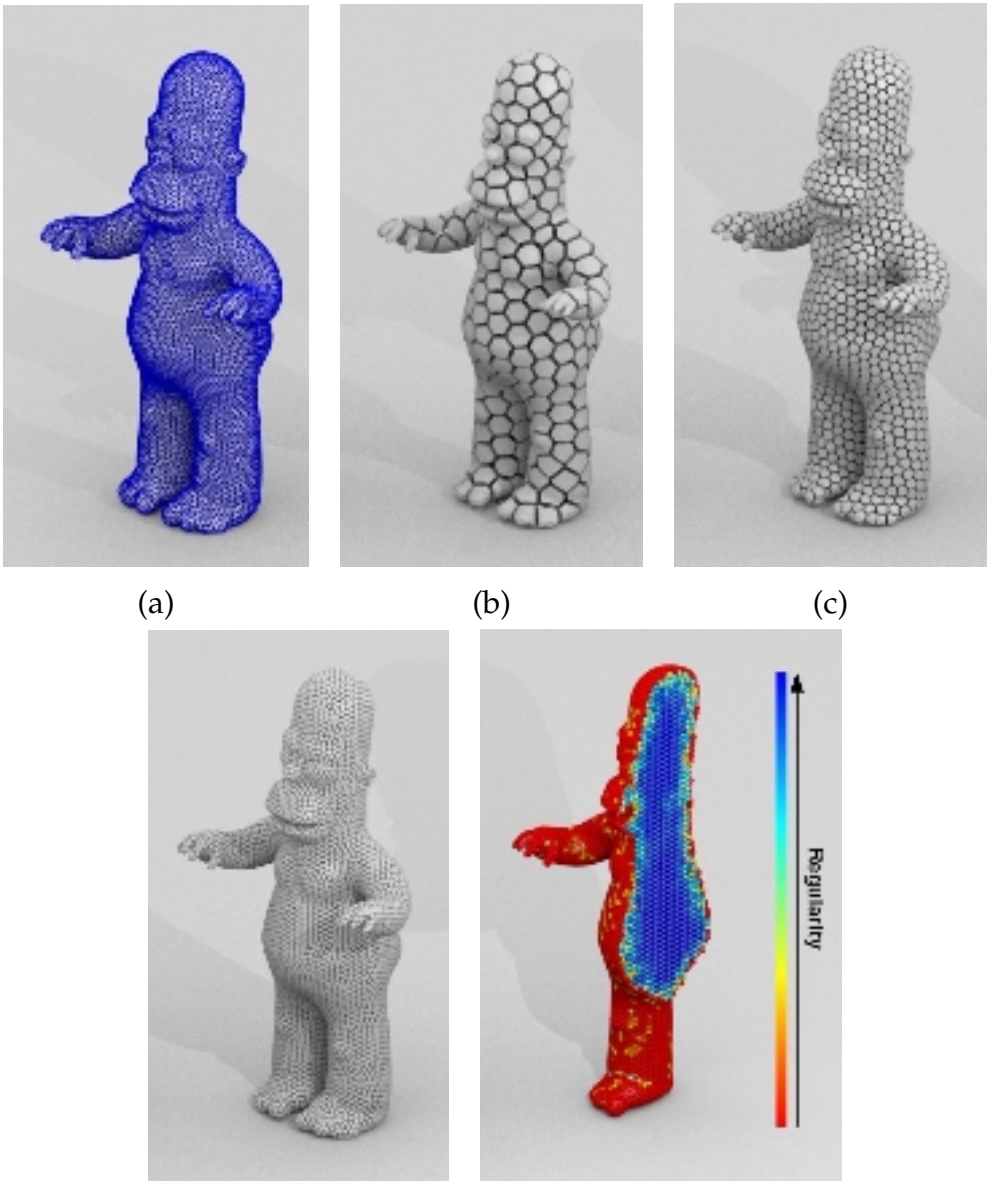


Figure 4.8 – Hierarchical CVT computation in 3D. (a) Input: a 3D shape bounded by a triangulated mesh. (b,c,d) Successive CVTs computed using our approach, with 546, 4375 and 35000 cells respectively. (e) A cut of Homer shows that most of the interior Voronoi cells present high regularity values.

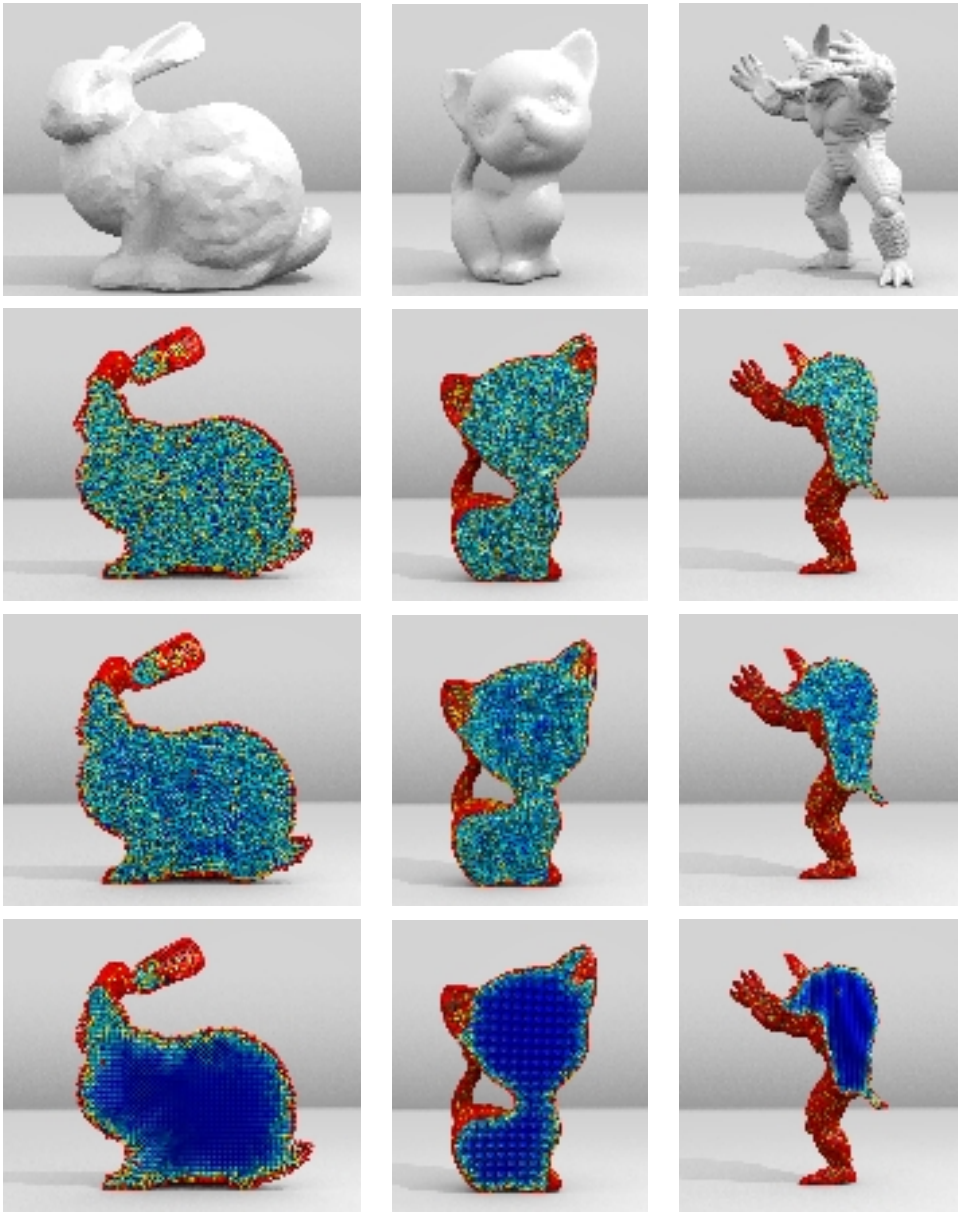
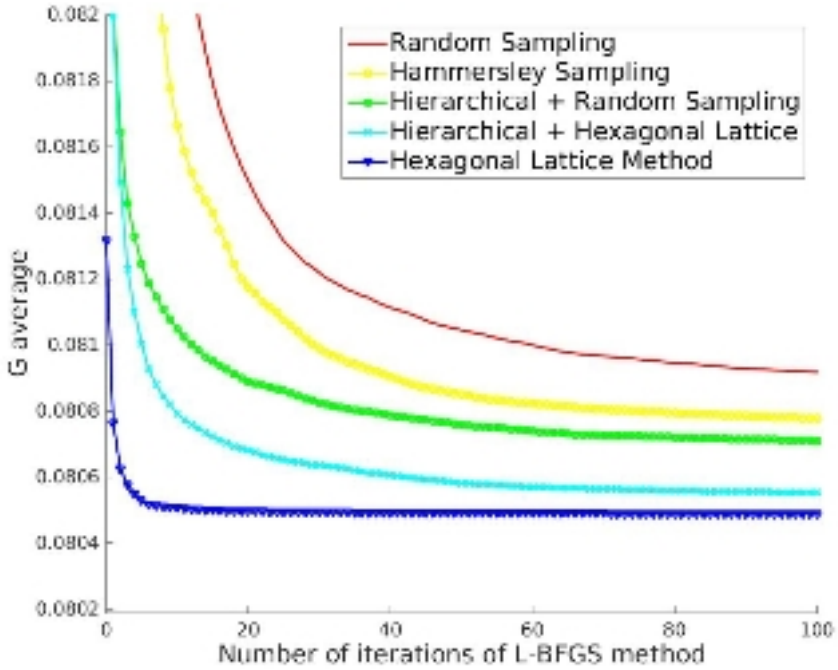
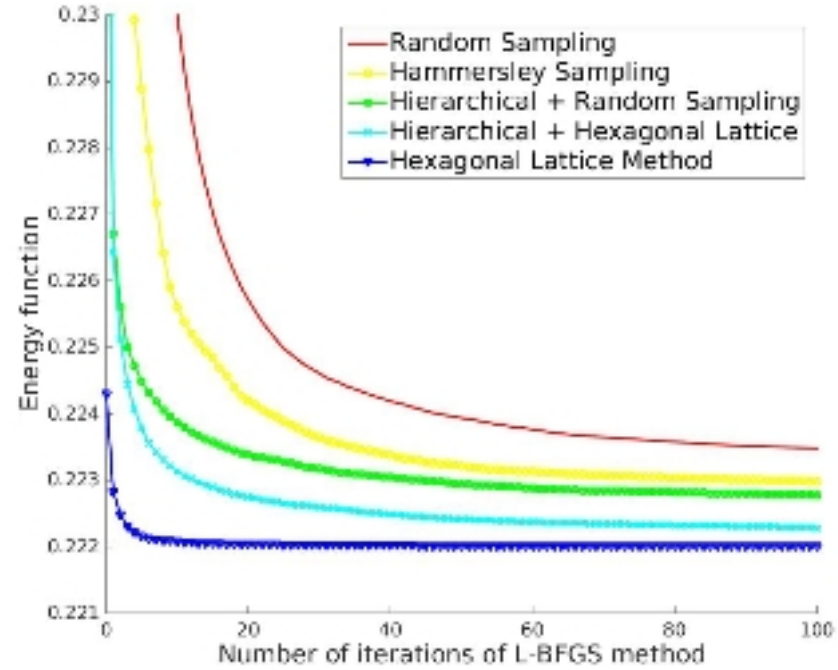


Figure 4.9 – More examples of comparaisons between a standard approach and our hierarchical one. From top to bottom: Input shape, Random sampling + L-BFGS update, Our approach with random sampling initialization, Our approach with lattice initialization.



(a)



(b)

Figure 4.10 – Average cell regularity (a) and CVT energy function value (b) with respect to the number of iterations of the CVT update, for the star shape displayed in Figure 4.6.

is computed with a small number of sites, which is very fast, while the next ones quickly converge since most of the sites do not move much.

Shape	Fig.	Sites	Method	Time (s)
Square	4.5	1000	Standard	2.84
			Hierarchical	2.16
Square	4.5	5000	Standard	10.85
			Hierarchical	9.50
Star	4.6	2000	Standard	3.56
			Hierarchical	4.91
Star	4.6	10000	Standard	26.94
			Hierarchical	26.97
Ball	4.7	1000	Standard	521.86
			Hierarchical	205.82
Ball	4.7	5000	Standard	1484.69
			Hierarchical	665.01
Homer	4.8	50000	Standard	15720
			Hierarchical	7860

Table 4.7 – Computation times for CVTs of shapes depicted in Figures 4.5, 4.6, 4.7 and 4.8.

Other initialization methods (Hammersley sampling and lattice sampling) are usually more time-consuming since finding the right density for a given number n of sites inside the shape is difficult in practice. As stated before, the computation time for the stochastic approach of Liu et al. [2009] depends on the number K of perturbations allowed. For a standard value $K = 200$, we found it to be very time consuming ($207.08s$ in the case of the 2D square with $n = 1000$ sites).

4.3 Discussion

We have proposed a hierarchical approach for generating CVTs with increased regularities. We have performed a thorough evaluation and comparison to previous work. It shows that our approach performs better than the others, both in terms of regularity and CVT energy function. However, since our framework is based on Gersho's conjecture under the

hypothesis that there is no boundary, we need the number of sites large enough to make the boundary effect insignificant. Our approach can be used for CVTs in higher dimensional spaces, combined with an adapted Voronoi clipping algorithm such as [Lévy, 2014], although Gersho's conjecture has not been proven in such cases. In future work, it would be interesting to consider extensions of the approach to generalised CVTs such as for instance weighted diagrams, power diagrams or L_p CVTs [Lévy and Liu, 2010].

Chapter 5

Voronoi-based Approximated Volumetric Tessellation

Contents

In this chapter, we introduce our CVT algorithm for implicit shapes and provide an extensive comparison of voxel-, Delaunay- and Voronoi-based strategies to produce volumetric tessellations. Because of the properties of CVT and our local improvement for the boundary cells reconstruction, our algorithm performs better than the two others in terms of accuracy and regularity. The remainder of this chapter is organized as follows. In Section 5.1, we review related work on evaluation of volumetric tessellations. We explain our algorithm in details and compare it with voxel- and Delaunay based strategies in Sections 5.2 and 5.3, respectively.

5.1 Context

5.1.1 Eulerian and Lagrangian Strategies

The main Eulerian and Lagrangian strategies such as the MC algorithm and the Delaunay triangulation have been explained in Chapter 2;

5.1.2 Surface Reconstruction

We focus in this chapter on methods for the polyhedrization of implicit forms however it is worth mentioning approaches that reconstruct the zero-set surface of an implicit form, see *e.g.* [de Araujo et al., 2015] for a review. Approaches in this category focus on surface tessellation where we consider volumetric tessellations.

5.1.3 Evaluation

In the literature, 2D and 3D shape reconstruction techniques are mostly evaluated according to the *quality* of the constructed cells. This is indeed crucial for applications such as Finite Element Modeling. Quality is usually defined with respect to the shape of the cell [Field, 2000]. In some cases the quality metrics reflect the fact that the cell should stay away from degenerate configurations. For instance, tetrahedra with at least one small angle are usually to be avoided [Cheng et al., 2000]. In other cases, an ideal shape is defined, and the quality metrics are defined as distance to this ideal. This is the case with CVTs, for which the ideal cell shape is known to be a truncated octahedron as mentioned in Section 3.3. We have introduced in Chapter 3, the dimensionless second moment of a polytope is introduced and can be used as a measure of the regularity of a CVT cell. Metrics have also been proposed for other types of cells, such as hexahedra [Field, 2000], as well as algebraically for general cells [Knupp, 2001].

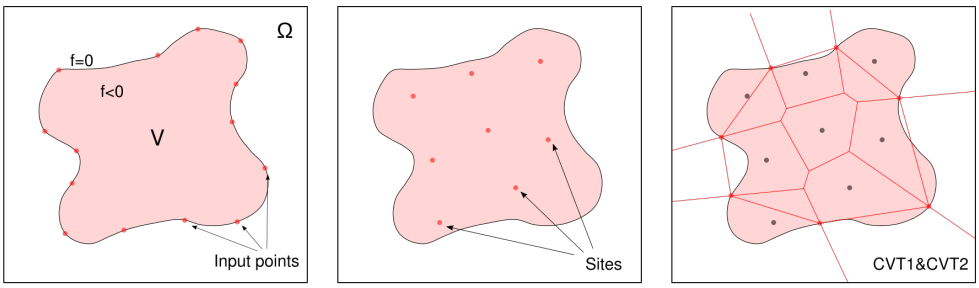
A few works have investigated the geometric accuracy of a given reconstruction. Geometric accuracy can be defined as a distance between the boundaries of the given input form and of the volumetric reconstruction obtained. *Metro* [Cignoni et al., 1998b], introduced in the context of mesh simplification, is a common tool to evaluate the distance between two triangulated surfaces. More recently and focusing on surface reconstruction, Berger et al. [2013] have introduced new metrics based on discrete differential geometry concepts in order to quantitatively evaluate distances between an implicit surface and a surface mesh. We build on this work for accuracy evaluation.

In Section 5.3, we compare the shape tessellations obtained with Marching Cubes, Delaunay refinement and CVT approaches using both shape quality and geometric accuracy criteria. We also discuss the theoretical guarantees given by each approach, as well as their computation times.

5.2 Algorithm

Approaches exist that compute CVTs given explicit forms for shapes, usually meshes as in [Yan et al., 2013] [Lévy, 2014] [Wang et al., 2016a]. We consider here the case of shapes defined by implicit forms in 3D and also, more specifically, implicit forms obtained from point clouds, a frequent case when modeling shapes with visual observations.

5.2.1 Algorithm overview



Implicit form | Initialization | Clipping (1)

Clipping (2) | Clipping (3) | Optimization

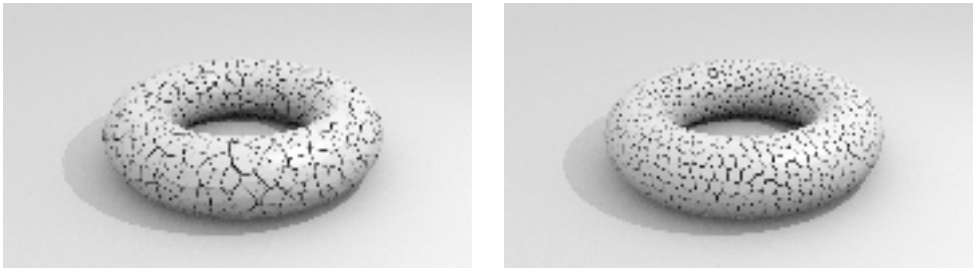
Figure 5.1 – The different steps of the CVT algorithm. The clipping and optimization steps are iterated until the sites are stabilized.

Our CVT algorithm considers as input an implicit function $f : \Omega \rightarrow \mathbb{R}$ defined over a domain $\Omega \in \mathbb{R}^3$ and such that $f(x) = 0$ on the boundary surface $\mathcal{S}$ of a shape $\mathcal{V}$. In the case of an indicator function, f has zero values outside $\mathcal{V}$ and the value 1 inside. The algorithm takes also as input the number of sites-cells n and follows the traditional CVT scheme below:

1. Initialization: find initial positions for the n sites inside $\mathcal{V}$.

2. Clipping: compute the Voronoi tessellation of the sites, then restrict it to $\mathcal{V}$ by computing its intersection with $\mathcal{S}$.
3. Optimization: update the position of the sites by minimizing the CVT energy function.

Steps 2 and 3 are iterated several times where the number of iterations is a user-defined parameter. Any initialization can, in principle, be applied here. In our experiments, the sites are randomly positioned inside $\mathcal{V}$. At the first iteration, the cells of the clipped Voronoi tessellation constructed after step 2 are not uniform nor regular (see Figure 5.2 (a)) and do not minimize the CVT energy. Step 3 optimizes therefore the site locations in order to minimize E. In the literature, the two main strategies for such minimization are Lloyd's gradient descent method and the L-BFGS quasi-Newton method. In our approach, we choose the latter since it is known to be faster [Liu et al., 2009]. As shown in Figure 5.2 (b), once convergence is reached, the clipped Voronoi cells are almost uniform and regularly spaced. Besides, they yield a better approximation of the shape $\mathcal{V}$, as shown in Section 5.3.



(a) (b)

Figure 5.2 – Voronoi tessellations of a torus with and without optimization. (a) A clipped Voronoi tessellation with random initial positions for the sites. (b) Clipped CVT after optimization.

5.2.2 Clipping

In order to clip a Voronoi tessellation with the implicit function f describing the shape $\mathcal{V}$, we introduce an algorithm that consists of the following main steps (see Figure 5.1):

1. Given the unbounded Voronoi tessellation $\bigcup_i \Omega_i$ of the sites $\{x_i\}$, identify which cells Ω_i intersect the implicit surface $\mathcal{S}$ bounding $\mathcal{V}$.

2. For each of these boundary cells Ω_j,
   a) Compute the intersection between the edges of Ω_j and $\mathcal{S}$, this intersection being represented as a set of points P_j (red dots in Figure 5.1 Clipping (1)).
   b) Build the boundary clipped Voronoi cell Ω'_j as the convex hull of the intersection points in P_j and the vertices of Ω_j that are inside $\mathcal{V}$ (see Figure 5.1 Clipping (1)).
   c) (CVT-2 only) Add to each boundary cell the point at the intersection between $\mathcal{S}$ and the ray along the normal to Ω'_j (red dots in Figure 5.1 Clipping (2)). Rebuild Ω'_j by connecting it with other intersection points of P_j (see Figure 5.1 Clipping (3)).

Step 1 is carried out by first converting infinite Voronoi cells into finite cells using a bounding surface around $\mathcal{V}$ and second by detecting boundary cells as cells with at least one vertex outside $\mathcal{V}$. Step 2 is discussed below.

5.2.2.1 Intersections of Ω_j With $\mathcal{S}$

Boundary cells are convex polytopes composed of bounded polygons and segments. We first interpolate f values along segments to find their intersections with f. To this purpose, several strategies can be considered depending on the information available on f. When both function values and derivatives are available, Hermite interpolation can be used, as advocated in [Fuhrmann et al., 2015]. It provides fast and accurate interpolated values as long as the local approximation is valid. When only function values are available, linear interpolation, with for instance the false position algorithm, can be used to iteratively locate the intersection. In any cases, the bisection method can be applied. It is slower than the previous strategies but more robust. A combination of the bisection and Hermite methods can also be considered to first reduce the search space so that the Hermite approximation becomes more valid. Note that polyhedrization approaches are independent of the interpolation scheme and can all consider any of them. For the purpose of evaluation, and without loss of generality, we use the bisection method with all approaches in the comparisons presented in the evaluation section 5.3.

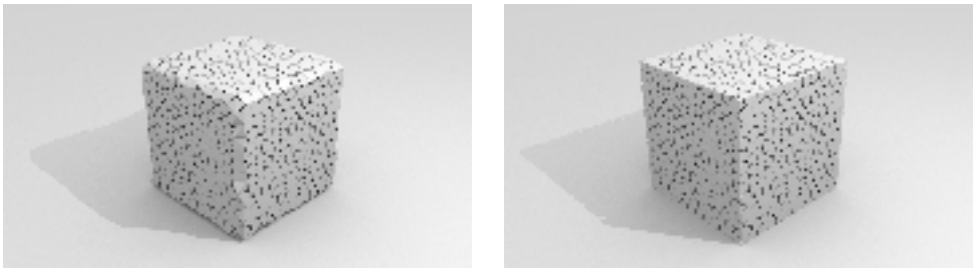
5.2.2.2 Convex Hull

Once the edge intersection points are determined, the clipped Voronoi cell is computed as the convex hull of the intersection points and the

cell vertices inside the surface. Since Voronoi cells are convex, this is guaranteed to provide a boundary surface with a correct topology (see Figure 5.1 Clipping (1)).

5.2.2.3 Sharp Features

Sharp features, in case they occur, are not preserved by default by the clipping algorithm CVT1 (see Figure 5.3 (a) for instance). This also true for Marching Cubes and Delaunay strategies for which specific solutions have been proposed, *e.g.* [Kobbelt et al., 2001] and [Cheng et al., 2010] respectively. CVT easily adapts to sharp features when identified as a list of points that can be simply assigned to their closest sites before the convex hull computation. This is a nice feature of the CVT strategy that is flexible and allows for such additional points without significantly increasing the complexity (only slightly modifying the convex hull computation) and while keeping the topological guarantees. This would be difficult to implement with the Marching Cubes or Delaunay strategies without fundamentally modifying the associated algorithms. Figure 5.3 (b) shows an example of such a reconstruction.



(a) (b)

Figure 5.3 – Tessellations of the characteristic function of a cube: (a) CVT1 algorithm without additional points. (b) CVT1 algorithm with sharp feature points added.

5.3 Evaluation

In order to compare the different strategies for shape modeling mentioned different criteria can be considered. As emphasized in [Meyer et al., 2007] for a similar evaluation, a given method will easily favor one of these criteria at the cost of the others, depending on the targeted application. In this work, our target is the reconstruction from implicit functions.

This includes theoretical guarantees, the accuracy of the approximation with respect to the input implicit boundary surface, the quality of the resulting cells, and the time complexity.

5.3.1 Methodology

5.3.1.1 Methods

The four different methods we compare are the following:

MC: an implementation of the Marching Cubes algorithm with topological guarantees [Lewiner et al., 2003];

CGAL: the CGAL [CGAL] implementation of the Delaunay tetrahedrization based algorithm, which uses the Delaunay refinement technique followed by mesh optimization to remove degenerated tetrahedra [Jamin et al., 2015];

CVT1: a first proposed implementation of the clipped CVT algorithm, *i.e.* only Voronoi edge intersections with the surface are considered as surface points (see Section 5.2.2);

CVT2: an extension of CVT1 where an additional point which is the intersection between the surface and a ray estimated by boundary clipped Voronoi cell is added. (see Section 5.2.2).

To compare shape tessellations on a fair basis, similar resolutions, *i.e.* cell numbers, are required. While the resolution is easily imposed with CVT, which can be an advantage when similar shape discretizations are required, it is less easy with MC and space discretizations. In practice, we first compute the Marching Cubes tessellation. We then use $\sqrt{3}/2$ times the length of a cube as the targeted radius of a tetrahedron's circumsphere for the Delaunay tetrahedrization approach, in order for the size of this circumsphere to be similar to the size of the cube's circumsphere. For the CVT approaches, we simply sample randomly as many sites as the number of cubes inside the shape.

5.3.1.2 Data

Methods are first evaluated on a set of 97 object meshes, from the Princeton Segmentation Benchmark [Chen et al., 2009], for statistical comparisons on the accuracy (see Figure 5.7). We next consider point clouds obtained from vision reconstructions and from which implicit forms are built. The DANCER (Figure 5.4) was obtained with a multiview system followed by a Poisson implicit function estimation [CGAL]. GARGOYLE (Figure 5.5) was obtained with [Berger et al., 2013] and the three other shapes, KNEELINGLADY, AQUARIUS and SKULL (Figure 5.6), were obtained with [Furukawa and Ponce, 2010] followed by the same Poisson estimation [CGAL] (other implicit function estimation could be considered). DANCER's MC results are extracted from 50 x 50 x 50 and 100 x 100 x 100 voxel grids for different resolution tests. Other MC results are from 100 x 100 x 100 voxel grids.

5.3.2 Theoretical Guarantees

Two theoretical guarantees are in practice often required: the manifoldness of the output volumetric mesh and the topological correctness with respect to the input form. A k-manifold is a k-dimensional object which is locally homeomorphic to a k-dimensional disk. A k-manifold allows for non ambiguous definitions of geometrical quantities such as the local geodesic neighborhood of any point, which is critical in many shape processing applications. It allows for instance to smooth and deform shapes in a consistent way. Topological correctness is the fact that input and output shapes present the same topology, for example the same number of components. This is an important property when considering properties over sets of shapes.

The original Marching Cubes algorithm [Cubes, 1987] is known to produce surface meshes which can be non-manifold and which topology can differ from the input implicit surface. Many methods have been proposed to solve for non-manifoldness and topological ambiguities, see [Newman and Yi, 2006] for a survey. 2D Delaunay triangulations are known to be, under some sampling assumptions, good geometrical and topological approximations of the input shape [Boissonnat and Oudot, 2005]. This has been used for instance in order to robustly model surfaces evolving over time [Pons and Boissonnat, 2007]. However, this is not the case in higher dimensions [Oudot, 2008]. Thus, an additional post-processing step, such as [Lhuillier, 2015], is required to guarantee manifoldness in the case of Delaunay tetrahedrizations. In contrast, a 3D CVT is manifold

by construction: it is composed of convex 3D cells and on its surface every edge, either on cell boundaries or within cell, is shared by exactly 2 faces (up to numerical errors in the interpolation of the implicit function along edges), because this edge belongs to a bisector of CVT.

5.3.3 Accuracy

The accuracy measures how close the estimated shape approximation is to the implicit form. To this aim we compare shape surfaces. The geometric similarity between two surfaces can be defined in several ways. Following the evaluation in [Berger et al., 2013], we consider distances between shapes in both directions and the following metrics:

Dmean: the mean of distances between the input shape and the reconstructed surface in both directions;

Drmse: the root-mean-square error (RMSE) of these distances;

Nmean: the mean angle deviation;

Nrmse: the RMSE angle deviation.

Distances are estimated at points regularly distributed on both shapes and by searching for the closest point on the other shape in the facet normal directions. The same principle applies for normal angles that are estimated between closest points in the evaluation sets on both the input and the reconstructed surfaces. In order to build regular evaluation point sets, we use a particle system to sample implicit surfaces [Berger et al., 2013], and we compute 2D CVTs on the reconstructed boundary surface to generate sample points regularly distributed on both surfaces. Using a particle system and 2D CVTs to regularly distribute evaluation points on the surface ensures that we do not estimate distances between two discretizations of the input implicit surface at different scales, as can be the case with *Metro* [Cignoni et al., 1998b].

5.3.4 Shape Quality

Besides the accuracy of the approximation, tessellations can also be compared with respect to the cell shape properties. Compactness, that

ensures regularity, is for instance desirable for, *e.g.*, local discrete operations on shapes such as deformations or quantification optimality. We first assess cell regularity for the Marching Cubes and CVT approaches (Delaunay tetrahedra are not compact by construction). For Delaunay tetrahedrizations, we compare them to CVTs using the dual tetrahedrizations of CVTs and boundary triangle quality metrics. Dual tetrahedrizations are computed by projecting the sites of the boundary Voronoi cells on the surface and then optimizing their position as in [Yan et al., 2013].

5.3.4.1 Cell Regularity

Marching Cubes-generated tessellations are mostly made of cubes, however the boundary cells may be very irregular. In order to assess cell regularity, we use the criterion G_3 defined in Chapter 3.

5.3.4.2 Tetrahedron Quality.

We use four standard quality metrics for tetrahedra [Yan et al., 2013]:

VQ_1: minimum dihedral angle of the tetrahedron;

VQ_2: maximum dihedral angle;

VQ_3: radius-ratio, defined as $\frac{3r_i}{r_c}$, where r_i and r_c are the radii of the inscribed and circumscribed spheres, respectively;

VQ_4: meshing quality, defined as $\frac{12\sqrt[3]{9V^2}}{\sum l_{i,j}^2}$, where V is the volume of the tetrahedron and $l_{i,j}$ is the length of an edge i, j of the tetrahedron.

5.3.4.3 Boundary Triangle Quality.

Four similar criteria are used for boundary triangles:

SQ_1: minimum angle of the triangle;

SQ_2: maximum angle;

SQ_3: radius-ratio, defined as $\frac{2r_i}{r_c}$, where r_i and r_c are the radii of the inscribed and circumscribed circles, respectively;

SQ_4: meshing quality, defined as $\frac{4\sqrt{3S}}{\sum l_{i,j}^2}$ where S is the area of the triangle and $l_{i,j}$ is the length of an edge i, j of the triangle.

5.3.5 Results and Discussion

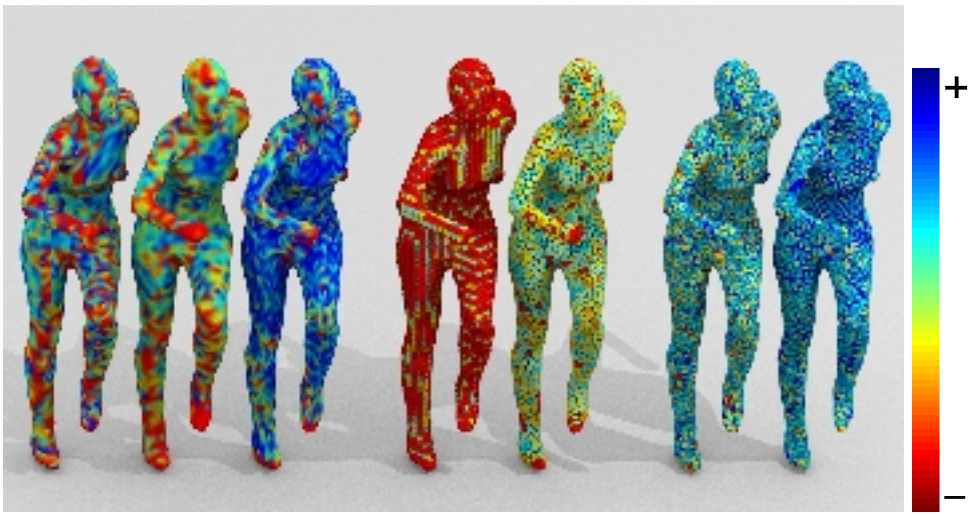
5.3.5.1 Accuracy

We first compare methods on a set of various shapes: 97 meshes taken from the Princeton Benchmark [Chen et al., 2009]. These meshes belong to 5 different categories and the algorithms have been run with a fixed resolution for a given category, which can however vary over categories: 50^3, 80^3 and 100^3 for the Marching Cubes. The number of sites for CVT is taken as the number of inner cubes with MC, this for each mesh. Results in Figure 5.7 show method rankings with respect to the accuracy criteria defined previously. It demonstrates that CVT approaches are statistically significantly better than MC and Delaunay.

Quantitative accuracy have also been conducted on the mentioned vision datasets (see Table 5.1). Additionally, qualitative results are color-coded in Figure 5.5 and Figure 5.4. In these experiments, 10 iterations are performed for both Delaunay tetrahedrization and CVT approaches. CGAL failed to compute a tetrahedrization on the GARGOYLE and SKULL datasets (the process was stopped after 18 hours of computation). These results show that CVT2 performs better than other approaches on all our experiments. On point datasets that describe smooth shapes (DANCER and GARGOYLE), CVT1 performs better than Marching Cubes even without the optimization step.

5.3.5.2 Shape quality.

Table 5.1 shows the maximum *Gmax* and the mean *Gmean* values of the cell regularity $G_3(\Omega)$ over all cells of Marching Cubes and CVTs tessellations. Qualitative results are also color-coded in Figure 5.6. Not surprisingly, some cells of the Marching Cubes are far from regularity, which also affects the mean and median cell regularities. In contrast CVT cells are more regular. These results also show the benefit of optimizing site positions to generate more regular cells. The mean tetrahedron and



(MC) (Del) (CVT1) (MC) (CVT1) (Del) (CVT1)

Figure 5.4 – Accuracy (left), cell regularity (middle) and tetrahedron quality (right) of 3D implicit form tessellations with MC, Delaunay refinement (Del) and CVT1.

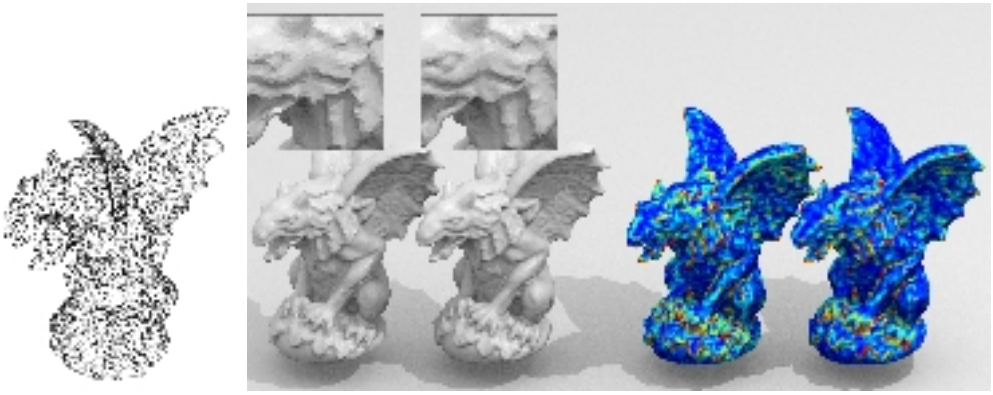


Figure 5.5 – The Gargoyle multi-view point cloud and the associated Poisson reconstructions with Marching Cubes and CVT. Distances to the implicit form are color encoded on the right, from low (blue) to high (red).

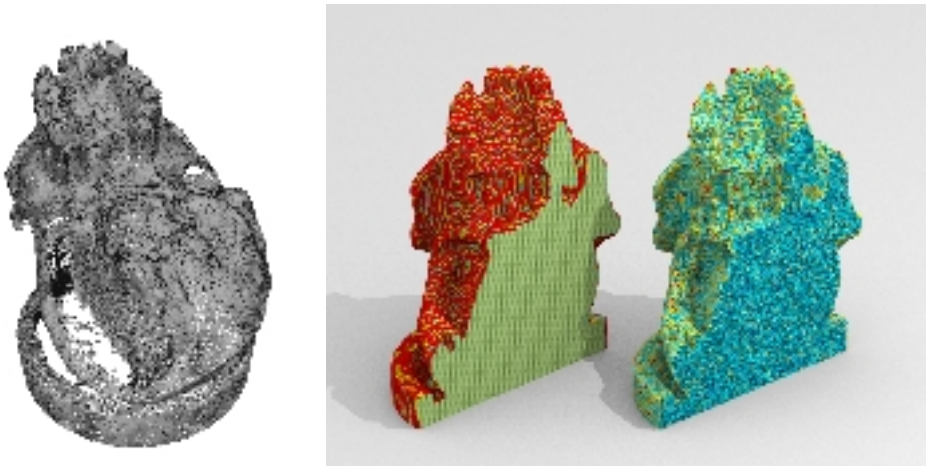


Figure 5.6 – SKULL: Input point cloud (left) and cell regularity for Marching Cubes and CVT1.

boundary triangle quality measures are given in Table 5.2 and Table 5.3, respectively. According to our experiments, the CVT approach gives better results than the Delaunay tetrahedrization for all of them, which is consistent with the results shown in [Yan et al., 2013].

5.3.5.3 Computation times.

Timings are shown in Table 5.4. CGAL fails to compute a tetrahedrization for the GARGOYLE and the SKULL datasets. The Marching Cubes algorithm is the fastest of the four tested methods, as a result of the increased complexity of Delaunay and Voronoi tessellation computations. The CVT approach performs anyway always faster than the Delaunay tetrahedrization.

5.4 Conclusion

We have presented a comparison of different strategies to convert an implicit form in 3D into an explicit shape representation. This includes a new strategy that uses Centroidal Voronoi Tessellations to build a 3D mesh composed of convex Voronoi cells around a pre-defined number of sites. The evaluation shows that CVTs provide the most accurate and the most regular shape tessellations when compared to Marching Cubes and Delaunay refinement strategies. As such, CVTs are a good alternative to

Dataset (nb of sites)	Method	Dmean	Drmse	Nmean	Nrmse	Gmax	Gmean	CVT2
DANCER LOW (2365)	MC	27.16	29.72	1.31	1.41	3210	19.30	
	CGAL (10 it.)	27.12	29.72	1.30	1.41	-	-	
	CVT1 (0 it.)	27.14	29.67	1.31	1.41	1647	11.00	
	CVT1 (10 it.)	27.13	29.66	1.30	1.41	11.84	8.18	
	CVT2 (10 it.)	27.05	29.64	1.30	1.41	11.84	8.15	
DANCER (14474)	MC	22.10	25.46	1.14	1.25	5999	14.49	
	CGAL (10 it.)	21.72	25.11	1.14	1.25	-	-	
	CVT1 (0 it.)	21.86	25.41	1.13	1.25	75.85	9.75	
	CVT1 (10 it.)	21.76	25.25	1.12	1.24	9.82	8.07	
	CVT2 (10 it.)	21.16	25.29	1.12	1.24	9.82	8.03	
AQUARIUS (64588)	MC	21.64	23.99	1.32	1.42	29825	12.17	
	CGAL (10 it.)	21.65	23.99	1.32	1.42	-	-	
	CVT1 (10 it.)	21.83	24.14	1.33	1.42	13.17	8.16	
	CVT2 (10 it.)	21.59	23.94	1.31	1.41	13.02	8.15	
KNEELING LADY (73150)	MC	2.55	3.98	0.40	0.63	5870	11.80	
	CGAL (10 it.)	2.60	3.83	0.40	0.63	-	-	
	CVT1 (10 it.)	2.58	4.00	0.40	0.63	19.05	8.16	
	CVT2 (10 it.)	2.50	3.93	0.39	0.62	19.05	8.15	
GARGOYLE (106497)	MC	0.38	0.54	0.20	0.29	54467	12.65	
	CVT1 (10 it.)	0.38	0.54	0.19	0.28	10.20	8.02	
	CVT2 (10 it.)	0.37	0.53	0.19	0.28	10.19	8.00	
SKULL (225034)	MC	7.12	9.76	0.82	1.00	82716	11.26	
	CVT1 (10 it.)	7.15	9.81	0.83	1.00	17.87	8.07	
	CVT2 (10 it.)	7.07	9.67	0.82	1.00	17.56	8.06	

Table 5.1 – Accuracy and regularity results on the datasets. (Dmean, Drmse, Gmax, Gmean) $\times 10^{-2}$.

Dataset	Method	$\overline{VQ_1}$	$\overline{VQ_2}$	$\overline{VQ_3}$	$\overline{VQ_4}$
DANCER	CGAL (10 it.)	50.75	96.48	0.84	0.90
	CVT1 (10 it.)	**51.32**	**95.49**	**0.85**	**0.91**
AQUARIUS	CGAL (10 it.)	51.35	95.89	0.83	0.90
	CVT1 (10 it.)	**51.55**	**95.33**	**0.84**	**0.92**
KNEELING LADY	CGAL (10 it.)	51.54	95.71	0.83	0.90
	CVT1 (10 it.)	**51.63**	**95.16**	**0.84**	**0.92**

Table 5.2 – Mean tetrahedron quality measures over all cells of the tessellation (best result in bold).

Dataset	Method	$\overline{SQ_1}$	$\overline{SQ_2}$	$\overline{SQ_3}$	$\overline{SQ_4}$
DANCER	CGAL (10 it.)	47.92	75.13	0.93	0.94
	CVT1 (10 it.)	**52.56**	**69.23**	**0.97**	**0.97**
AQUARIUS	CGAL (10 it.)	47.73	75.37	0.92	0.94
	CVT1 (10 it.)	**52.56**	**69.23**	**0.97**	**0.97**
KNEELING LADY	CGAL (10 it.)	48.32	74.59	0.93	0.94
	CVT1 (10 it.)	**50.95**	**71.27**	**0.96**	**0.96**

Table 5.3 – Mean triangle quality measures over all boundary triangle of the tessellation (best result in bold).

Method	DANCER LOW	DANCER	AQUARIUS	KNEELING LADY	GARGOYLE	SKULL
MC	0.1	0.6	1.6	1.1	1.3	1.9
CGAL (10 it.)	1036	1052	162	198	-	-
CVT1 (0 it.)	0.4	1.4	7.2	7.4	9.1	18
CVT1 (10 it.)	4.9	25	135	123	160	342
CVT2 (10 it.)	5.1	25	136	125	165	363

Table 5.4 – Computational time (s) for each experiment.

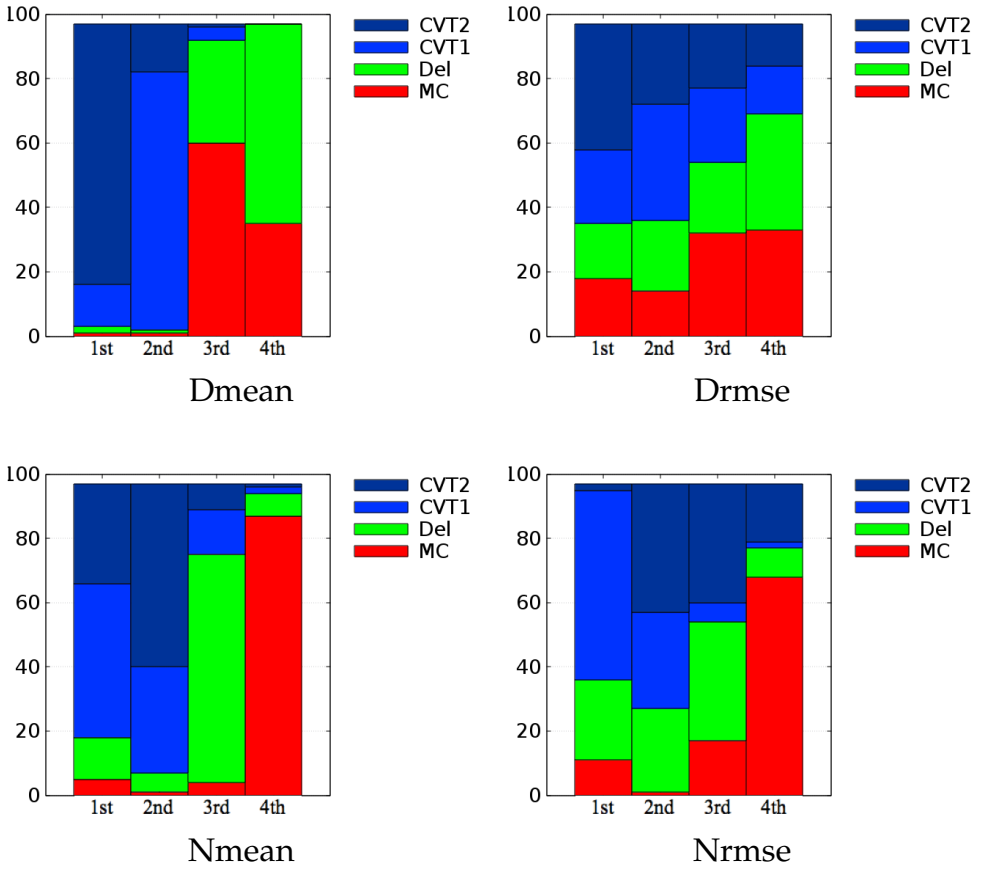


Figure 5.7 – Accuracy rankings on 100 meshes from the Princeton Benchmark [Chen et al., 2009]. Implicit forms were obtained using Poisson reconstructions [CGAL] and accuracies measured on samples obtained using the particle system approach [Berger et al., 2013].

Marching cubes, in particular when modeling dynamic scenes for which accuracy, regularity and invariance to rigid transformation are desirable properties of shape tessellation. Delaunay refinement, while providing boundary surfaces with *good* triangles does not outperform CVTs nor Marching cubes in any case. Finally, Marching cubes is always the fastest strategy, hence a solution for applications requiring fast solutions, though less accurate and regular than CVTs.

Chapter 6

Applications

Contents

In this chapter, we introduce applications of CVT. The intersection between a shape $\mathcal{V}$ and a Voronoi tessellation (of space) is called a clipped Voronoi tessellation of $\mathcal{V}$ which is an important step for computing a CVT. In Section 6.1, we propose a novel clipping algorithm for computing clipped Voronoi tessellations which is simple, robust and efficient. In Section 6.2, we propose a pipeline for shape animation from captured shape motion data using different applications of CVT which include 3D volumetric reconstruction, tracking, morphing and physical simulations.

6.1 Clipped Voronoi Tessellation

6.1.1 Introduction

Computing a clipped Voronoi tessellation of an arbitrary 3D shape, usually described by its meshed boundary surface, is not an easy problem. Yan et al. [2010] have proposed an algorithm to compute clipped Voronoi diagrams of 3D shapes described by tetrahedral meshes. This algorithm consists of two main steps: detection of boundary sites by computing surface restricted Voronoi diagram [Edelsbrunner and Shah, 1994] [Yan et al., 2009] and computation of the intersection between the Voronoi cells of boundary sites and the input tetrahedral mesh using Sutherland's clip-

ping algorithm [Sutherland and Hodgman, 1974]. Recently, Lévy [2014] proposed another efficient method based on iterative convex clipping. This method expresses the clipping problem as a 3D volume intersection problem but also requires a tetrahedral mesh as input. When the input 3D shape is given as a closed triangle mesh, a 3D constraint Delaunay triangulation must be computed first [Shewchuk, 1998a] [Shewchuk, 2008]. This is a complex problem which has many degenerate cases and usually requires additional (Steiner) points to ensure the existence of a solution. The complexity highly depends on the quality of the input surface triangle mesh [Erickson, 2001]. Inspired by [Zaharescu et al., 2011], we overcome this problem and propose an algorithm that exploits a 2D constrained Delaunay triangulation to determine triangles on the input mesh that intersect a given Voronoi cell, without the need of a tetrahedral mesh inside the shape.

6.1.2 Algorithm

Our algorithm first triangulates the polygonal boundaries of the Voronoi cells. In case of an infinite Voronoi cell, the infinite rays edging the cell are replaced by finite length segments, with a length greater than the diameter of the input shape. The boundaries of the cell, now finites, are then triangulated. Since the boundary of the 3D shape is given as a triangulated mesh, the clipping problem now reduces to the computation of triangle-triangle intersections. Once such intersections have been found, we set them as constraints. Constraints are represented as line segments. The intersection I of two triangles is processed according to the following rules:

Case 1: if I is a point, ignore it.

Case 2: if I is a line segment, add it to the constraints.

Case 3: if I is a triangle, add its three edges to the constraints.

Case 4: otherwise, I is a polygon, construct segments using adjacent vertices of this polygon and add them to the constraints.

These cases are illustrated in Figure 6.1. The interior of each intersected triangle of either the cell boundary or the mesh is then robustly triangulated using a 2D constrained Delaunay triangulation.

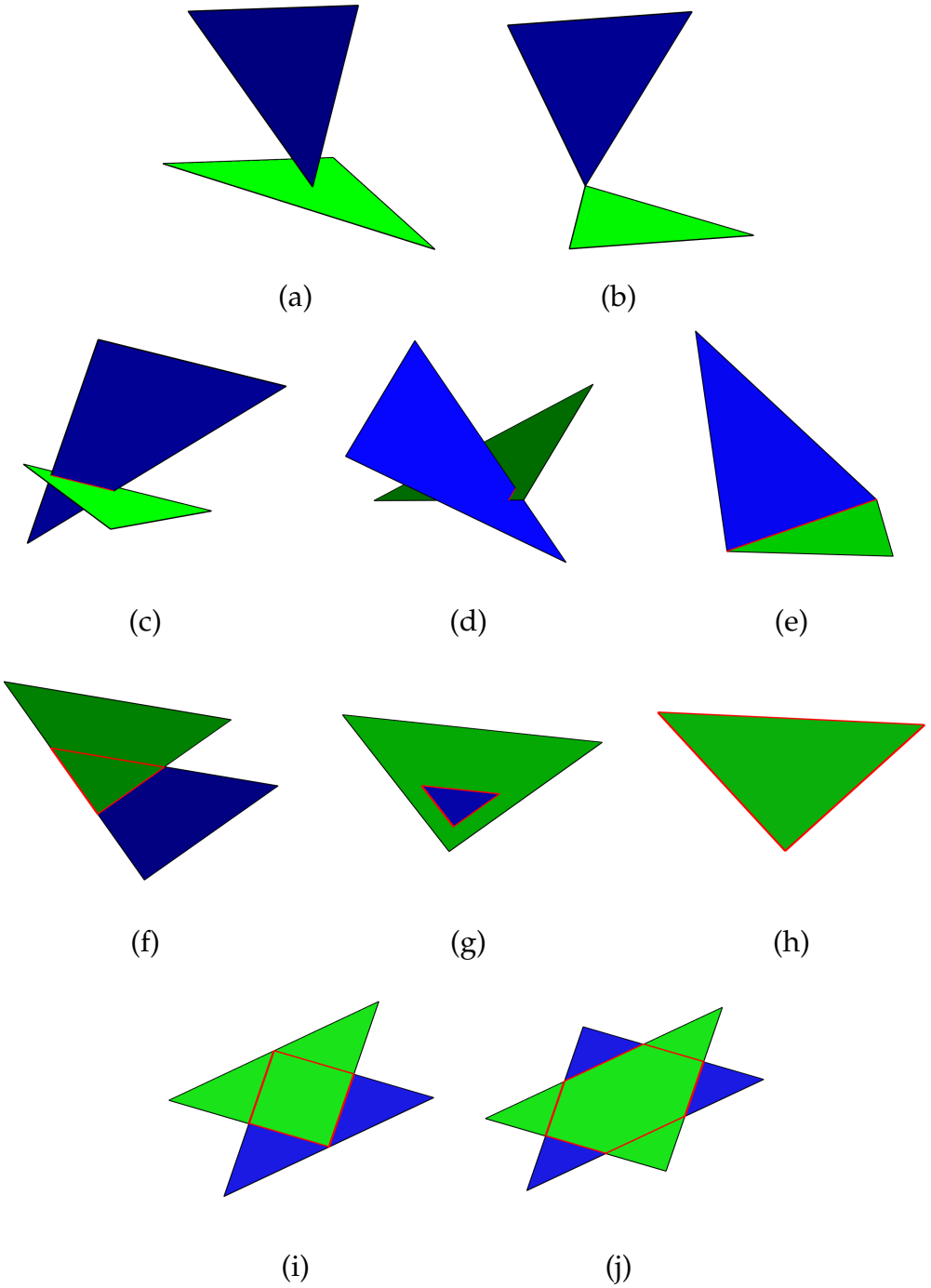


Figure 6.1 – Different intersection cases. Constraints (line segments) are shown in red. (a, b) Case 1. (c, d, e) Case 2. (f, g, h) Case 3. (i, j) Case 4. (b,e,h) represent singularities.

Our clipping algorithm is summarized below (Algorithm 2). Figure 6.2 depicts its main steps.

Algorithm 2: Clipping algorithm.

```
Data: cell Ω, 3D shape V bounded by a triangulated mesh S
Result: clipped cell Ω'
S_Ω := TriangulateBoundary(Ω);
I := Intersection(S_Ω, S);
if I not empty then
    T := IntersectedTriangles(S_Ω, I);
    for each triangle t of T do
        T_1 := ConstrainedDelaunay(t, I);
        for each triangle t_1 of T_1 do
            if IsInside(t_1, V) then
                Add(t_1, Ω');
            end
        end
    end
    T := IntersectedTriangles(S, I);
    for each triangle t of T do
        T_2 := ConstrainedDelaunay(t, I);
        for each triangle t_2 of T_2 do
            if IsInside(t_2, Ω) then
                Add(t_2, Ω');
            end
        end
    end
end
else
    Ω' := Ω;
end
```

6.1.3 Experimental Results

Our algorithm is implemented in C++ and we use the CGAL library [CGAL] for linear geometry Kernel which defines data structures and their operations. All computations were performed on an Intel Xeon E5-2643 with 3.30 GHz CPU.

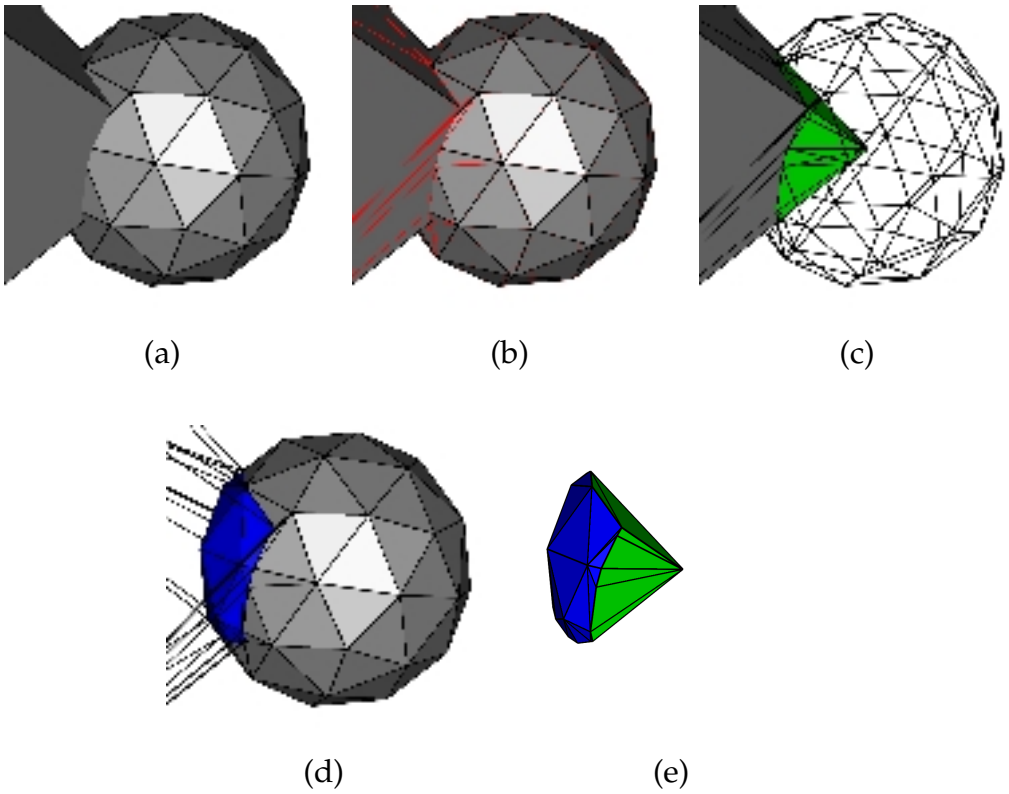


Figure 6.2 – Clipping algorithm. (a) Input: a Voronoi cell and a 3D shape (here: a closed ball) bounded by a mesh. (b) Constrained Delaunay triangulations of the boundary of the cell and of the mesh. (c) In green: boundary of the cell inside the closed ball. (d) In blue: part of the mesh inside the cell. (e) Result: the clipped cell is bounded by the green and the blue triangulations.

Computation times for our clipping method are shown in Table 6.1. In particular, we have tested this method on complex and badly triangulated shapes and scenes to show its efficiency and robustness, see Figure 6.3.

6.1.4 Discussion

The advantage of our clipping algorithm is to reduce a 3D problem to 2D. It eliminates many complicated degenerate case and facilitates the process. The most costly part of our clipping algorithm is the computation of intersections between triangles for which we use the CGAL Kernel *Exact predicates exact constructions kernel* which is not efficient enough. The performance of our algorithm would be much improved if an adaptive

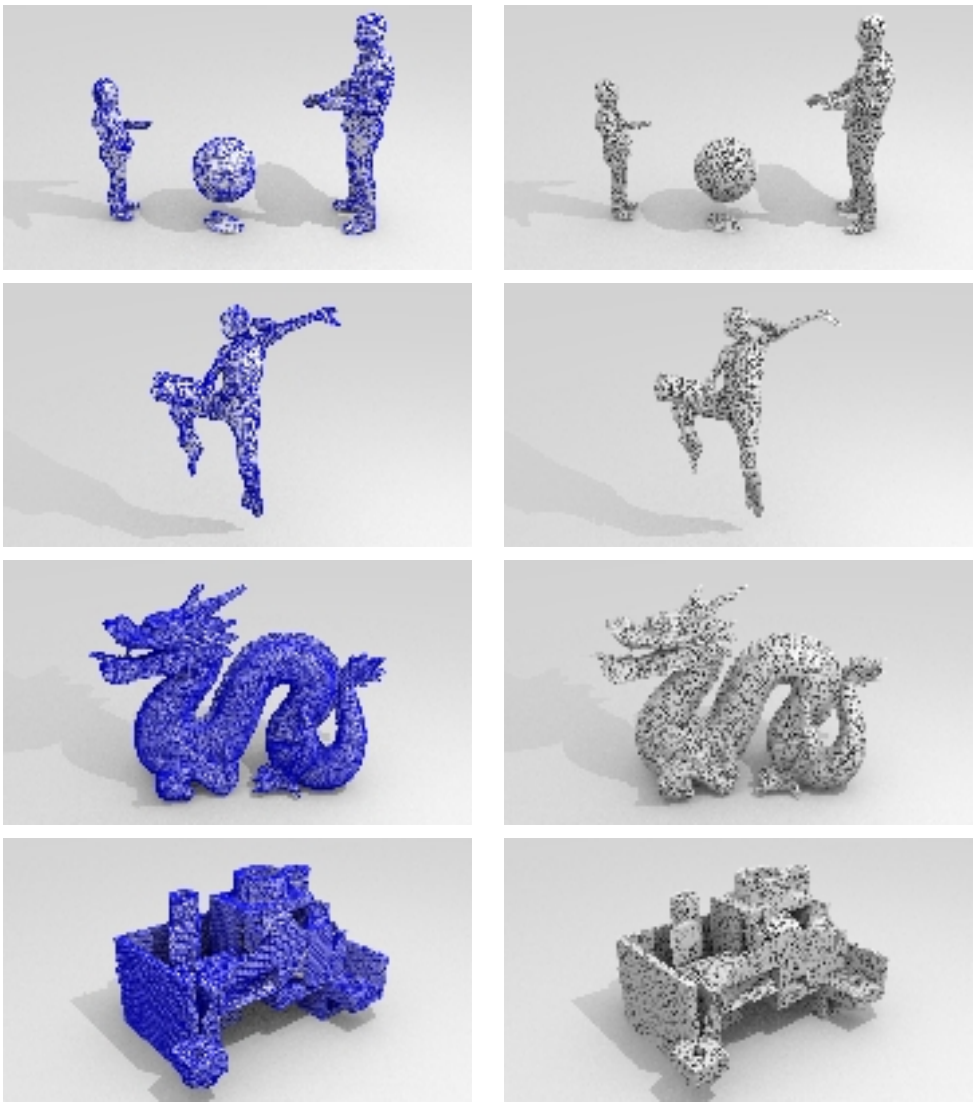


Figure 6.3 – More examples of clipped (non Centroidal) Voronoi diagrams. Left: input triangulations. Right: clipped Voronoi diagrams.

Shape	Figure	Vertices (k)	Triangles (k)	Sites (k)	Time (s)
Homer	4.8	10	20	35	28.66
Ball	4.7	0.5	1	1	0.55
Bunny	4.9	10	20	50	30.06
Kitten	4.9	10	20	80	54.57
Armadillo	4.9	173	346	100	103.06
Ballgame	6.3	12.4	24.8	10	17.92
Dancer	6.3	15.1	30.2	5	10.52
Dragon	6.3	100	200	0.1	1.89
Dragon	6.3	100	200	1	6.14
Dragon	6.3	100	200	10	21.26
Dragon	6.3	100	200	100	99.09
CAD model	6.3	182	364	100	269.67

Table 6.1 – Computation times for clipped Voronoi diagrams.

and more efficient geometry kernel could be used. Furthermore, our algorithm is currently implemented on CPU, a GPU version would probably be much faster.

6.2 Shape Animation with Combined Captured and Simulated Dynamics

In this section, we introduce an application of shape animation using CVTs. A novel framework is proposed to deal with the capture, shape tracking and animation generation. We consider CVTs in order to apply combined kinemamtic and physical constraint over cells using rigid body simulations. Moreover, regular and uniform cells ease the implementation of local constraints such as physical constraints for simulation or local deformation constraints when tracking shapes over time sequences.

6.2.1 Introduction

Creation of animated content has become of major interest for many applications, notably in the entertainment industry, where the ability to produce animated virtual characters is central to video games and special effects. Plausibility of the animations is a significant concern for such

productions, as they are critical to the immersion and perception of the audience. Because of the inherent difficulty and necessary time required to produce such plausible animations from scratch, motion capture technologies are now extensively used to obtain kinematic motion data as a basis to produce the animations, and are now standard in the industry.

However, motion capture is usually only the first stage in a complicated process, before the final animation can be obtained. The task requires large amounts of manual labor to rig the kinematic data to a surface model, correct and customize the animation, and produce the specifics of the desired effect. This is why, in recent times, video-based 3D performance capture technologies are gaining more and more attention, as they can be used to directly produce 3D surface animations with more automation, and to circumvent many intermediate stages in this process. They also make it possible to automatically acquire complex scenes, shapes and interactions between characters that may not be possible with the standard sparse-marker capture technologies. Still, the problem of customizing the surface animations produced by such technologies to yield a modified animation or a particular effect has currently no general and widespread solution, as it is a lower level representation to begin with.

In this work, we propose a novel system towards this goal, which produces animations from a stream of 3D observations acquired with a video-based capture system. The system provides a framework to push the automation of animation generation to a new level, dealing with the capture, shape tracking, and animation generation from end-to-end with a unified representation and solution. In particular, we entirely circumvent the need for kinematic rigging and present results in this report obtained without any manual surface correction.

Although the framework opens many effect possibilities, for the purpose of the demonstration here, we focus our effort on combining the real raw surface data captured with physics and procedural animation, in particular using a physics-based engine. To this aim, we propose to use regular Voronoi tessellations to decompose acquired shapes into volumetric cells, as a dense volume representation upon which physical constraints are easily combined with the captured motion constraints. Hence, shape motions can be perturbed with various effects in the animation, through forces or procedural decisions applied on volumetric cells. Motion constraints are obtained from captured multi-view sequences of live actions. We do not consider skeletal or surfacic motion models for that purpose but directly track volumetric cells instead. This ensures high flexibility in both the class of shapes and the class of physical constraints

that can be accounted for. We have evaluated our method with various actor performances and effects. We provide both quantitative results for the shape tessellation approach and qualitative results for the generated 3D content. They demonstrate that convincing and, to the best of our knowledge, unprecedented animations can be obtained using video-based captured shape models.

In summary, this work considers video-based animation and takes the field a step further by allowing for physics-based or procedural animation effects. The core innovation that permits the combination of real and simulated dynamics lies in the volumetric shape representation we propose. The associated tessellated volumetric cells can be both tracked and physically perturbed hence enabling new computer animations.

6.2.2 Related Work

This work deals with the combination of simulated and captured shape motion data. As mentioned earlier, this has already been explored with marker based mocap data as kinematic constraints. Following the work of Popović and Witkin [1999], a number of researchers have investigated such combination. They propose methods where mocap data can be used either as reference motion [Popović and Witkin, 1999] [Zordan and Hodgins, 2002] [Sulejmanpašić and Popović, 2005], or to constrain the physics-based optimization associated to the simulation with human-like motion styles [Safonova et al., 2004] [Liu et al., 2005] [Ye and Liu, 2008] [Wei et al., 2011]. Although sharing conceptual similarities with these methods, our work differs substantially. Since video-based animations already provide natural animations, our primary objective is not to constrain a physical model with captured kinematic constraints but rather to enhance captured animations with user-specified animation constraints based on physics or procedural effects. Consequently, our simulations are not based on biomechanical models but on dynamic simulations of mechanical effects. Nevertheless, our research draws inspiration from these works.

With the aim to create new animations using recorded video-based animations, some works consider the concatenation of elementary segments of animations, *e.g.* [Casas et al., 2013], the local deformation of a given animation, *e.g.* [Cashman and Hormann, 2012], or the transfer of a deformation between captured surfaces, *e.g.* [Sumner and Popović, 2004]. While we also aim at generating new animations, we tackle a different issue in this research with the perturbation of recorded animations according to simulated effects.

Our method builds on results obtained in video-based animations with multi-camera setups, to obtain the input data of our system. Classically, multi-view silhouettes can be used to build free viewpoint videos using visual hulls [Matusik et al., 2000] [Gross et al., 2003] or to fit a synthetic body model [Carranza et al., 2003]. Visual quality of reconstructed shape models can be improved by considering photometric information [Starck and Hilton, 2007] [Tung et al., 2009] and also by using laser scanned models as templates that are deformed and tracked over temporal sequences [De Aguiar et al., 2007] [Vlasic et al., 2008] [De Aguiar et al., 2008]. Interestingly, these shape tracking strategies provide temporally coherent 3D models that carry therefore motion information. In addition to geometric and photometric information, considering shading cues allows to recover finer scale surface details as in [Vlasic et al., 2009] [Wu et al., 2012].

More recent approaches have proposed to recover both shapes and motions. They follow various directions depending on the prior information assumed for shapes and their deformations. For instance in the case of human motion, a body of work assumes articulated motions that can be represented by the poses of skeleton based models, *e.g.* [Vlasic et al., 2008] [Gall et al., 2009] [Straka et al., 2012]. We base our system on a different class of techniques aiming at more general scenarios, with less constrained motion models simply based on locally rigid assumptions in the shape volume [Allain et al., 2015]. This has the advantage that a larger class of shapes and deformations can be considered, in particular motions of humans with loose clothes or props. The technique also has the significant advantage that it allows to track dense volumetric cell decompositions of shapes, thereby allowing for consistent captured motion information to be associated and propagated with each cell in the volume, a key property to build our animation generation framework on.

In the following, we provide a system overview, followed by a detailed explanation of how we tessellate 3D input observations into regular polyhedral cells, to be subsequently tracked and used as primary animation entity (see Section 6.2.4). In order to recover kinematic constraints from real actions, our system then tracks polyhedral cells using surface observations and a locally rigid deformation model (see Section 6.2.5). Finally, a physics or procedural simulation integrates the animation constraints over the shape (see Section 6.2.6). To our knowledge, this is the first attempt to propose such an end-to-end system and framework to generate animations from real captured dynamic shapes.

6.2.3 System Overview

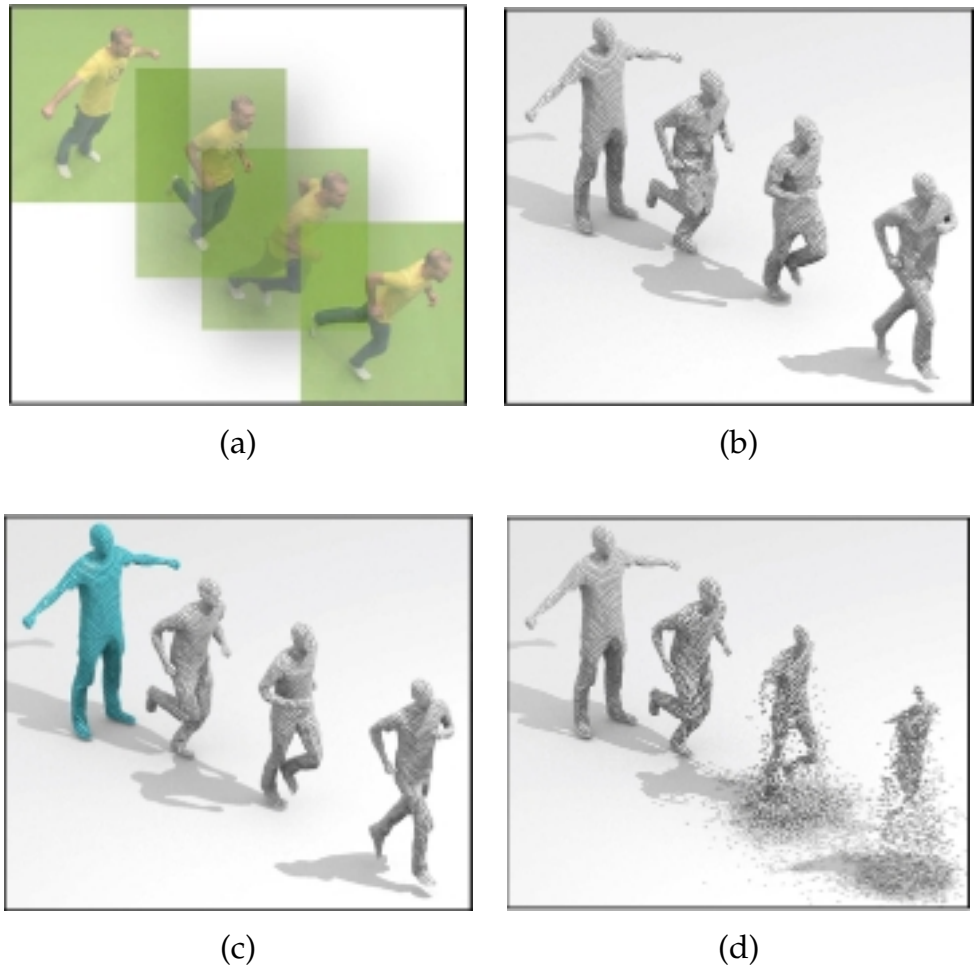


Figure 6.4 – From video-based shape capture to physic simulation. (a) Original videos. (b) Volumetric shape representations. (c) Volumetric shape tracking (template in blue). (d) Physics-based simulation. The approach uses multiple videos and Voronoi tessellations to capture the volumetric kinematic of a shape motion which can then be reanimated with additional mechanical effects, for instance volumetric erosion with gravity in the figure.

We generate a physically plausible animation given a sequence of 3D shape observations as well as user specifications for the desired effect to be applied on the animation. 3D observations are transformed into temporally consistent volumetric models using centroidal Voronoi tessellations and shape tracking. Kinematic and physical constraints are then

combined using rigid body physics simulation. The approach involves the following main steps depicted in Figure 6.4.

6.2.3.1 Video Based Acquisition

Input to our system are 3D observations of a dynamic scene. Traditional and probably most common dynamic scenes in graphics are composed of human movements; however our system can consider a larger class of shapes since only local rigidity is assumed to get temporally consistent shape models. 3D observations are assumed to be obtained using a multi-camera system and can be in any explicit, *e.g.* meshes or implicit, *e.g.* from point clouds through Poisson function, forms. Our own apparatus is composed of 68 calibrated and synchronized cameras with a resolution up to 2048×2048 pixels. The acquisition space is about $8m \times 4m$ and the camera frame rate can go up to 50 fps at full resolution. The outputs of this step are point clouds with around $100k$ points.

6.2.3.2 Volumetric Representation

Input 3D observations are tessellated into polyhedral cells. This volumetric representation is motivated by two aspects of our animation goal: first, the representation is well suited to physical simulation; second, volumetric deformation models are more flexible than skeleton based models, hence enabling non rigid shape deformations. Still, they allow for locally rigid volumetric constraints, a missing feature with surface deformation models when representing shapes that are volumes. We adopt centroidal Voronoi tessellations since they produce regular and uniform polyhedral cells.

6.2.3.3 Tracking

In this step, incoherent volumetric shape models of a temporal sequence are transformed into coherent representations where a single shape model is evolving over time. This provides kinematic information at the cell level that will further be used in the simulation. We use a tracking method [Allain et al., 2015] that finds the poses of a given template shape at each frame. The template shape is taken as one of the volumetric models at a frame. The approach uses a volumetric deformation model, instead of surface or skeleton based model, to track shapes. It optimizes the pose of the template shape cells so as to minimize a distance cost to an input shape model while enforcing rigidity constraints on the local cell configurations.

6.2.3.4 Simulation

The tracked cell representation is both suitable for tracking and convenient for solid based physics. We embed the tracked volumetric model in a physical simulation, by considering each cell to be a rigid solid shape in mechanical interaction with other cells and scene shapes. We ensure cohesion of cells by attaching a kinematic recall force in the simulation, and offer various controls as to how the scene may deform, collide, or rupture during contacts and collisions. This simple framework allows for a number of interesting effects demonstrated in 6.2.7.

6.2.4 Volumetric Shape Modeling

As mentioned before, we compute CVTs of input observations because combined kinematic and physical constraints can be easily applied over cells using rigid body simulations. Moreover, CVTs provide regular and uniform cells that ease the implementation of local constraints or deformation constraints compared with voxel- and Delaunay-based tessellations. Figure 6.5 shows the comparison of the three approaches. For more details about the construction of CVTs from implicit shapes and the comparison of the three approaches, please see Chapter 5.

6.2.5 Volumetric Shape Tracking

With the volumetric decomposition proposed above, we now need to define a model by which scene dynamics can be captured through deformations expressed over this decomposition. We consider here as input a time sequence of inconsistent CVT decompositions independently estimated at each frame and we look for a time consistent volumetric decomposition that encode cell motions. We opt for a capture by deformation approach where a template CVT, taken from the input sequence in our case, is tracked throughout the sequence. This volumetric strategy is motivated by two observations. First, attaching the deformation model to the cell shape representation used for the animation directly provides the necessary cell dynamic information to the simulation. It avoids therefore the interpolation between an intermediate motion model, *e.g.* a skeleton or a mesh, and the animation model; Such interpolation being difficult to perform consistently over time. Second, as shown in [Allain et al., 2015], it provides a simple tool for embedding volume-preservation constraints that increase the robustness of the tracking over the dynamic scenes we

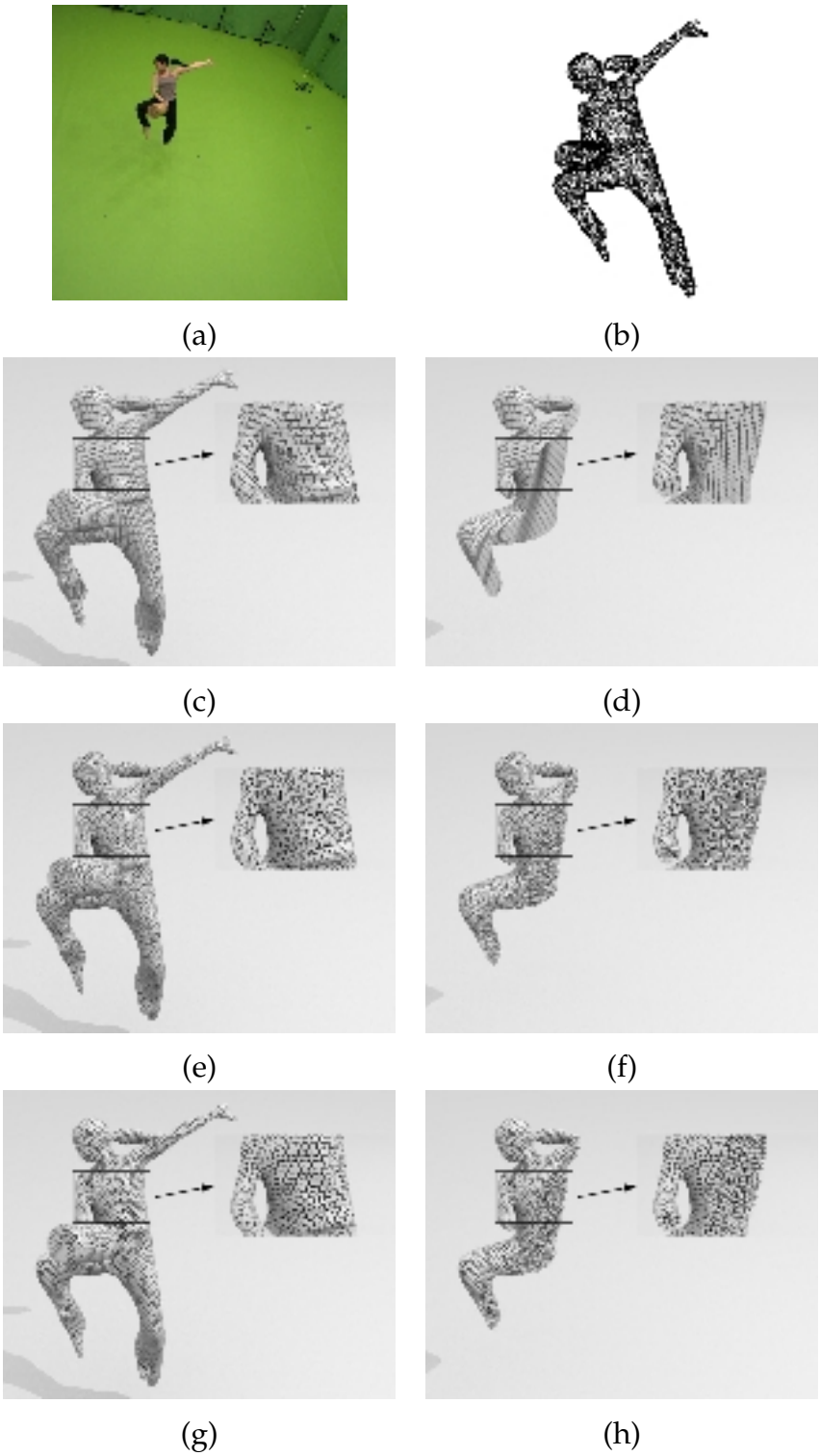


Figure 6.5 – (a, b) Input multi-camera observation and point clouds (65386 pts). (c, d) Tessellations generated using voxels [Chernyaev, 1995]. (e, f) Tetrahedrisations generated using Delaunay refinement [CGAL]. (g, h) Clipped Centroidal Voronoi Tessellations (14455 sites).

consider. We describe below the generative approach [Allain et al., 2015] that we follow.

6.2.5.1 Tracking Formulation

We are given a sequence of CVTs $\mathcal{V}$ and a template model $\hat{V}$. $\hat{V}$ can be one model taken from the sequence or any other model (*e.g.* a 3D scan) decomposed into a CVT. The tracking consists then in fitting $\hat{V}$ to each $V \in \mathcal{V}$. This can be formulated as a maximum a posteriori (MAP) estimation of the deformation parameters $\hat{\Theta}$ that maximizes the posterior distribution $\mathcal{P}(\Theta|\mathcal{V})$ of the parameters Θ given the observations $\mathcal{V}$:

$$\hat{\Theta} = \arg\max_{\Theta} \; P(\Theta|\mathcal{V}) \simeq \arg\max_{\Theta} \; P(\mathcal{V}|\Theta) \; P(\Theta).$$

Taking the log of the above expression yields the following optimization problem:

$$\hat{\Theta} = \arg\max_{\Theta} \; E_{data}(\mathcal{V}, \Theta) + E_{prior}(\Theta), \tag{6.1}$$

where the data term E_{data} evaluates the log-likelihood of a set of deformation parameters Θ given the observations $\mathcal{V}$, and the regularization term E_{prior} enforces prior constraints on the deformation, *e.g.* local rigidity. We detail below the parameterization Θ we use for the deformation and the associated energy terms.

6.2.5.2 Motion Parameterization

The deformation model is defined over CVT cells in the shape decomposition and, for efficiency, on aggregates of cells which reduces the number of terms and parameters. To this goal, CVT cells are grouped together as a set of volumetric patches P_k using a k-medoids algorithm, as shown in Figure 6.6. Such patches can be either adjacent to the surface or completely inside the template shape's volume, which is of particular interest to express non-rigid deformation of the model while preserving the local volume and averting over-compression or dilation. The positional information of a patch is represented as a rigid transform $\mathbf{T}_k^t \in SE(3)$ at every time t. Each position $\mathbf{x}_{k,q}$ of a CVT sample is indiscriminately labeled as a point q. Its position can be written as a transformed version of its template position $\mathbf{x}_q^0$ as follows, once the patch's rigid transform is applied:

Figure 6.6 – The template model used to recover the runner sequence motion, with its CVT decomposition cells, and the cell clusters in different colors.

$$\mathbf{x}_{k,q} = \mathbf{T}_k(\mathbf{x}_q^0). \tag{6.2}$$

A *pose* of the shape is thus defined as the set of patch transforms $\mathbf{T} = \{\mathbf{T}_k\}_{k\in\mathcal{K}}$, which expresses the deformation of every component in the shape. The parameterization Θ is then the set of pose parameters of the template over the considered time sequence $\mathcal{T}$:

$$\Theta = \{\mathbf{T}^t\}_{t\in\mathcal{T}}.$$

6.2.5.3 Data Term

We assume the observed shape V^t at time t is described by the point cloud $\mathbf{Y}^t = \{\mathbf{y}_o^t\}$. We assume this point cloud to include inner volume points and surface points, *i.e.* the CVT sites as well as the outside surface points.

In order to measure how a deformed version of the template explains the observed shape, first the associations between the observations and the template must be determined. This is achieved via a soft ICP strategy that iteratively reassigns each observation $\mathbf{y}_o^t$ to the template volumetric patches. For simplicity, each observation $\mathbf{y}_o^t$ is associated to the patch

P_k via the best candidate point $\mathbf{x}_{k,q}$ of patch P_k and with an association penalty $\alpha_{o,k}$.

The matching penalty $E(\mathbf{y}_o^t, \mathbf{T}_k^t)$ that evaluates how well P_k explains $\mathbf{y}_o^t$ is then the weighted distance between $\mathbf{y}_o^t$ and the best candidate $\mathbf{x}_{k,q}$, that is the template point $\mathbf{x}_q^0$ transformed by the current pose $\mathbf{T}_k^t$ of the template model:

$$E(\mathbf{y}_o^t, \mathbf{T}_k^t) = \alpha_{o,k}^t \|\mathbf{y}_o^t - \mathbf{T}_k^t(\mathbf{x}_q^0)\|. \tag{6.3}$$

Associations are additionally filtered using a compatibility test. Observed surface points are associated to template surface points with similar orientations with respect to a user defined threshold θ_{max}; Observed inner points are associated to template inner points that present similar distances to the surface up to a user defined tolerance ϵ. If there is no compatible candidate in a patch P_k, then P_k is discarded for the association with $\mathbf{y}_o^t$, *i.e.* $\alpha_{o,k}^t = 0$. Finally:

$$E_{data}(V^t, \Theta^t) = \sum_{o,k} E(\mathbf{y}_o^t, \mathbf{T}_k^t). \tag{6.4}$$

6.2.5.4 Regularization Term

The pose of a shape is defined by the set of rigid motion parameters of the shape volumetric patches. While these parameters hardly constraint the patch motions, they do not define a coherent shape motion since each patch moves independently of the others. In order to enforce shape cohesion, soft local rigidity constraints, reflecting additional prior knowledge on shape deformation parameters, are considered. These constraints rely on a pose distance function that evaluates how distant from a rigid transformation a deformation between two poses is. Once such a distance is defined, the regularization term is defined as the sum of distances between all poses in the sequence and a given pose that can be a reference pose or an estimated mean pose, as explained below.

Pose Distance

To simplify the estimation, the shape distance is expressed over coordinates of points belonging to patches and not on the parameters of the pose itself:

$$\mathcal{D}(\mathbf{T}^i, \mathbf{T}^j) = \sum_{(P_k, P_t) \in \mathcal{N}} \mathcal{D}_{kl}(\mathbf{T}^i, \mathbf{T}^j), \quad \text{with} \tag{6.5}$$

$$\mathcal{D}_{kl}(\mathbf{T}^i, \mathbf{T}^j) = \sum_{q \in P_k \cup P_l} \|\mathbf{T}^i_{k-l}(\mathbf{x}^0_q) - \mathbf{T}^j_{k-l}(x^0_q)\|^2, \tag{6.6}$$

where $\mathbf{T}^i_{k-l} = \mathbf{T}^{i-1}_l \circ \mathbf{T}^i_k$ is the relative transformation between patches P_k and P_l for pose i, and $\mathcal{N}$ is the set of neighboring patch pairs within the shape. Intuitively, this distance measures whether the relatives poses between neighboring volumetric patches are preserved during motion between two shape poses.

Deformation Energy

The pose distance above allows to compute a deformation energy between two poses and with respect to a reference pose from which the patch transformations $\mathbf{T}^k_t$ are expressed. This reference pose can be taken as the identity pose of the initial template model (see Figure 6.6 for instance). However such a strategy is biased toward the template pose and discourage locally rigid motions between poses that are distant from the template pose. A more appropriate approach is to exploit the pose distance function to first define a mean pose $\bar{\mathbf{T}}$ over a time window $\{t\}$:

$$\bar{\mathbf{T}} = \arg\min_{\mathbf{T}} \sum_t \mathcal{D}(\mathbf{T}^t, \mathbf{T}).$$

This averaged pose can then be taken as an evolving reference pose from which non-rigid deformations are measured to define the deformation energy over the associated time interval:

$$E(\{\mathbf{T}^t\}, \bar{\mathbf{T}}) = \sum_t \mathcal{D}(\mathbf{T}^t, \bar{\mathbf{T}}). \tag{6.7}$$

This imposes general proximity of poses to a sequence specific "rest" pose. These rest poses must also be constrained to some form of inner cohesion. This is ensured by minimizing the distance from the mean pose to the identity pose of our initial template:

$$E(\bar{\mathbf{T}}) = \mathcal{D}(\bar{\mathbf{T}}, \mathbf{Id}). \tag{6.8}$$

This definition of deformation has a number of advantages: first it enforces geometric cohesion and feature preservation, and second it is quite simple to formulate and to optimize, as the minimization of (6.7)

and (6.8) translates to a sum of least square constraints over the set of CVT sites.The prior energy term finally writes:

$$E_{prior}(\Theta = \{\mathbf{T}^t\}) = \mathcal{D}(\bar{\mathbf{T}}, \mathbf{Id}) + E(\{\mathbf{T}\}, \bar{\mathbf{T}}). \tag{6.9}$$

We jointly extract poses and a sequence mean pose by minimizing the sum of the data terms (6.4) and the prior term (6.9). Section 6.2.7 shows results on various sequences and gives run time performances.

6.2.6 Combined Animation

The template representation of the subject now being consistently tracked across the sequence, we can use the tracked cells as input for solid physics-based animation. We have purposely chosen CVTs as a common representation as they can be made suitable for shape and motion capture as we have shown, while being straightforwardly convenient for physics-based computations. In fact CVT cells are compact, convex or easily approximated by their convex hull. This is an advantage for the necessary collision detection phase of physics models, as specific and efficient algorithms exist for this case [Gilbert et al., 1988] [Rabbitz, 1994]. We here describe the common principles of our animation model, with more specific applications being explored and reported on in the following sections.

As our animation framework is solid-based, we base our description on commonly available solid-based physics models, *e.g.* [Baraff, 1997]. Each CVT cell is considered a homogeneous rigid body, whose *simulated* state is parameterized by its 3D position, rotation, linear momentum and angular momentum. The cell motion is determined by Newton's laws through a differential equation involving the cell state, the sum of forces and sum of torques that are applied to the cell.

The animation is thus obtained by defining the set of forces and torques applied at each instant, and iteratively solving these differential equations to obtain a new cell position and orientation for a target time step, using one of many available techniques. For our demonstrator, we use the simple and efficient off-the-shelf Bullet Physics engine [Coumans et al., 2013] [Catto, 2005].

6.2.6.1 Ordinary Applied Forces

We classically apply the constant gravity force $F_g = Mg$. We apply additional external forces or constraints as needed for the target application, as will be detailed in the coming sections. Additionally, contact

forces such as collisions are handled with scene shapes, as well as between different cells of the CVT. For this purpose the physics engine first needs to detect the existence of such contacts. It relies on a hierarchical space decomposition structure, such as an AABB-tree, for broad-phase collision detection, *i.e*. coarse elimination of collision possibilities. A narrow-phase collision test follows, between shapes lying in the same region of space as determined by the AABB-tree traversal. In this narrow phase the full geometry of shapes is examined, *e.g*. using the GJK algorithm [Gilbert et al., 1988] for pairs of convex polyhedra. Once the existence of a contact is established, various strategies exist to deal with the collision, *e.g*. by introducing impulse repulsion forces to produce a collision rebound. We follow the common approach of modeling contacts as a linear complementary problem (LCP) popularized by [Baraff, 1994], which derives contact forces as the solution of a linear system that satisfies certain inequality constraints. These constraints are typically formulated using a constraint Jacobian over the combined state spaces of rigid bodies. [Catto, 2005] expose the specific variant applied in the context of the Bullet Physics engine.

6.2.6.2 Physical Modeling of Kinematic Control

To relate the physical simulation to the acquired non-rigid poses of the model, we need to introduce coupling constraints. Our goal is to allow the model to materialize and control the tradeoff between the purely kinematic behavior acquired from visual inputs for the cell, and the purely mechanically induced behavior in the simulation. First it is important to note that the temporal discretization used for acquisition and for simulation and rendering of the effects are generally different. Consequently the first stage in achieving our goal is to compute a re-sampling of the pose sequence, to the target simulation and rendering frequency, using position and quaternion interpolation. The poses so obtained are here referred as the acquired cell poses $\hat{x}^a(t)$ and $R^a(t)$. Second, we formulate the coupling by introducing a new kinematic recall force, in the form of a damped spring between the acquired cell poses and the simulated cell poses:

$$F_r(t) = k.(\hat{x}^a(t) - \hat{x}(t)) - \lambda.\frac{\mathrm{d}}{\mathrm{d}t}(\hat{x}^a(t) - \hat{x}(t)), \tag{6.10}$$

where k and λ are respectively the rigidity and damping coefficients of the spring, which control the strength and numerical stability of the coupling.

6.2.7 Visual Effects

Figure 6.7 – An animation that combines video-based shape motion (left) and physical simulation (right). Our method allows to apply mechanical effects on captured dynamic shapes and generates therefore plausible animations with real dynamics.

This section presents various animation results on three captured animations of variable nature and speed. RUNNER shows a male character running in a straight line, during 3 motion cycles. This animation lasts 2.5 *s*. In CAGEBIRDDANCE a female dancer moves while holding a bird cage. This sequence is 56 *s* long and shows a complex sequence of motions which would be difficult to synthesize without sensors. Finally, in SLACKLINE a male acrobat evolves on a non rigid line above the ground, for 25 *s*. The input sequences of temporally inconsistent 3D point clouds are made respectively of 126 (RUNNER), 2800 (CAGEBIRDDANCE) and 1240 (SLACKLINE) temporal frames.

Parameters

The interior of all shapes has been tessellated according to 6.2.4 using 5000 cells. 10 iterations of the L-BFGS quasi-Newton algorithm were applied, except for the template shape where 50 iterations were applied. We use a temporal window of 10 frames for the tracking, and cluster the 5000 cells into 200 patches. 60 iterations were applied.

We list below examples of visual effects that we were able to generate using the proposed approach. First, we present animations combining tracking results with solid dynamics simulation (6.2.7.1). Then we present other visual effects that also exploit the volumetric tracking information (6.2.7.2 and 6.2.7.3).

6.2.7.1 Asynchronous Kinematic Control Deactivation

In order to show the effect of gravity while keeping the dynamic motion of the input sequence, we deactivate kinematic control forces independently for each cell. After being deactivated, a cell usually falls to the ground, since it follows a trajectory determined only by gravity, collision forces and its initial velocity. Asynchronous cell deactivations result in an animation combining cells that follow their tracking trajectory and cells that fall. By choosing diverse strategies for scheduling cell deactivations, a wide variety of animations can be obtained.

We explore here three possible deactivation strategies that are based on different criteria. Note that while these effects are straight-forward to produce with our volumetric framework, it would be difficult to obtain them if only surface or skeleton-based tracking was available.

6.2.7.1.1 Rupture under Collisions

Collisions with obstacles sometimes deviate a cell from its theoretical trajectory, which results in an increase of the recall force magnitude (see Eq. 6.10). This phenomenon can be detected and used for simulating the rupture of the material: when the recall force magnitude of a cell is above a given threshold, we deactivate the recall force (for this cell only). This makes the rupture looks like the consequence of the collision.

Figure 6.8 shows a heavy pendulum that hits the subject and makes a hole in it. Under the intensity of the collision, several cells are ejected (rupture) and fall to the ground.

6.2.7.1.2 Heat Diffusion

In order to make the cell deactivation both temporally and spatially progressive, we rely on a diffusion algorithm. We diffuse an initial temperature distribution inside the volume according to the diffusion equation. Deactivation is triggered when the cell temperature is above a given threshold.

Heat diffusion in a CVT

The CVT provides a graph structure on centroids, which is a subgraph of the Delaunay tetrahedralization of the cells centroids. The heat diffusion on a graph structure is expressed by the heat equation:

$$(\partial/\partial t + \mathbf{L})\mathbf{F}_t = 0,$$

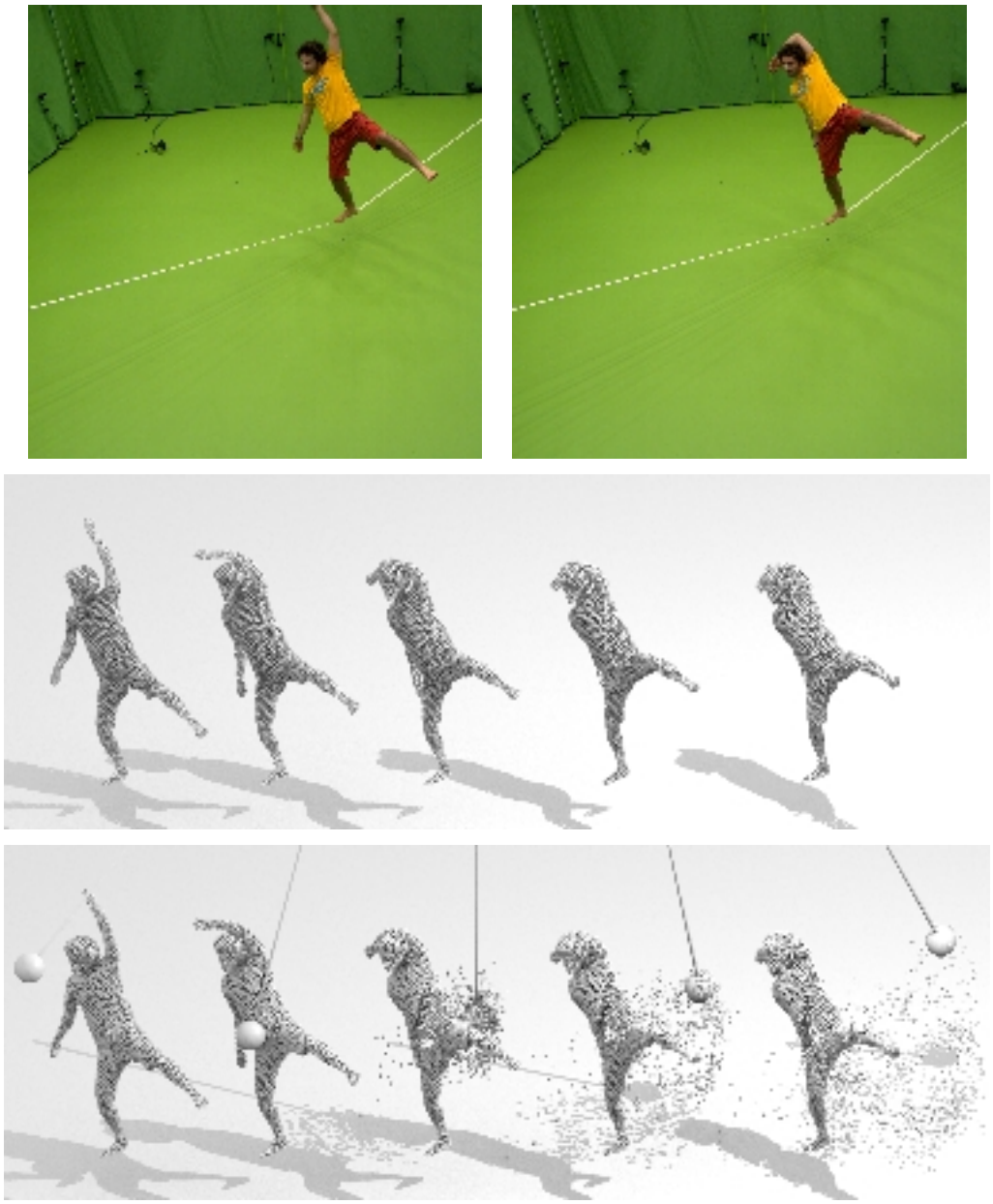


Figure 6.8 – Input SLACKLINE Multi-camera observations (left), tracking result of the SLACKLINE sequence (top) and combination with the effect of collision with a pendulum (bottom).

where $\mathbf{F}_t$ is the column vector of centroids temperatures at time t, and $\mathbf{L}$ is the combinatorial graph Laplacian matrix (note that a geometric Laplacian is not necessary since centroids are regularly distributed in space). Given an initial temperature distribution $\mathbf{F}_0$, this equation has a unique solution

$$\mathbf{F}_t = \mathbf{H}_t \mathbf{F}_0,$$

where $\mathbf{H}_t = e^{t\mathbf{L}}$ is the heat diffusion kernel, which can be computed by means of the spectral decomposition of $\mathbf{L}$.

We compute the temperature evolution on the CAGEBIRDDANCE sequence for an initial temperature distribution where all cell temperatures are zero, except for the dancer's head top cells (which are set to 1). We observe in Figure 6.7 that cells fall progressively across time, from the upper to the lower body parts. Note that the cage cells remain kinematically controlled since heat is not transferred between different connected components.

6.2.7.1.3 Morphological Erosion

The discrete cell decomposition of shapes allows to apply morphological operators. To illustrate this principle, we have experimented erosion as shown in Figure 6.4. In this example, cells are progressively eroded starting from the outside. The morphological erosion is performed by deactivating each cell after a delay proportional to the distance between the cell centroid and the subject's surface. The distance is computed only once for each cell, on the template shape. Figure 6.4 shows the erosion animation on the RUNNER sequence. Note that the operation progressively reveals the dynamic of the inner part of the shape.

6.2.7.2 Time Persistence

In this example, we experiment time effects over cell decompositions. To this aim, dynamic copies of the model are generated at regular time intervals. These copies are equipped with deceleration and erosion effects over time and create therefore ghost avatars that vanish with time (see Figure 6.9). The benefit of the tracked volumetric representation in this simulation is the ability to attach time effect to the model behavior at the cell level, for instance lifetime and deceleration in the example.

Figure 6.9 – Time persistence on the RUNNER sequence: a slower copy of the shape that erodes over time is generated at regular intervals.

6.2.7.3 Morphing

Our dynamic representations allows to apply volumetric morphing between evolving shapes, enabling therefore new visual effects with real dynamic scenes. To this purpose, cells of the source shape are first matched to the target shape. Second, each cell is individually morphed to its target cell at a given time within the sequence. Time ordering is chosen such that cells in the source shape are ordered from the outside to the inside, and associated with the cells of the target shape ordered from the inside to the outside. Cells are transformed from the source to the destination by interpolating their positions and using Kent et al. [1992] to morph their polyhedral shapes. Figure 6.10 shows the dynamic morphing of the RUNNER sequence onto the CAGEBIRDDANCE sequence.

6.2.7.4 Run Time Performance

Our approach has been tested on a dual Intel Xeon E5-2665 processor with 2.40 GHz each. For each animation, the Poisson runs in 0.67 *s* per frame on average, and the volumetric decomposition for each frame runs in 6.27 *s*, except for the template model for which it runs in 25.52 *s* since

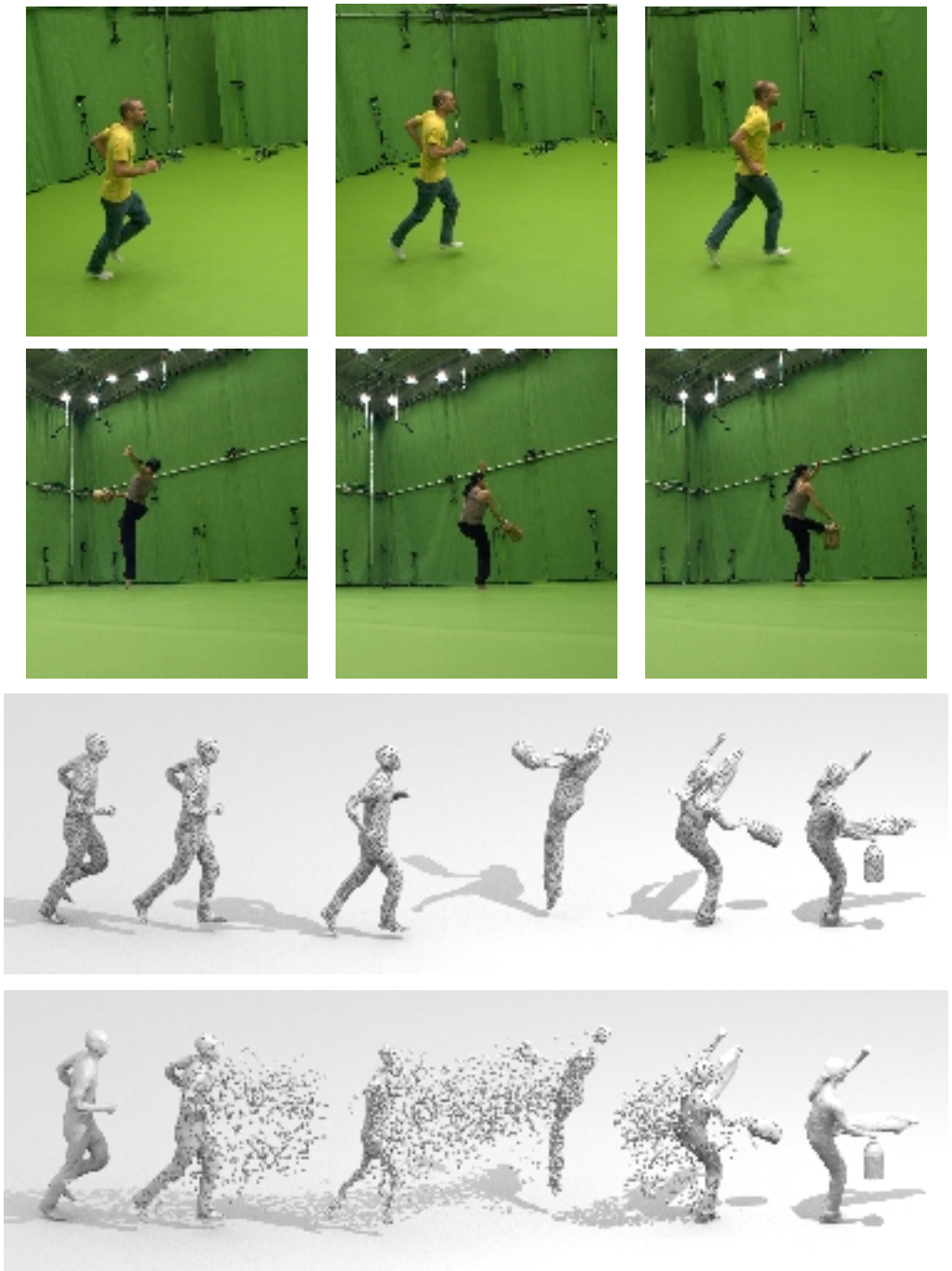


Figure 6.10 – Tracking result of the RUNNER and the CAGEBIRDDANCE sequences (middle) and combination with volumetric morphing with 5000 cells (bottom).

more iterations are applied. Our tracking algorithm runs in 45 *s* per frame on average. The physical simulation usually needs about 350 *ms* per simulation step on a single thread. The morphing runs on multiple threads in about 2.35 *s*.

6.2.8 Limitations

As shown in the previous examples, our approach generates plausible results for a variety of captured and simulated motions. However, a few limitations must be noted. First, the true captured shape must be volumetric in nature, since we tessellate its interior into 3D cells. Thin shapes such as clothes may cause some cells to be flat, leading to volume variation among cells and ill-defined cohesion constraints that would cause difficulty to the tracking model.

Regarding the physical simulation, our current demonstrator is limited to rigid body interactions, but could be extended to other physical models such as soft body physics and fluid simulation. Since a CVT provides neighboring information, soft body simulation could be achieved by introducing soft constraints between neighboring cells. This would lead to animations where cell sets behave more like a whole rather than independent bodies.

6.2.9 Conclusion

We have proposed a framework that allows video-based animations to be combined with physical simulation to create unprecedented and plausible animations. Our approach benefits of both regularity and uniformity of CVTs. Both volumetric shape tracking and animation generation in our framework rely on CVTs to perform well.

Chapter 7

Conclusion

Contents

In this thesis, we have focused on the problem of regular volumetric tessellation based on centroidal Voronoi tessellations. Our contributions are threefold: (1) propose criteria to evaluate quantitatively the regularity of volumetric tessellations; (2) introduce a novel algorithm that build CVTs with better regularity than the state-of-the-art methods; (3) extend CVT to implicit shapes and build an extensive comparison of voxel-, Delaunay- and Voronoi-based approaches for approximated volumetric tessellations.

This chapter is organized as follows: In Section 7.1, we summarize the contributions of this thesis. Future work perspectives inspired from the work done in this thesis are provided in Section 7.2.

7.1 Summary of Contributions

In Chapter 3, we have built a theoretical work on the regularity of volumetric tessellations. Based on the normalized second order moments, we have proposed criteria that allow to evaluate and compare the regularity of tessellations. We have also shown that these criteria are linked to the CVT energy function which allows to compute the value of the optimal CVT energy given a shape and a number of sites. We have also proposed a novel stopping criterion defined by this optimal value for the CVT algorithm as an application.

In Chapter 4, we have proposed a hierarchical algorithm for regular CVTs. The experimental results have shown that the subdivision schema

of our algorithm preserves the local optimality of CVTs on unbounded domains. With a relatively large number of inner cells than boundary cells, our algorithm builds more regular CVTs with better efficiency than the state-of-the-art algorithms.

In Chapter 5, we have extended the CVT algorithm to implicit shapes. In order to have a good approximation to the implicit shape, a local improvement has been proposed to clip the CVT. We have also presented an extensive comparison of the existing methods and ours in terms of accuracy, regularity and computational time. The experimental results have shown that our algorithm performs the best except for the computational time. This is because our algorithm includes an unavoidable step of Delaunay triangulation that is relatively costly compared to the Marching Cubes algorithm.

In Chapter 6, a novel algorithm has been proposed to clip Voronoi tessellations. The variant experimental results have shown its robustness. We have also proposed another application that is a shape animation framework using CVTs. This framework benefits from the properties of CVTs so that it allows to combine kinematic and physical constraints using rigid body simulation.

7.2 Future Research Perspectives

In the following, we gives some perspectives for further research based on the contributions in this thesis.

Theoretical Work on CVT

CVTs correspond to local minima of the energy function and an optimal CVT achieves the global minimum of this function. However, since the CVT energy function is non linear and non convex, it appears to be very difficult to find its global minimum. Lu et al. [2012] have proposed a stochastic approach that tries to find all local minima of the CVT energy function in order to find the global minimum. This approach can theoretically achieve the global minimum in infinite computational time which leads to inefficient computation in practice. According to our study on the relation between our regularity criteria and the energy function, we can compute the global minimum of this function for any three-dimensional shape. This allows to measure how close a CVT is to its optimal one. However, our work is based on Gersho's conjecture [Gersho, 1979] that has not been proven in three dimensions. We also assume that the effect of

shape boundary is negligible. It would be worth exploring this theoretical work. In practice, it would be also interesting to eliminate the effect of shape boundary or to determine the number of sites necessary to neglect this effect for a given shape.

Generalized CVT Algorithm

In this thesis, we have proposed a CVT algorithm for implicit shapes. Since this is the first algorithm to build CVTs of implicit shapes, many extensions such as weighted Voronoi tessellations and power diagrams can be investigated. Lévy and Liu [2010] have introduced a generalized CVT whose energy function uses a L_p distance instead of the Euclidean distance. This generalized distance allows to change the shape of CVT cells. It would be interesting to implement a version of this L_p CVT for implicit shapes.

Another direction is to improve the performance of our CVT algorithm. For instance, in this thesis, we have also proposed a hierarchical approach in order to build regular CVTs, this approach could be easily integrated into our CVT algorithm to improve the regularity of the generated CVTs. The experimental results have shown that our algorithm is less efficient than the Marching Cubes algorithm. As mentioned before, this is because of the unavoidable Delaunay triangulation in our algorithm. One way to reduce the computational time is to reduce the number of sites. We believe that applying our algorithm on a shape with non uniform density may achieve this purpose without losing the accuracy of generated CVTs. For instance, if we define the density near the boundary is much higher than the one far from the boundary, our algorithm would build CVTs with large inner cells and small boundary cells. Thus, we reduce the number of sites while keeping the accuracy of approximation to the shape.

Bibliography

D. Akio and A. Koide. An efficient method of triangulating equi-valued surfaces by using tetrahedral cells. *IEICE TRANSACTIONS on Information and Systems*, 74(1):214–224, 1991. Cited on page 12.

B. Allain, J.-S. Franco, and E. Boyer. An efficient volumetric framework for shape tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 268–276, 2015. Cited on pages 35, 36, 94, 96, 97, and 99.

B. Allain, L. Wang, J.-S. Franco, F. Hetroy, and E. Boyer. Shape animation with combined captured and simulated dynamics. *arXiv preprint arXiv:1601.01232*, 2016. Cited on pages 1, 4, and 35.

P. Alliez, D. Cohen-Steiner, M. Yvinec, and M. Desbrun. Variational tetrahedral meshing. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 617–625. ACM, 2005. Cited on page 25.

B. Anderson. http://www.cs.carleton.edu/cs_comps/0405/shape/marching_cubes.html. Cited on pages vi and 11.

F. Aurenhammer. Power diagrams: properties, algorithms and applications. *SIAM Journal on Computing*, 16(1):78–96, 1987. Cited on page 29.

F. Aurenhammer. Voronoi diagramsa survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991. Cited on pages 27 and 39.

D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 23–34. ACM, 1994. Cited on page 104.

D. Baraff. An introduction to physically based modeling: rigid body simulation iunconstrained rigid body dynamics. *SIGGRAPH Course Notes*, 1997. Cited on page 103.

E. Barnes and N. Sloane. The optimal lattice quantizer in three dimensions. *SIAM Journal on Algebraic Discrete Methods*, 4(1):30–41, 1983. Cited on page 40.

N. Barnes, G. Loy, and D. Shaw. The regular polygon detector. *Pattern Recognition*, 43(3):592–602, 2010. Cited on page 37.

M. Berger, J. A. Levine, L. G. Nonato, G. Taubin, and C. T. Silva. A benchmark for surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(2):20, 2013. Cited on pages ix, 70, 76, 77, and 84.

M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, and C. Silva. State of the art in surface reconstruction from point clouds. In *EUROGRAPHICS star reports*, volume 1, pages 161–185, 2014. Cited on page 8.

V. Blatov, L. Carlucci, G. Ciani, and D. Proserpio. Interpenetrating metal–organic and inorganic 3d networks: a computer-aided systematic investigation. part i. analysis of the cambridge structural database. *CrystEngComm*, 6(65):378–395, 2004. Cited on pages vii and 41.

J.-D. Boissonnat and S. Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451, 2005. Cited on page 76.

A. Bowyer. Computing dirichlet tessellations. *The Computer Journal*, 24 (2):162–166, 1981. Cited on page 18.

K. Q. Brown. Voronoi diagrams from convex hulls. *Information Processing Letters*, 9(5):223–228, 1979. Cited on page 17.

J. Bucklew. Upper bounds to the asymptotic performance of block quantizers. *Information Theory, IEEE Transactions on*, 27(5):577–581, 1981. Cited on page 39.

J. A. Bucklew and G. L. Wise. Multidimensional asymptotic quantization theory with¡ img src=”/images/tex/130. gif” alt=” r”¿ th power distortion measures. *Information Theory, IEEE Transactions on*, 28(2): 239–247, 1982. Cited on page 39.

H. Carr, T. Theußl, and T. Möller. Isosurfaces on optimal regular samples. In *ACM International Conference Proceeding Series*, volume 40, pages 39–48. Citeseer, 2003. Cited on page 12.

J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *ACM transactions on graphics (TOG)*, 22(3): 569–577, 2003. Cited on page 94.

D. Casas, M. Tejera, J.-Y. Guillemaut, and A. Hilton. Interactive animation of 4d performance capture. *Visualization and Computer Graphics, IEEE Transactions on*, 19(5):762–773, 2013. Cited on page 93.

T. J. Cashman and K. Hormann. A continuous, editable representation for deforming mesh sequences with separate signals for time, pose and shape. In *Computer Graphics Forum*, volume 31, pages 735–744. Wiley Online Library, 2012. Cited on page 93.

E. Catto. Iterative dynamics with temporal coherence. In *Game developer conference*, volume 2, page 5, 2005. Cited on pages 103 and 104.

CGAL. Computational Geometry Algorithms Library. http://www.cgal.org/. Cited on pages vi, ix, x, 23, 63, 75, 76, 84, 88, and 98.

R. Chalmeta, F. Hurtado, V. Sacristán, and M. Saumell. Measuring regularity of convex polygons. *Computer-Aided Design*, 45(2):93–104, 2013. Cited on page 37.

L. Chen and J.-c. Xu. Optimal delaunay triangulations. *Journal of Computational Mathematics*, pages 299–308, 2004. Cited on page 25.

X. Chen, A. Golovinskiy, and T. Funkhouser. A benchmark for 3d mesh segmentation. In *ACM Transactions on Graphics (TOG)*, volume 28, page 73. ACM, 2009. Cited on pages ix, 76, 79, and 84.

S.-W. Cheng, T. K. Dey, H. Edelsbrunner, M. A. Facello, and S.-H. Teng. Silver exudation. *Journal of the ACM (JACM)*, 47(5):883–904, 2000. Cited on pages 25 and 70.

S.-W. Cheng, T. K. Dey, and E. A. Ramos. Delaunay refinement for piecewise smooth complexes. *Discrete & Computational Geometry*, 43(1): 121–166, 2010. Cited on pages 23 and 74.

E. V. Chernyaev. Marching cubes 33: Construction of topologically correct isosurfaces. *Institute for High Energy Physics, Moscow, Russia, Report CN/95-17*, 42, 1995. Cited on pages x, 14, 15, and 98.

L. P. Chew. Constrained delaunay triangulations. *Algorithmica*, 4(1-4): 97–108, 1989. Cited on page 20.

L. P. Chew. Guaranteed-quality mesh generation for curved surfaces. In *Proceedings of the ninth annual symposium on Computational geometry*, pages 274–280. ACM, 1993. Cited on page 24.

P. Chew. There is a planar graph almost as good as the complete graph. In *Proceedings of the second annual symposium on Computational geometry*, pages 169–177. ACM, 1986. Cited on page 17.

P. Cignoni, C. Montani, and R. Scopigno. Dewall: A fast divide and conquer delaunay triangulation algorithm in e d. *Computer-Aided Design*, 30(5):333–341, 1998a. Cited on page 18.

P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. In *Computer Graphics Forum*, volume 17, pages 167–174. Wiley Online Library, 1998b. Cited on pages 70 and 77.

J. H. Conway and N. Sloane. Voronoi regions of lattices, second moments of polytopes, and quantization. *Information Theory, IEEE Transactions on*, 28(2):211–226, 1982. Cited on pages xi, 3, 38, 39, and 40.

E. Coumans et al. Bullet physics library. *Open source: bulletphysics. org*, 2013. Cited on page 103.

H. Coxeter. Regular skew polyhedra in three and four dimension, and their topological analogues. *Proceedings of the London Mathematical Society*, 2(1):33–62, 1938. Cited on page 37.

H. S. M. Coxeter. *Twelve geometric essays*. Southern Illinois Univ Pr, 1968. Cited on page 37.

M. Cubes. A high resolution 3d surface construction algorithm/william e. *Lorensen, Harvey E. Cline–SIG 87*, 1987. Cited on page 76.

L. Custodio, T. Etiene, S. Pesco, and C. Silva. Practical considerations on marching cubes 33 topological correctness. *Computers & Graphics*, 37 (7):840–850, 2013. Cited on page 15.

E. De Aguiar, C. Theobalt, C. Stoll, and H.-P. Seidel. Marker-less deformable mesh tracking for human shape and motion capture. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007. Cited on page 94.

E. De Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun. Performance capture from sparse multi-view video. *ACM Transactions on Graphics (TOG)*, 27(3):98, 2008. Cited on page 94.

B. de Araujo, D. S. Lopes, P. Jepp, J. A. Jorge, and B. Wyvill. A survey on implicit surface polygonization. *ACM Computing Surveys (CSUR)*, 47 (4):60, 2015. Cited on page 70.

B. Delaunay. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934. Cited on page 15.

Q. Du and M. Emelianenko. Acceleration schemes for computing centroidal voronoi tessellations. *Numerical linear algebra with applications*, 13(2-3):173, 2006. Cited on page 33.

Q. Du and D. Wang. Tetrahedral mesh generation and optimization based on centroidal voronoi tessellations. *International journal for numerical methods in engineering*, 56(9):1355–1373, 2003. Cited on page 25.

Q. Du, V. Faber, and M. Gunzburger. Centroidal voronoi tessellations: applications and algorithms. *SIAM review*, 41(4):637–676, 1999. Cited on pages 25, 29, 30, and 33.

M. J. Dürst. Re: additional reference to marching cubes. *ACM SIGGRAPH Computer Graphics*, 22(5):243, 1988. Cited on page 14.

H. Edelsbrunner and N. R. Shah. Triangulating topological spaces. In *Proceedings of the tenth annual symposium on Computational geometry*, pages 285–292. ACM, 1994. Cited on pages 31 and 85.

T. T. Elvins. A survey of algorithms for volume visualization. *ACM Siggraph Computer Graphics*, 26(3):194–201, 1992. Cited on page 12.

J. Erickson. Nice point sets can have nasty delaunay triangulations. In *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 96–105. ACM, 2001. Cited on pages 31 and 86.

T. Etiene, L. G. Nonato, C. Scheidegger, J. Tierny, T. J. Peters, V. Pascucci, R. M. Kirby, C. T. Silva, et al. Topology verification for isosurface extraction. *IEEE Transactions on Visualization and Computer Graphics*, 18(6):952–965, 2012. Cited on page 15.

M. A. Facello. Implementation of a randomized algorithm for delaunay and regular triangulations in three dimensions. *Computer Aided Geometric Design*, 12(4):349–370, 1995. Cited on page 18.

D. A. Field. Qualitative measures for initial meshes. *International Journal for Numerical Methods in Engineering*, 47(4):887–906, 2000. Cited on page 70.

M. A. Figueiredo. Scalar and vector quantization. Technical report, Tech. rep. Nov, 2008. Cited on page 38.

S. Fortune. Voronoi diagrams and delaunay triangulations. *Computing in Euclidean geometry*, 1:193–233, 1992. Cited on pages 17, 27, and 39.

S. Fuhrmann, M. Kazhdan, and M. Goesele. Accurate isosurface interpolation with hermite data. In *3D Vision (3DV), 2015 International Conference on*, pages 256–263. IEEE, 2015. Cited on pages 14 and 73.

Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, 32(8):1362–1376, 2010. Cited on page 76.

J. Gall, C. Stoll, E. De Aguiar, C. Theobalt, B. Rosenhahn, and H.-P. Seidel. Motion capture using joint skeleton tracking and surface estimation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1746–1753. IEEE, 2009. Cited on page 94.

A. Gersho. Asymptotically optimal block quantization. *Information Theory, IEEE Transactions on*, 25(4):373–380, 1979. Cited on pages 38, 39, and 114.

A. Gersho and R. M. Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012. Cited on page 38.

E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *Robotics and Automation, IEEE Journal of*, 4(2):193–203, 1988. Cited on pages 103 and 104.

M. Gross, S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Van Gool, S. Lang, et al. blue-c: a spatially immersive display and 3d video portal for telepresence. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 819–827. ACM, 2003. Cited on page 94.

R. Grosso. Construction of topologically correct and manifold isosurfaces. In *Computer Graphics Forum*, volume 35, pages 187–196. Wiley Online Library, 2016. Cited on page 15.

A. Hausner. Simulating decorative mosaics. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 573–580. ACM, 2001. Cited on pages 35 and 36.

C.-H. Huang, B. Allain, J.-S. Franco, N. Navab, S. Ilic, and E. Boyer. Volumetric 3d tracking by detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 2016. Cited on page 35.

C. Jamin, P. Alliez, M. Yvinec, and J.-D. Boissonnat. Cgalmesh: a generic framework for delaunay mesh generation. *ACM Transactions on Mathematical Software (TOMS)*, 41(4):23, 2015. Cited on pages 7, 8, 9, 20, and 75.

B. Joe. Construction of three-dimensional delaunay triangulations using local transformations. *Computer Aided Geometric Design*, 8(2):123–142, 1991. Cited on page 18.

L. Ju, T. Ringler, and M. Gunzburger. Voronoi tessellations and their application to climate and global modeling. In *Numerical techniques for global atmospheric models*, pages 313–342. Springer, 2011. Cited on page 35.

J. R. Kent, W. E. Carlson, and R. E. Parent. Shape transformation for polyhedral objects. In *ACM SIGGRAPH Computer Graphics*, volume 26, pages 47–54. ACM, 1992. Cited on page 109.

P. M. Knupp. Algebraic mesh quality metrics. *SIAM journal on scientific computing*, 23(1):193–218, 2001. Cited on page 70.

L. P. Kobbelt, M. Botsch, U. Schwanecke, and H.-P. Seidel. Feature sensitive surface extraction from volume data. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 57–66. ACM, 2001. Cited on page 74.

J. Kohout and I. Kolingerová. Parallel delaunay triangulation in e3: make it simple. *The Visual Computer*, 19(7-8):532–548, 2003. Cited on page 18.

J. Kohout, I. Kolingerová, and J. Žára. Parallel delaunay triangulation in e 2 and e 3 for computers with shared memory. *Parallel Computing*, 31 (5):491–522, 2005. Cited on page 18.

B. Lévy. Restricted voronoi diagrams for (re)-meshing surfaces and volumes. In *8th International Conference on Curves and Surfaces*, volume 6, page 14, 2014. Cited on pages 31, 68, 71, and 86.

B. Lévy. A numerical algorithm for l2 semi-discrete optimal transport in 3d. *ESAIM: Mathematical Modelling and Numerical Analysis*, 49(6): 1693–1715, 2015. Cited on page 29.

B. Lévy and Y. Liu. L p centroidal voronoi tessellation and its applications. In *ACM Transactions on Graphics (TOG)*, volume 29, page 119. ACM, 2010. Cited on pages 29, 68, and 115.

T. Lewiner, H. Lopes, A. W. Vieira, and G. Tavares. Efficient implementation of marching cubes' cases with topological guarantees. *Journal of graphics tools*, 8(2):1–15, 2003. Cited on pages 15 and 75.

M. Lhuillier. 2-manifold tests for 3d delaunay triangulation-based surface reconstruction. *Journal of Mathematical Imaging and Vision*, 51(1): 98–105, 2015. Cited on page 76.

X.-Y. Li. Spacing control and sliver-free delaunay mesh. In *IMR*, pages 295–306, 2000. Cited on page 25.

C. K. Liu, A. Hertzmann, and Z. Popović. Learning physics-based motion style with nonlinear inverse optimization. *ACM Transactions on Graphics (TOG)*, 24(3):1071–1081, 2005. Cited on page 93.

Y. Liu, W. Wang, B. Lévy, F. Sun, D.-M. Yan, L. Lu, and C. Yang. On centroidal voronoi tessellationenergy smoothness and fast computation. *ACM Transactions on Graphics (ToG)*, 28(4):101, 2009. Cited on pages 30, 33, 40, 53, 67, and 72.

S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. Cited on page 32.

S. Lo. Parallel delaunay triangulation in three dimensions. *Computer Methods in Applied Mechanics and Engineering*, 237:88–106, 2012. Cited on page 18.

J. A. Loera, J. Rambau, and F. Santos. Triangulations: Structures for algorithms and applications. *Algorithms and Computation*, 2010. Cited on page 17.

A. Lopes and K. Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):16–29, 2003. Cited on pages 13, 14, and 15.

W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *ACM siggraph computer graphics*, volume 21, pages 163–169. ACM, 1987. Cited on pages 7, 9, 11, and 12.

L. Lu, F. Sun, H. Pan, and W. Wang. Global optimization of centroidal voronoi tessellation with monte carlo approach. *IEEE transactions on visualization and computer graphics*, 18(11):1880–1890, 2012. Cited on pages viii, 30, 33, 49, 57, 58, 59, 63, and 114.

W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 369–374. ACM Press/Addison-Wesley Publishing Co., 2000. Cited on page 94.

M. Meyer, R. M. Kirby, and R. Whitaker. Topology, accuracy, and quality of isosurface meshes using dynamic particles. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1704–1711, 2007. Cited on page 74.

J. S. Mitchell. Geometric shortest paths and network optimization. *Handbook of computational geometry*, 334:633–702, 2000. Cited on page 17.

M. Moriguchi and K. Sugihara. A new initialization method for constructing centroidal voronoi tessellations on surface meshes. In *2006 3rd International Symposium on Voronoi Diagrams in Science and Engineering*, pages 159–165. IEEE, 2006. Cited on pages 30 and 49.

B. K. Natarajan. On generating topologically consistent isosurfaces from uniform samples. *The Visual Computer*, 11(1):52–62, 1994. Cited on page 14.

D. J. Newman. The hexagon theorem. *Information Theory, IEEE Transactions on*, 28(2):137–139, 1982. Cited on page 40.

T. S. Newman and H. Yi. A survey of the marching cubes algorithm. *Computers & Graphics*, 30(5):854–879, 2006. Cited on pages 12, 15, and 76.

G. M. Nielson. On marching cubes. *IEEE Transactions on visualization and computer graphics*, 9(3):283–297, 2003. Cited on pages 14 and 15.

G. M. Nielson and B. Hamann. The asymptotic decider: resolving the ambiguity in marching cubes. In *Proceedings of the 2nd conference on Visualization'91*, pages 83–91. IEEE Computer Society Press, 1991. Cited on page 14.

A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*, volume 501. John Wiley & Sons, 2009. Cited on pages 27, 29, and 39.

N. Okazaki and J. Nocedal. libLBFGS: a library of Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS), 2010. http://www.chokkan.org/software/liblbfgs/. Cited on page 63.

J. O'Rourke and J. E. Goodman. *Handbook of discrete and computational geometry*. CRC Press, 2004. Cited on page 18.

S. Oudot, L. Rineau, and M. Yvinec. Meshing volumes bounded by smooth surfaces. In *Proceedings of the 14th International Meshing Roundtable*, pages 203–219. Springer, 2005. Cited on page 24.

S. Y. Oudot. On the topology of the restricted delaunay triangulation and witness complex in higher dimensions. *arXiv preprint arXiv:0803.1296*, 2008. Cited on pages 24 and 76.

J.-P. Pons and J.-D. Boissonnat. Delaunay deformable models: Topology-adaptive meshes based on the restricted delaunay triangulation. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. Cited on page 76.

Z. Popović and A. Witkin. Physically based motion transformation. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 11–20. ACM Press/Addison-Wesley Publishing Co., 1999. Cited on page 93.

J. Quinn, F. Sun, F. C. Langbein, Y.-K. Lai, W. Wang, and R. R. Martin. Improved initialisation for centroidal voronoi tessellation and optimal delaunay triangulation. *Computer-Aided Design*, 44(11):1062–1071, 2012. Cited on pages viii, 30, 31, 49, 55, 57, 58, 60, 62, and 63.

R. Rabbitz. Fast collision detection of moving convex polyhedra. *Graphics Gems IV*, pages 83–109, 1994. Cited on page 103.

D. Raviv, M. M. Bronstein, A. M. Bronstein, and R. Kimmel. Volumetric heat kernel signatures. In *Proceedings of the ACM workshop on 3D object retrieval*, pages 39–44. ACM, 2010. Cited on pages 1, 10, 35, and 36.

T. Ringler, L. Ju, and M. Gunzburger. A multiresolution method for climate system modeling: application of spherical centroidal voronoi tessellations. *Ocean Dynamics*, 58(5-6):475–498, 2008. Cited on pages 35 and 36.

J. Ruppert. A delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of algorithms*, 18(3):548–585, 1995. Cited on page 24.

A. Safonova, J. K. Hodgins, and N. S. Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 514–521. ACM, 2004. Cited on page 93.

E. Schulte. 18 symmetry of polytopes and polyhedra. 2004. Cited on page 37.

D. Shaw, N. Barnes, et al. Regular polygon detection as an interest point operator for slam. In *Australasian Conference on Robotics and Automation*, 2004. Cited on page 37.

R. Shekhar, E. Fayyad, R. Yagel, and J. F. Cornhill. Octree-based decimation of marching cubes surfaces. In *Visualization'96. Proceedings.*, pages 335–342. IEEE, 1996. Cited on page 12.

J. R. Shewchuk. A condition guaranteeing the existence of higher-dimensional constrained delaunay triangulations. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 76–85. ACM, 1998a. Cited on pages 26, 31, and 86.

J. R. Shewchuk. Tetrahedral mesh generation by delaunay refinement. In *Proceedings of the fourteenth annual symposium on Computational geometry*, pages 86–95. ACM, 1998b. Cited on page 24.

J. R. Shewchuk. Updating and constructing constrained delaunay and constrained regular triangulations by flips. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 181–190. ACM, 2003. Cited on page 26.

J. R. Shewchuk. General-dimensional constrained delaunay and constrained regular triangulations, i: Combinatorial properties. *Discrete & Computational Geometry*, 39(1-3):580–637, 2008. Cited on pages 31 and 86.

J. R. Shewchuk and H. Si. Higher-quality tetrahedral mesh generation for domains with small angles by constrained delaunay refinement. In *Proceedings of the thirtieth annual symposium on Computational geometry*, page 290. ACM, 2014. Cited on page 27.

R. Shu, C. Zhou, and M. S. Kankanhalli. Adaptive marching cubes. *The Visual Computer*, 11(4):202–217, 1995. Cited on page 11.

H. Si. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Transactions on Mathematical Software (TOMS)*, 41(2):11, 2015. Cited on pages vi, 8, 9, and 26.

H. Si and K. Gärtner. Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proceedings of the 14th international meshing roundtable*, pages 147–163. Springer, 2005. Cited on page 26.

H. Si and K. Gärtner. 3d boundary recovery by constrained delaunay tetrahedralization. *International Journal for Numerical Methods in Engineering*, 85(11):1341–1364, 2011. Cited on page 26.

H. Si and J. R. Shewchuk. Incrementally constructing and updating constrained delaunay tetrahedralizations with finite-precision coordinates. *Engineering with Computers*, 30(2):253–269, 2014. Cited on page 26.

J. Starck and A. Hilton. Surface capture for performance-based animation. *Computer Graphics and Applications, IEEE*, 27(3):21–31, 2007. Cited on page 94.

M. Straka, S. Hauswiesner, M. Rüther, and H. Bischof. Simultaneous shape and pose adaption of articulated models using linear optimization. In *Computer Vision–ECCV 2012*, pages 724–737. Springer, 2012. Cited on page 94.

STUDYBLUE. http://www.studyblue.com. Cited on pages vii and 37.

A. Sulejmanpašić and J. Popović. Adaptation of performed ballistic motion. *ACM Transactions on Graphics (TOG)*, 24(1):165–179, 2005. Cited on page 93.

R. W. Sumner and J. Popović. Deformation transfer for triangle meshes. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 399–405. ACM, 2004. Cited on page 93.

I. E. Sutherland and G. W. Hodgman. Reentrant polygon clipping. *Communications of the ACM*, 17(1):32–42, 1974. Cited on pages 12, 31, and 86.

T. Takahashi, Y. Mekada, H. Murase, and T. Yonekura. High quality isosurface generation from volumetric data and its application to visualization of medical ct data. In *Pattern Recognition, 2004. ICPR 2004. Proceedings*

of the 17th International Conference on, volume 3, pages 734–737. IEEE, 2004. Cited on page 12.

J. Tournois, R. Srinivasan, and P. Alliez. Perturbing slivers in 3d delaunay meshes. In *Proceedings of the 18th international meshing roundtable*, pages 157–173. Springer, 2009. Cited on page 25.

T. Tung, S. Nobuhara, and T. Matsuyama. Complete multi-view reconstruction of dynamic scenes from probabilistic fusion of narrow and wide baseline stereo. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1709–1716. IEEE, 2009. Cited on page 94.

D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics (TOG)*, 27(3):97, 2008. Cited on page 94.

D. Vlasic, P. Peers, I. Baran, P. Debevec, J. Popović, S. Rusinkiewicz, and W. Matusik. Dynamic shape capture using multi-view photometric stereo. In *ACM Transactions on Graphics (TOG)*, volume 28, page 174. ACM, 2009. Cited on page 94.

L. Wang, F. Hétroy-Wheeler, and E. Boyer. A hierarchical approach for regular centroidal voronoi tessellations. In *Computer Graphics Forum*, volume 35, pages 152–165. Wiley Online Library, 2016a. Cited on pages 3, 4, 12, 33, and 71.

L. Wang, F. Hétroy-Wheeler, and E. Boyer. On volumetric shape reconstruction from implicit forms. In *ECCV 2016-European Conference on Computer Vision*, 2016b. Cited on pages 1, 4, and 31.

D. F. Watson. Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *The computer journal*, 24(2):167–172, 1981. Cited on page 18.

G. H. Weber, O. Kreylos, T. J. Ligocki, J. M. Shalf, H. Hagen, B. Hamann, and K. I. Joy. Extraction of crack-free isosurfaces from adaptive mesh refinement data. In *Hierarchical and Geometrical Methods in Scientific Visualization*, pages 19–40. Springer, 2003. Cited on page 12.

X. Wei, J. Min, and J. Chai. Physically valid statistical models for human motion generation. *ACM Transactions on Graphics (TOG)*, 30(3):19, 2011. Cited on page 93.

C. Wu, K. Varanasi, and C. Theobalt. Full body performance capture under uncontrolled and varying illumination: A shading-based approach. In *Computer Vision–ECCV 2012*, pages 757–770. Springer, 2012. Cited on page 94.

Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. Cited on pages 10 and 35.

Y. Yamada, S. Tazaki, and R. M. Gray. Asymptotic performance of block quantizers with difference distortion measures. *Information Theory, IEEE Transactions on*, 26(1):6–14, 1980. Cited on page 39.

D.-M. Yan, B. Lévy, Y. Liu, F. Sun, and W. Wang. Isotropic remeshing with fast and exact computation of restricted voronoi diagram. In *Computer graphics forum*, volume 28, pages 1445–1454. Wiley Online Library, 2009. Cited on pages 31 and 85.

D.-M. Yan, W. Wang, B. Lévy, and Y. Liu. Efficient computation of 3d clipped voronoi diagram. In *Advances in Geometric Modeling and Processing*, pages 269–282. Springer, 2010. Cited on page 85.

D.-M. Yan, W. Wang, B. Lévy, and Y. Liu. Efficient computation of clipped voronoi diagram for mesh generation. *Computer-Aided Design*, 45(4): 843–852, 2013. Cited on pages 8, 29, 31, 71, 78, and 81.

Y. Ye and C. K. Liu. Animating responsive characters with dynamic constraints in near-unactuated coordinates. In *ACM Transactions on Graphics (TOG)*, volume 27, page 112. ACM, 2008. Cited on page 93.

P. L. Zador. Topics in the asymptotic quantization of continuous random variables. *IEEE Trans. Inform. Theory*, 28(2):139–149, 1982. Cited on page 39.

A. Zaharescu, E. Boyer, and R. Horaud. Topology-adaptive mesh deformation for surface evolution, morphing, and multiview reconstruction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(4): 823–837, 2011. Cited on page 86.

V. B. Zordan and J. K. Hodgins. Motion capture-driven simulations that hit and react. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 89–96. ACM, 2002. Cited on page 93.

J. Zunic and P. L. Rosin. A new convexity measure for polygons. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(7):923–934, 2004. Cited on page 37.