



HAL
open science

Efficient computation of clipped Voronoi diagram for mesh generation

Dong-Ming Yan, Wenping Wang, Bruno Lévy, Yang Liu

► **To cite this version:**

Dong-Ming Yan, Wenping Wang, Bruno Lévy, Yang Liu. Efficient computation of clipped Voronoi diagram for mesh generation. *Computer-Aided Design*, 2011, <10.1016/j.cad.2011.09.004>. <hal-00647979>

HAL Id: hal-00647979

<https://inria.hal.science/hal-00647979v1>

Submitted on 5 Dec 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization

Efficient Computation of Clipped Voronoi Diagram for Mesh Generation

Dong-Ming Yan^{a,b,c}, Wenping Wang^a, Bruno Lévy^b, Yang Liu^{b,d}

^aDepartment of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong

^bProject ALICE, INRIA/LORIA, Campus scientifique 615, rue du Jardin Botanique, 54600, Villers les Nancy, France

^cGeometric Modeling and Scientific Visualization Center, KAUST, Thuwal 23955-6900, Kingdom of Saudi Arabia

^dMicrosoft Research Asia, Building 2, No. 5 Danling Street, Haidian District, Beijing, 100800, P.R. China

Abstract

The Voronoi diagram is a fundamental geometric structure widely used in various fields, especially in computer graphics and geometry computing. For a set of points in a compact domain (i.e. a bounded and closed 2D region or a 3D volume), some Voronoi cells of their Voronoi diagram are infinite or partially outside of the domain, but in practice only the parts of the cells inside the domain are needed, as when computing the centroidal Voronoi tessellation. Such a Voronoi diagram confined to a compact domain is called a clipped Voronoi diagram. We present an efficient algorithm to compute the clipped Voronoi diagram for a set of sites with respect to a compact 2D region or a 3D volume. We also apply the proposed method to optimal mesh generation based on the centroidal Voronoi tessellation.

Keywords: clipped Voronoi diagram, Delaunay triangulation, centroidal Voronoi tessellation, mesh generation.

1. Introduction

The Voronoi diagram is a fundamental geometric structure which has numerous applications in various fields, such as shape modeling, motion planning, scientific visualization, geography, chemistry, biology and so on.

Suppose that a set of sites in a compact domain in \mathbb{R}^d is given. Each site is associated with a Voronoi cell containing all the points in \mathbb{R}^d closer to the site than to any other sites; these cells constitute the Voronoi diagram of the set of sites. Voronoi cells of those sites on the convex hull are infinite, and some of Voronoi cells may be partially outside of the specified domain. However, in many applications one usually needs only the parts of Voronoi cells inside the specific domain. That is, the Voronoi diagram restricted to the given domain, which is defined as the intersection of the Voronoi diagram and the domain, and is therefore called the *clipped Voronoi diagram* [1]. The corresponding Voronoi cells are called the *clipped Voronoi cells* (see Figure 1).

Computing the clipped Voronoi diagram in a convex domain is relatively easy – one just needs to compute the intersection of each Voronoi cell and the domain, both being convex. However, directly

computing the clipped Voronoi diagram with respect to a complicated input domain is a difficult problem and there is no efficient solution in the existing literature. There has been no previous work on computing the exact clipped Voronoi diagram for non-convex domains with arbitrary topology. A brute-force implementation would be inefficient because of the complexity of the domain.

The motivation of the work is inspired by the recent work [2, 3]. They showed in [2] that the CVT energy function is C^2 -continuous, which can be minimized by the Newton-like algorithm, such as the L-BFGS method presented. In [3], an efficient CVT-based surface remeshing algorithm was presented with an exact algorithm for computing the restricted Voronoi diagram on mesh surfaces. In this paper, we aim at applying the fast CVT remeshing framework to 2D/3D mesh generation. To minimize the CVT energy function, one needs to compute the clipped Voronoi diagram in the input domain for function evaluation and gradient computation (see Section 2).

In this paper, we shall present practical algorithms for computing clipped Voronoi diagrams based on several simple operations. The main idea

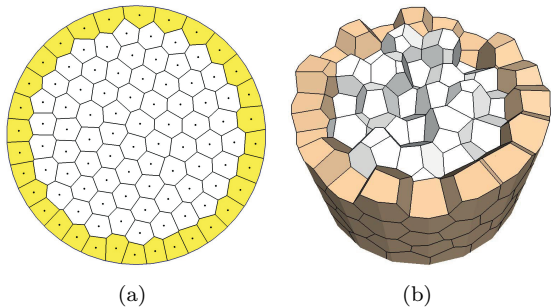


Figure 1: Examples of clipped Voronoi diagram in a circle (a) and a cylinder (b). The clipped Voronoi cells on the boundary are shaded.

of our approach is that instead of computing the intersection of Voronoi diagram and the domain directly, we first detect the Voronoi cells that have intersections with domain boundary and then apply computation for those cells only. We use a simple and efficient algorithm based on connectivity propagation for detecting the cells that intersect with the domain boundary (i.e., polygons in 2D and mesh surfaces in 3D, respectively). We also utilize the presented techniques for mesh generation as applications. The contributions of this paper include :

- introduce new methods for computing the clipped Voronoi diagram in 2D regions (Section 3) and 3D volumes (Section 4);
- present practical algorithms for 2D/3D mesh generation based on the presented clipped Voronoi diagram computation techniques (Section 5).

1.1. Previous work

The properties of the Voronoi diagram have been extensively studied in the past decades. Existing techniques compute the Voronoi diagram for point sites in 2D and 3D Euclidean spaces efficiently. There are several robust implementations that are publicly available, such as CGAL [4] and Qhull [5]. A thorough survey of the Voronoi diagram is out of the scope of this paper, the reader is referred to [6, 7, 8] for details of theories and applications of the Voronoi diagram. We shall restrict our discussion to the approaches of computing the Voronoi diagram restricted to a specific 2D/3D domain and their applications.

Voronoi diagram of surfaces/volumes. It is natural to use the geodesic metric to define the so called Geodesic Voronoi Diagram (GVD) on surfaces. Kunze *et al.* [9] presented a divide-and-conquer algorithm of computing GVD for parametric surfaces. Peyré and Cohen [10] used the fast marching algorithm to compute a discrete approximated GVD on a mesh surface. However, the cost of computing the exact GVD on surfaces is high, for instance, the fast marching method requires to solve the nonlinear Eikonal equation.

The restricted Voronoi diagram (RVD) [11] is defined as the intersection of the 3D Voronoi diagram and the surface, which is applied for computing constrained/restricted CVT on continuous surfaces by Du *et al.* [12]. The concept of the constrained CVT was extended to mesh surfaces in recent work [2, 3] and applied for isotropic surface remeshing. Yan *et al.* [3] proposed an exact algorithm to construct the RVD on mesh surfaces which consist of triangle soups. They processed each triangle independently where a kd -tree was used to find the nearest sites of each triangle in order to identify its incident Voronoi cells and compute the intersection. In this paper, we further improve the efficiency of the RVD computation by applying a neighbor propagation approach instead of using kd -tree query, assuming the availability of the mesh connectivity information (Section 4.1).

The clipped Voronoi diagram is defined as the intersection of the 3D (resp. 2D) Voronoi diagram and the given 3D volume (resp. a 2D region). Chan *et al.* [1] introduced an output-sensitive algorithm for constructing the 3D clipped Voronoi diagram of a convex polytope. Kyons *et al.* [13] presented an $O(n \log(n))$ algorithm to compute the clipped Voronoi diagram in a 2D square and applied it to network visualization. Yan *et al.* [14] utilized the clipped Voronoi diagram to compute the periodic CVT in 2D periodic space. Hudson *et al.* [15] computed the 3D clipped Voronoi diagram in the bounding box of the sites and used it to improve the time and space complexities of computing the full persistent homological information. However, the handling of non-convex objects was not addressed in these approaches. Existing algorithms used a discrete approximation in specific applications. Hoff III *et al.* [16] proposed a method for computing the discrete generalized Voronoi diagram using graphics hardware. The Voronoi diagram computation was formulated as a clustering problem in the discrete voxel/pixel space. Sud *et al.* [17] presented an

135 n -body proximity query algorithm based on com- 186
136 puting the discrete 2^{nd} order Voronoi diagram on 187
137 the GPU. GPU-based algorithms were fast but pro- 188
138 duced only a discrete approximation of the true 189
139 Voronoi diagram. In this paper, we shall present 190
140 efficient algorithms to compute the exact clipped 191
141 Voronoi diagram for both 2D and 3D domains. 192

142 *Mesh generation.* Mesh generation has been ex- 193
143 tensively studied in meshing community over past 194
144 decades. The detailed reviews of mesh genera- 195
145 tion techniques are available in [18, 19]. In the 196
146 following, we will focus on the work based on 197
147 Voronoi/Delaunay concepts, which are most related 198
148 to ours. We also briefly review the main categories 199
149 of tetrahedral mesh generation techniques. 200

150 The concept of Voronoi diagram has been suc- 201
151 cessfully used for meshing and analyzing point data. 202
152 Amenta *et al.* [20] presented a new surface recon- 203
153 struction algorithm based on Voronoi filtering. This 204
154 algorithm has provable guarantees when the sam- 205
155 ple points of a smooth surface satisfy the *lfs* (local 206
156 feature size) property. Alliez *et al.* [21] proposed a 207
157 surface reconstruction algorithm from noisy input 208
158 data based on the Voronoi-PCA estimation. Ley- 209
159 marie and Kimia introduced the medial scaffold of 210
160 point cloud data [22], which is a hierarchical rep- 211
161 resentation of the medial axis of 3D objects. Al- 212
162 though these works deal with point data, they can 213
163 be extended further for volumetric meshing. 214

164 The medial axis, which is a subset of Voronoi 215
165 diagram, has been applied in applications such as 216
166 2D quadrilateral meshing [23] and 3D hexahedral 217
167 meshing [24]. Given a closed 2D polygon or 3D 218
168 triangulated surface as the input domain, a set of 219
169 dense points is first sampled on the domain bound- 220
170 ary and the medial axis/surface is computed di- 221
171 rectly from the Voronoi diagram of samples. The 222
172 final mesh is generated by first meshing the medial 223
173 axis(2D)/surface(3D) and extruding to the domain 224
174 boundary [25]. The medial axis based method is 225
175 suitable for models which have well defined medial 226
176 axis, such as CAD/CAM models, but the medial 227
177 axis computation is sensitive to noise or small fea- 228
178 tures of the domain boundary. 229

179 In this paper, we focus on the tetrahedral mesh- 230
180 ing as an application of the clipped Voronoi diagram 231
181 computation (see Section 5). The shape quality and 232
182 boundary preservation are two main issues of tetra- 233
183 hedral meshing algorithms, since the quality of sim- 234
184 plices is crucial to finite element applications. We 235
185 refer the reader to [26] for the theoretic study of the

relationship between element qualities and interpo-
lation error/condition number. In the following, we
briefly discuss the main categories of tetrahedral
meshing.

- The *octree-based* approaches (e.g. [27, 28])
subdivide the bounding box of input model
repeatedly until a pre-specified resolution is
reached, then connect those cells to form the
tetrahedra. In general, this kind of approaches
cannot prevent bad elements near the bound-
ary.
- *Advancing front* methods start from the do-
main boundary and stuff the interior of the do-
main progressively, guided by specified heuristic
to control the shape/size. Advancing front
methods are fast but a high-quality triangulated
boundary is required.
- *Delaunay/Voronoi* based approaches generate
meshes satisfying Delaunay properties, which
maximize the minimal angle of shape elements.
Given an input domain, Delaunay/Voronoi
based methods repeatedly insert Steiner points
into the mesh, until all the elements meet the
Delaunay property. This approach aims at
generating meshes which conform to the input
domain boundary, but often leads to unsatis-
fied results if the given domain boundary is
poorly triangulated. An alternative way is to
approximate the boundary instead of conform-
ing, which results the better shape/size quality.
- *Variational approach* is one of the most ef-
fective ways of generating isotropic tetrahe-
dral meshes. Recent work includes both CVT-
based and ODT-based techniques. The CVT-
based approach aims at optimizing the dual
Voronoi structure of Delaunay triangulation,
while ODT tends to optimize the shape of pri-
mal elements [29]. The CVT-based mesh gen-
eration has been extensively studied in the lit-
erature [12], while ODT was recently intro-
duced to graphics community [30, 31]. One of
the main difficulties of both CVT and ODT-
based tetrahedral meshing is the boundary
conforming issue. Alliez *et al.* [30] used dense
quadrature samples to approximate restricted
Voronoi cells on mesh surface. Dardenne *et al.* [32]
used a discrete version of the CVT to generate
tetrahedral meshes from the discrete volume
data. The voxels are clustered into n

235 cells via Lloyd iteration, with each cell corre- 258
 236 sponding to a site. The tetrahedral mesh is ob- 259
 237 tained from the connectivity relations of cells. 260
 238 However, such an approach is limited to the 261
 239 resolution of voxels. 262

240 2. Problem Formulation 263

241 We first provide mathematical definitions and no- 264
 242 tations, then introduce the main idea of the clipped 265
 243 Voronoi diagram computation. 266

244 2.1. Definitions 267

Definition 2.1. *The Voronoi Diagram of a given 270
 271 set of distinct sites $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ in \mathbb{R}^d is defined by 272
 273 a collection of Voronoi cells $\{\Omega_i\}_{i=1}^n$, where 274*

$$\Omega_i = \{\mathbf{x} \in \mathbb{R}^d \mid \|\mathbf{x} - \mathbf{x}_i\| \leq \|\mathbf{x} - \mathbf{x}_j\|, \forall j \neq i\}.$$

245 Each Voronoi cell Ω_i is the intersection of a set of 275
 246 half-spaces, delimited by the bisectors of the Delaunay 276
 247 edges incident to the site \mathbf{x}_i . 277

Definition 2.2. *The Clipped Voronoi Diagram for 278
 279 the sites \mathbf{X} with respect to a connected compact domain 280
 281 Ω is the intersection of the Voronoi diagram 282
 283 and the domain, denoted as $\{\Omega_i|_{\Omega}\}_{i=1}^n$, where 284*

$$\Omega_i|_{\Omega} = \{\mathbf{x} \in \Omega \mid \|\mathbf{x} - \mathbf{x}_i\| \leq \|\mathbf{x} - \mathbf{x}_j\|, \forall j \neq i\}$$

248 Each clipped Voronoi cell is the intersection of the 285
 249 Voronoi cell Ω_i and the domain Ω , i.e., $\Omega_i|_{\Omega} = 286$
 250 $\Omega_i \cap \Omega$. We call $\Omega_i|_{\Omega}$ the *clipped Voronoi cell* with 287
 251 respect to Ω (see Figure 1 for examples). 288

Definition 2.3. *Centroidal Voronoi Tessellation 289
 290 of a set of distinct sites \mathbf{X} with respect to a com- 291
 292 pact domain Ω is the minimizer of the CVT energy 293
 294 function [33] :*

$$F(\mathbf{X}) = \sum_{i=1}^n \int_{\Omega_i|_{\Omega}} \rho(\mathbf{x}) \|\mathbf{x} - \mathbf{x}_i\|^2 d\sigma. \quad (1)$$

In the above definition, $\rho(\mathbf{x}) > 0$ is a user-defined 295
 296 density function. The partial derivative of the en- 297
 298 ergy function with respect to each site is given 299
 300 by [34] :

$$\frac{\partial F}{\partial \mathbf{x}_i} = 2m_i(\mathbf{x}_i - \mathbf{x}_i^*), \quad (2)$$

252 here $m_i = \int_{\Omega_i|_{\Omega}} \rho(\mathbf{x}) d\sigma$, and $\mathbf{x}_i^* = \frac{\int_{\Omega_i|_{\Omega}} \rho(\mathbf{x}) \mathbf{x} d\sigma}{\int_{\Omega_i|_{\Omega}} \rho(\mathbf{x}) d\sigma}$ 301
 253 is the centroid of the clipped Voronoi cell $\Omega_i|_{\Omega}$. 302
 254 We use the L-BFGS method [2] for computing the 303
 255 CVT. The clipped Voronoi diagram is used to assist 304
 256 the function evaluation (Eqn. 1) and the gradient 305
 257 computation (Eqn. 2). 306

258 2.2. Algorithm overview 261

There are two types of clipped Voronoi cells of a 262
 263 clipped Voronoi diagram : *inner Voronoi cells* and 264
 265 *boundary Voronoi cells*, whose corresponding sites 266
 267 are called inner sites and boundary sites, respec- 268
 269 tively. The *inner Voronoi cells* are entirely con- 270
 271 tained in the interior of the domain Ω , which can be 272
 273 deduced from the Delaunay triangulation directly. 274
 275 The *boundary Voronoi cells* are those cells that in- 276
 277 tersect with the domain boundary $\partial\Omega$, as shown in 278
 279 Figure 1. In the following, we will focus on how to 280
 281 compute the boundary Voronoi cells. 282

To compute a clipped Voronoi diagram with re- 283
 284 spect to a given domain, we first need to classify the 285
 286 sites into inner and boundary sites, and then com- 287
 288 pute the clipped Voronoi cells for boundary sites. 289
 290 As discussed above, the boundary cells have inter- 291
 292 sections with the domain boundary $\partial\Omega$ (i.e., poly- 293
 294 gons in 2D and mesh surfaces in 3D), which can be 294
 295 found by intersecting the boundary with the 295
 296 Voronoi diagram. We present efficient algorithms 296
 297 for computing the intersection of a Voronoi diagram 297
 298 and 2D polygons or 3D mesh surfaces, respectively. 298
 299 Once the boundary sites are identified, we are able 299
 300 to compute the clipped Voronoi cells efficiently by 300
 301 clipping the domain Ω against boundary Voronoi 301
 302 cells. 302

In the following sections, we shall present efficient 303
 304 algorithms for computing clipped Voronoi diagram 304
 305 in 2D (Section 3) and 3D (Section 4) spaces, respec- 305
 306 tively. Furthermore, we show how to utilize the pre- 306
 307 sented clipped Voronoi diagram computation techni- 307
 308 ques for practical mesh generation (Section 5). 308

291 3. 2D Clipped Voronoi Diagram Computa- 292 293 tion 294

Suppose that the input domain Ω is a compact 309
 310 2D region, whose boundary is represented by a 2D 310
 311 counter-clockwise outer polygon, and several clock- 311
 312 wise inner polygons without self-intersections. As- 312
 313 sume that the boundary is represented by a set of 313
 314 ordered edge segments $\{e_i\}$. The main steps of our 314
 315 method are illustrated in Figure 2. For a given set 315
 316 of sites inside the given domain, we first compute 316
 317 the Voronoi diagram of the sites. Then we identify 317
 318 the boundary sites and finally compute the clipped 318
 319 Voronoi cells of boundary sites . 319

314 3.1. Voronoi diagram construction 315

We first construct a Delaunay triangulation from 320
 321 input sites $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$. The corresponding 321

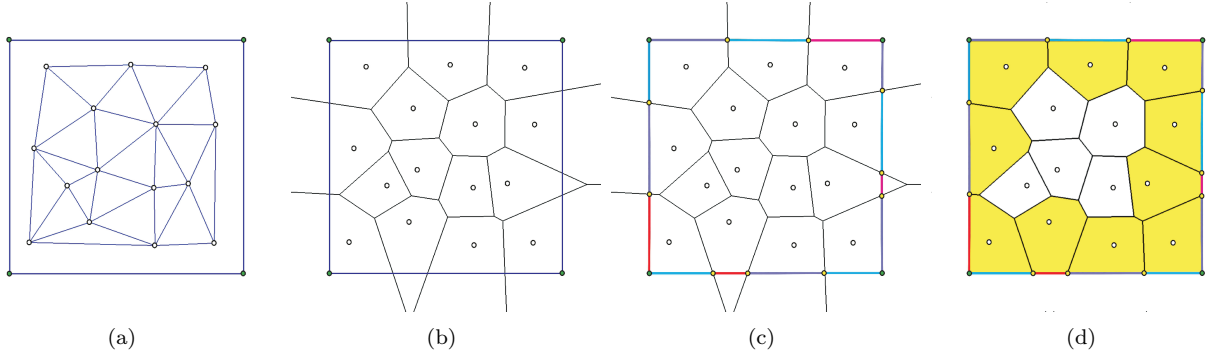


Figure 2: Illustration of main steps for computing clipped Voronoi diagram in 2D. (a) Delaunay triangulation, (b) 2D Voronoi diagram, (c) detect boundary sites, (d) compute clipped Voronoi diagram.

307 Voronoi diagram $\{\Omega_i\}_{i=1}^n$ is constructed as the dual
 308 of the Delaunay triangulation, as defined in Sec-
 309 tion 2. Each Voronoi cell is stored as a set of bi-
 310 secting planes, which is used for clipping operations
 311 in the following steps.

3.2. Detection of boundary cells

313 In this step, we shall identify the boundary
 314 Voronoi cells by computing the intersection of
 315 boundary edges and the Voronoi diagram $\{\Omega_i\}$. We
 316 repeatedly find the incident cell-edge pairs with the
 317 assistance of an FIFO queue. An incident Voronoi
 318 cell of a boundary edge e_i is the cell that intersects
 319 with e_i , i.e., a boundary Voronoi cell.

320 We assign a boolean tag to each boundary edge
 321 e_i which indicates whether e_i has been processed or
 322 not. This flag is initialized as **false**. Once the edge
 323 is visited, the flag is switched to **true**. Starting
 324 from an unvisited boundary edge e_i , we first find
 325 its nearest incident Voronoi cell Ω_j , then use the
 326 barycenter (or midpoint) of e_i to query the nearest
 327 site \mathbf{x}_j . Any linear search function can be used here
 328 for the nearest point query.

329 The FIFO queue is initialized by the initial inci-
 330 dent cell-edge pair (Ω_j, e_i) . We repeatedly pop out
 331 the cell-edge pair from the queue and compute the
 332 intersection of the current Voronoi cell Ω_c and the
 333 boundary edge e_c . The intersected segment is de-
 334 noted as s_c . The current boundary edge is marked
 335 as **visited** and the current Voronoi cell is marked
 336 as **boundarycell**. We detect new cell-edge pairs
 337 by examining the current intersected segment s_c .
 338 There are two cases of s_c 's endpoints :

- 339 (a) if the endpoints of s_c contain a boundary ver-
 340 tex of the current edge e_c (green dots in Fig-
 341 ure 2(c)), the adjacent boundary edge who

342 shares the same vertex with e_c is pushed into
 343 the queue together with the current Voronoi
 344 cell Ω_c ;

- 345 (b) if the endpoints of s_c contain an intersection
 346 point, i.e., the intersection point between a
 347 Voronoi edge of Ω_c and e_c (yellow dots in Fig-
 348 ure 2(c)), the neighboring Voronoi cell who
 349 shares the intersecting Voronoi edge with Ω_c
 350 is pushed into the queue together with e_c .

351 The boundary detection process terminates when
 352 all the edges have been visited.

3.3. Computation of clipped Voronoi cells

354 Once the boundary sites are identified, we com-
 355 pute the clipped Voronoi cells by clipping the do-
 356 main against their corresponding bounding line seg-
 357 ments. A straightforward extension of [3] should
 358 first triangulate the boundary polygons and then
 359 do computation on the resulting planar mesh, which
 360 will be the same as the surface RVD computation
 361 described in Section 4.1. Given that the average
 362 number of bisectors of 2D Voronoi cells is six [33], it
 363 is efficient enough to clip the 2D domain by Voronoi
 364 cells directly. Here we simply use the Sutherland-
 365 Hodgman clipping algorithm [35] to compute the
 366 intersection. More examples of 2D clipped Voronoi
 367 diagram are given in Section 6.

4. 3D Clipped Voronoi Diagram Computa- tion

370 In this section we describe an efficient algorithm
 371 for computing the clipped Voronoi diagram of 3D
 372 objects. Suppose that the input volume Ω is given
 373 by a tetrahedral mesh $\mathcal{M} = \{\mathcal{V}, \mathcal{T}\}$, where $\mathcal{V} =$

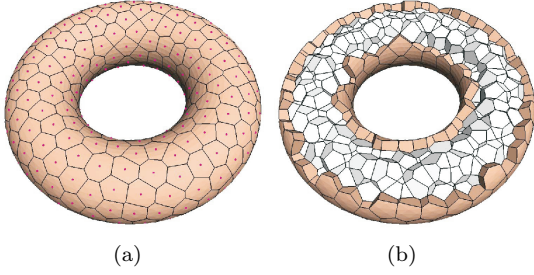


Figure 3: Illustration of clipped Voronoi diagram computation of 500 sites in a torus. (a) Surface RVD of 227 boundary sites, (b) Clipped Voronoi diagram.

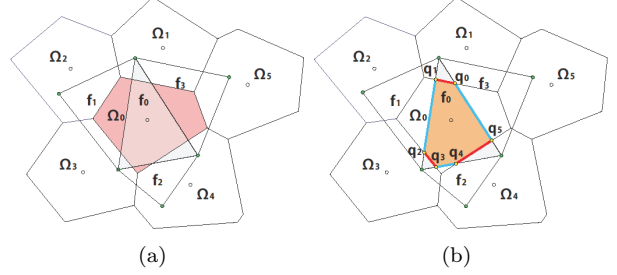


Figure 4: Illustration of the propagation process. The green points are the vertices of input boundary mesh and the white points are the sites. The yellow points in (b) are the vertices of RVD.

374 $\{\mathbf{v}_k\}_{k=1}^{n_v}$ is the set of mesh vertices and $\mathcal{T} = \{\mathbf{t}_i\}_{i=1}^m$ 408
 375 the set of tetrahedral elements. Each tetrahedron 409
 376 (*tet* for short in the following) \mathbf{t}_i stores the informa- 410
 377 tion of its four incident vertices and four adjacent 411
 378 tets. The four vertices are assigned indices 0, 1, 2, 3 412
 379 and so are the four adjacent tets. The index of an 413
 380 adjacent tet is the same as the index of the vertex 414
 381 which is opposite to the tet. The boundary of \mathcal{M} is 415
 382 a triangle mesh, denoted as $\mathcal{S} = \{\mathbf{f}_j\}_{j=1}^{n_f}$, which is 416
 383 assumed to be a 2-manifold. Each boundary trian- 417
 384 gular facet \mathbf{f}_j stores the indices of three neighbor- 418
 385 ing facets and the index of its containing tet. Note that 419
 386 although other types of convex primitives can also 420
 387 be used for domain decomposition, we use tetrahe- 421
 388 dral mesh here for simplicity. 422

389 The 3D clipped Voronoi diagram computation is 423
 390 similar to the 2D counterpart. After constructing 424
 391 the 3D Voronoi diagram $\{\Omega_i\}$ of the sites \mathbf{X} (see 425
 392 Section 3.1), there are two main steps, as illustrated 426
 393 in Figure 3 : 427

- 394 1. detect boundary sites by intersecting Voronoi 428
 395 diagram with the boundary surface \mathcal{S} , i.e., 429
 396 compute the surface RVD (Section 4.1); 430
- 397 2. compute the clipped Voronoi cells for all the 431
 398 boundary sites (Section 4.2). 432

399 4.1. Detection of boundary sites 433

400 For the given set of sites $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ and 437
 401 the boundary surface $\mathcal{S} = \{\mathbf{f}_j\}_{j=1}^{n_f}$, the restricted 438
 402 Voronoi diagram (RVD) is defined as the intersec- 439
 403 tion of the 3D Voronoi diagram and the surface \mathcal{S} , 440
 404 denoted as $\mathcal{R} = \{\mathcal{R}_i\}_{i=1}^n$, where $\mathcal{R}_i = \Omega_i \cap \mathcal{S}$ [11]. 441
 405 Each \mathcal{R}_i is called a *restricted Voronoi cell* (RVC). 442
 406 The sites corresponding to non-empty RVCs are re- 443
 407 garded as boundary sites. 444

We use the algorithm presented in [3] for comput-
 ing the surface RVD. The performance of RVD com-
 putation is improved by using a neighbor propaga-
 tion approach for finding the incident cell-triangles
 pairs, instead of using a *kd*-tree structure to query
 the nearest site for each triangle, as shown by our
 tests.

Now we are going to explain the propagation step
 (refer to Figure 4). We assign a boolean flag (initial-
 ized as `false`) for each boundary triangle at the ini-
 tialization step. The flag is used to indicate whether
 a triangle is processed or not. Starting from an
 unprocessed triangle and one of its incident cells,
 which is the cell corresponding to the nearest site of
 the triangle by using the barycenter of the triangle
 as the query point. Here we assume that a triangle
 \mathbf{f}_0 on \mathcal{S} is the unprocessed triangle and the Voronoi
 cell Ω_0 is the corresponding cell of the nearest site
 of \mathbf{f}_0 , as shown in Figure 4(a). We use an FIFO
 queue \mathcal{Q} to store all the incident cell-triangle pairs
 to be processed. To start, the initial pair $\{\mathbf{f}_0, \Omega_0\}$
 is pushed into the queue. The algorithm repeatedly
 pops out the pair in the front of \mathcal{Q} and computes
 their intersection. During the intersection process,
 the current triangle is marked as `processed`, new
 valid pairs are identified and pushed back into \mathcal{Q} .
 The process terminates when \mathcal{Q} is empty and all
 the triangles are processed.

The key issue now is how to identify all the valid
 cell-triangle pairs during the intersection. Assume
 that $\{\mathbf{f}_0, \Omega_0\}$ is popped out from \mathcal{Q} , as shown in Fig-
 ure 4. In this case, we clip \mathbf{f}_0 against the bound-
 ing planes of Ω_0 , which has five bisecting planes,
 i.e., $[\mathbf{x}_0, \mathbf{x}_1]$, $[\mathbf{x}_0, \mathbf{x}_2]$, ..., $[\mathbf{x}_0, \mathbf{x}_5]$. The resulting poly-
 gon is represented by $\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_5$, as shown in Fig-
 ure 4(b). Since the line segment $\overline{\mathbf{q}_0\mathbf{q}_1}$ is the inter-
 section of \mathbf{f}_0 and $[\mathbf{x}_0, \mathbf{x}_1]$, we know that the oppo-

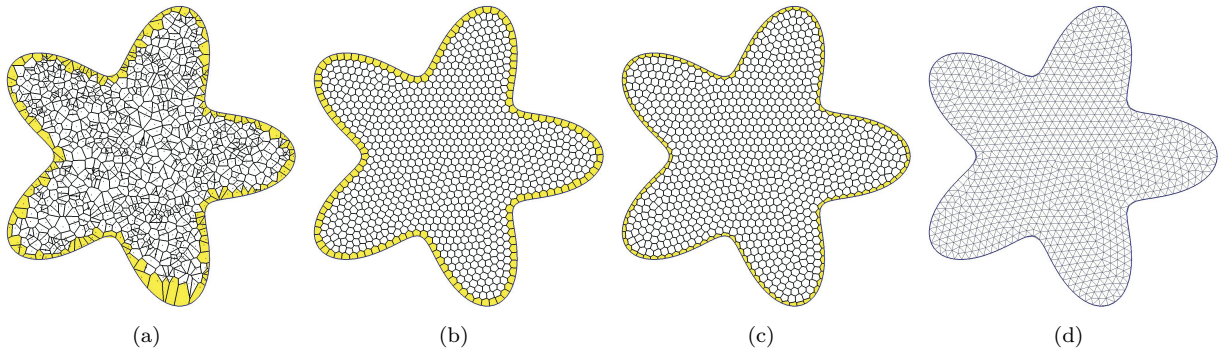


Figure 5: 2D CVT-based meshing. (a) The clipped Voronoi diagram of initial sites; (b) the result of CVT with $\rho = 1$; (c) the result of constrained optimization. Notice that boundary seeds are constrained on the border; (d) the final uniform 2D meshing.

445 site cell Ω_1 is also an incident cell of \mathbf{f}_0 , thus the 479
 446 pair $\{\mathbf{f}_0, \Omega_1\}$ is an incident pair. Since the com- 480
 447 mon edge of $[\mathbf{f}_0, \mathbf{f}_1]$ has intersection with Ω_0 , the 481
 448 adjacent facet \mathbf{f}_1 also has intersection with cell Ω_0 , 482
 449 thus the pair $\{\mathbf{f}_1, \Omega_0\}$ is also an incident pair. So 483
 450 is the pair $\{\mathbf{f}_2, \Omega_0\}$. The other incident pairs are 484
 451 found in the same manner. To keep the same pair 485
 452 from being processed multiple times, we store the 486
 453 incident facet indices for each cell. Before pushing 487
 454 a new pair into the queue, we add the facet 488
 455 index to the incident facet index set of the cell. The 489
 456 pair is pushed into the queue only if the facet is 490
 457 not contained in the incident facet set of the cell; 491
 458 otherwise the pair is discarded. At each time after 492
 459 intersection computation, the resulting polygon is 493
 460 associated with the surface RVC of the current site. 494
 461 The surface RVD computation terminates when the 495
 462 queue is empty. Those sites that have non-empty 496
 463 surface RVC are marked as the boundary sites, de- 497
 464 noted as $\mathbf{X}_b = \{\mathbf{x}_i | \mathcal{R}_i \neq \emptyset\}$.

465 4.2. Construction of clipped Voronoi cells

466 Once the boundary sites \mathbf{X}_b are found, we com-
 467 pute the clipped Voronoi cells for these sites. The
 468 computation of boundary Voronoi cells is similar
 469 to the surface RVD computation presented in Sec-
 470 tion 4.1, with the difference that we restrict the
 471 computation on boundary cells only. For each
 472 boundary cell, we have recorded the indices of its in-
 473 cident boundary triangles. We know that the neigh-
 474 boring tet of each boundary triangle is also incident
 475 to the cell. We also store the indices of the incident
 476 tet for each boundary cell. The incident tet set
 477 is initialized as the neighboring tet of the incident
 478 boundary triangle.

We use an FIFO queue to facilitate this process.
 The queue is initialized by a set of incident cell-
 tet pairs (Ω_i, \mathbf{t}_j) , which can be obtained from the
 boundary cell and its initial incident tet set.

The pair (Ω_i, \mathbf{t}_j) in front of \mathcal{Q} is popped out re-
 peatedly. We compute the intersection of Ω_i and
 \mathbf{t}_j again by the Sutherland-Hodgman clipping al-
 gorithm [35] and identify new incident pairs at the
 same time. We clip the tet \mathbf{t}_j by bounding planes
 of cell Ω_i one by one. If the current bounding
 plane has intersection with \mathbf{t}_j , we check the oppo-
 site Voronoi cell Ω_o that shares the current bisect-
 ing plane with Ω_i ; if Ω_o is a boundary cell and \mathbf{t}_j
 is not in the incident set of Ω_o , a new pair (Ω_o, \mathbf{t}_j)
 is found. We also check the neighboring tets who
 share the facets clipped by the current bisecting
 plane. Those tets that are not in the incident set of
 Ω_i are added to its set, and new pairs are pushed
 into the queue. After clipping, the resulting poly-
 hedron is associated with the clipped Voronoi cell
 $\Omega_i|_{\mathcal{M}}$ of site \mathbf{x}_i . This process terminates when \mathcal{Q}
 is empty.

501 5. Applications for mesh generation

We present two applications of the presented
 clipped Voronoi diagram computation techniques,
 including 2D triangular meshing and 3D tetrahe-
 dral meshing.

506 5.1. 2D mesh generation

Triangle mesh generation is a well-known appli-
 cation of CVT optimization. In this section we
 present such an application based on our 2D clipped
 Voronoi diagram computation. The input domain

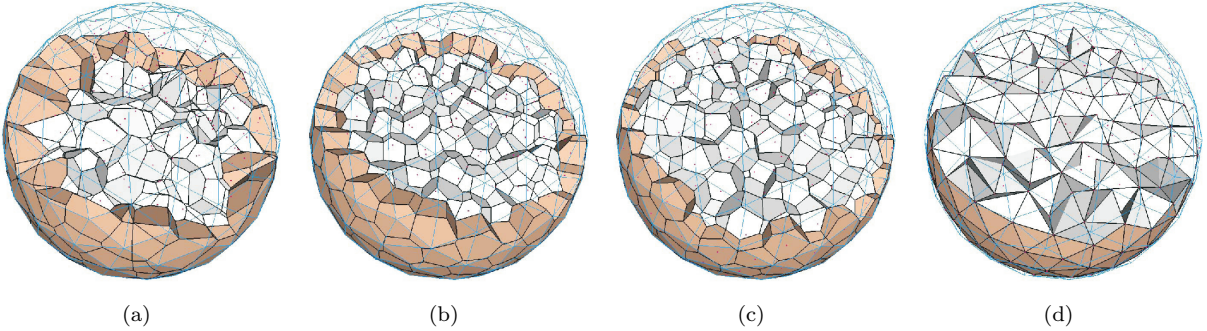


Figure 6: Illustration of the CVT-based tetrahedral meshing algorithm. The wireframe is the boundary of the input mesh. (a) The clipped Voronoi diagram of the initial sites (the boundary Voronoi cells are shaded); (b) the result of the unconstrained CVT with $\rho = 1$; (c) the result of the constrained optimization. Notice that boundary seeds are constrained on the surface \mathcal{S} ; (d) the final isotropic tetrahedral meshing result.

511 Ω is a 2D polygon, which can be single connected or 543
 512 with multiple components. We first sample a set of 544
 513 initial points inside the input domain (Figure 5(a)) 545
 514 and then compute a CVT (Eqn. 1) from this initial 546
 515 sampling (Figure 5(b)). Once we have a set of well 547
 516 distributed samples, we snap the seeds correspond-
 517 ing to boundary Voronoi cells to the boundary and
 518 run optimization again, with the boundary seeds
 519 constrained on the border (Figure 5(c)). Finally,
 520 we keep the primal triangles whose circumscribing
 521 centers are inside the domain as the meshing re-
 522 sult (5(d)). Our 2D meshing framework also allows
 523 the user to insert vertices of input polygon and tag
 524 these vertices as fixed. By doing this, the geomet-
 525 ric properties of the input domain can be better
 526 preserved. More results are given in Section 6.

527 5.2. Tetrahedral mesh generation

528 There are three main steps of the CVT-based 544
 529 meshing framework: initialization, iterative opti- 545
 530 mization, and mesh extraction, which are illus- 546
 531 trated by the example in Figure 6.

532 **Initialization.** In this step, we build a uniform 548
 533 grid to store the sizing field for adaptive meshing. 549
 534 Following the approach in [30], we first compute 550
 535 the *local feature size* (lfs) for all boundary vertices 551
 536 and then use a fast matching method to construct 552
 537 a sizing field on the grid. This grid is also used for 553
 538 efficient initial sampling (Figure 6(a)). The reader 554
 539 is referred to [30] for details.

540 **Optimization.** There are two phases of the global 548
 541 optimization: the unconstrained CVT optimization 549
 542 and the constrained CVT optimization. In the first 550

543 phase, we optimize the positions of the sites inside 544
 545 the input volume without any constraints, which 546
 547 yields a well-spaced distribution of the sites within 548
 549 the domain, with no sites lying on the boundary 550
 551 surface (Figure 6(b)).

552 During the second phase of optimization, all the 553
 554 boundary sites will be constrained on the boundary. 555
 556 The partial derivative of the energy function with 557
 558 respect to each boundary site is computed as:

$$\frac{\partial F}{\partial \mathbf{x}_i} \Big|_{\mathcal{S}} = \frac{\partial F}{\partial \mathbf{x}_i} - \left[\frac{\partial F}{\partial \mathbf{x}_i} \cdot \mathbf{N}(\mathbf{x}_i) \right] \mathbf{N}(\mathbf{x}_i), \quad (3)$$

559 where $\mathbf{N}(\mathbf{x}_i)$ is the unit normal vector of the bound- 560
 561 ary surface at the boundary site \mathbf{x}_i [2]. The partial 562
 563 derivative with respect to an inner site is still com- 564
 565 puted by Eqn. 2. Both boundary and inner sites 566
 567 will be optimized simultaneously, applying again 568
 569 the L-BFGS method to minimize the CVT energy 570
 571 function (Figure 6(c)).

572 Sharp features are preserved in a similar way as 573
 574 how the boundary sites are treated. For example, 575
 576 we project sites on sharp edges on the boundary and 577
 578 allow them to vary only along these edges during 579
 580 the second stage of optimization. For details, please 581
 582 refer to [3] where these steps are described in the 583
 584 context of surface remeshing.

585 **Final mesh extraction.** Once the optimization is 586
 587 finished, we extract the tetrahedral cells from the 588
 589 primal Delaunay triangulation (Figure 6(d)). As 590
 591 discussed in [30], the CVT energy cannot eliminate 592
 593 the slivers from the resulting tetrahedral mesh. We 594
 595 perform a post-processing to perturb slivers using 596
 597 the approach of [36]. The results are given in Sec- 598
 599 tion 6.



Figure 7: Results of clipped Voronoi diagram computation.

6. Experimental results

Our algorithm is implemented in C++ on both Windows and Linux platform. We use the CGAL library [4] for 2D and 3D Delaunay triangulation and TetGen [37] for background mesh generation when the input 3D domain is given as a closed triangle mesh. All the experimental results are tested on a laptop with 2.4GHz processor and 2GB memory.

Efficiency. We first demonstrate the performance of the proposed clipped VD computation algorithm. The 2D version is very efficient. All the examples shown in this paper take only several milliseconds. To detect the boundary sites, we have implemented a propagation based approach for surface RVD computation. This new implementation of RVD performs better than the previous kd -tree based approach [3] since there is no kd -tree query required, as shown in Figure 8. The performance of the 3D clipped Voronoi diagram computation is demonstrated in Figure 9. We progressively sample the input domain with number of sites from 10 to 6×10^5 . Note that the time of surface RVD computation is much less than the Delaunay triangulation, since only a small portion of all the sites are boundary sites. The time cost of the clipped VD computation algorithm is proportional to the total number of incident cell-tet pairs (Section 4.2). Therefore, an input mesh with a small number of tetrahedral elements would help to improve the efficiency. In our experiments, all the input tetrahedral

meshes are generated by the robust meshing software TetGen [37] with the conforming boundary. More results of the clipped Voronoi diagram computation of various 3D objects are given in Figure 7 and the timing statistics is given in Table 1.

Model	$ \mathcal{T} $	$ \mathcal{S} $	$ \mathbf{X} $	$ \mathbf{X}_b $	Time
Twoprism	68	30	1k	572	0.2
Bunny	10k	3k	2k	734	1.8
Elk	34.8k	10.4k	2k	1,173	3.1
Block	77.2k	23.4	1k	659	4.7
Homer	16.2k	4,594	10k	2,797	6.3
Rockerarm	212k	60.3k	3k	1,722	12.1
Bust	68.5k	20k	30k	5k	16.2

Table 1: Statistics of clipped Voronoi diagram computation on various models. $|\mathcal{T}|$ is the number of the input tetrahedra. $|\mathcal{S}|$ is the number of the boundary triangles. $|\mathbf{X}|$ is the number of the sites. $|\mathbf{X}_b|$ is the number of the boundary sites. Time (in seconds) is the total time for clipped Voronoi diagram computation, including both Delaunay triangulation and surface RVD computation.

Robustness. We use exact predicates to predicate the side of a vertex against a Voronoi plane during the clipping process. We use Meyer and Pion’s FGP predicate generator [38] provided by CGAL in our implementation, as also done in [3]. We did not encounter any numerical issue for all the examples shown in the paper. Our clipped Voronoi diagram is robust even for extreme configurations. We show an example of computing the clipped Voronoi diagram on a sphere in Figure 10. The sites are set to the vertices of the boundary mesh and there is no

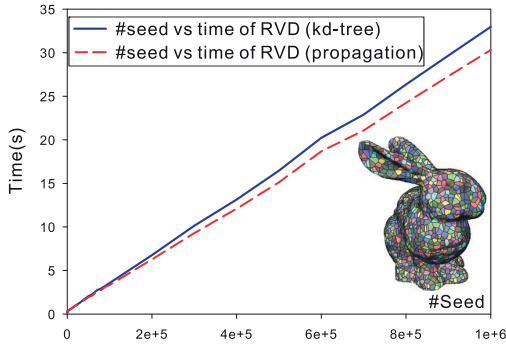


Figure 8: Comparison of the propagation-based surface RVD computation with the kd-tree-based approach.

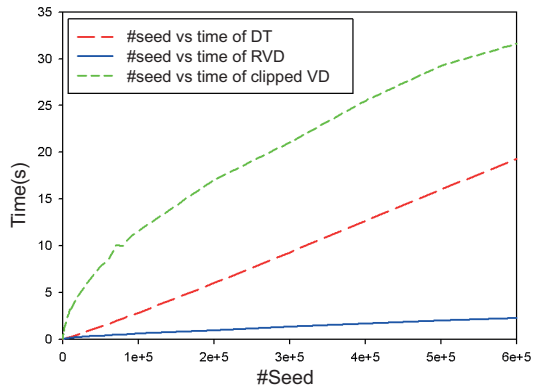


Figure 9: The timing curve of the clipped Voronoi diagram computation against the number of sites on Bone model.

616 inner site. Furthermore, we give another example
 617 of computing clipped Voronoi diagram in a cubic
 618 domain. The boundary mesh of the cube is shown
 619 in Figure 11(a). We sample the eight corners of
 620 the cube as sites, in this case, the bounding planes
 621 of Voronoi diagram are passing through the edges
 622 of the boundary mesh. The surface RVD and the
 623 volume clipped Voronoi diagram are shown in Fig-
 624 ure 11(b) and (c), respectively.

625 **2D meshing.** We show some 2D mesh generation
 626 results based on our fast clipped Voronoi diagram
 627 computation. Figure 12 demonstrates that our al-
 628 gorithm works well for multiple connected domains.
 629 Figure 13 shows that we insert original vertices of
 630 input polygon for the better preservation of the ge-
 631 ometric properties.

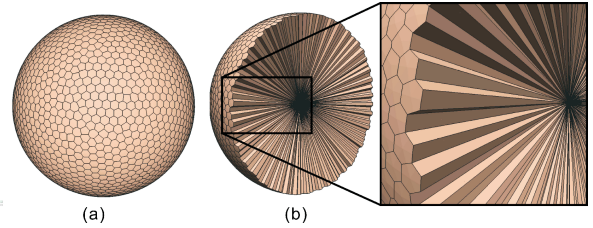


Figure 10: Clipped Voronoi diagram of a sphere. The sites are the vertices of the sphere. (a) The surface RVD, (b) the clipped Voronoi diagram.

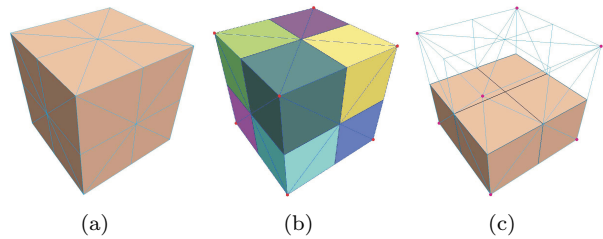


Figure 11: Clipped Voronoi diagram of a cube. Red points represent the sites. (a) The input domain, (b) the surface RVD, (c) the clipped Voronoi diagram.

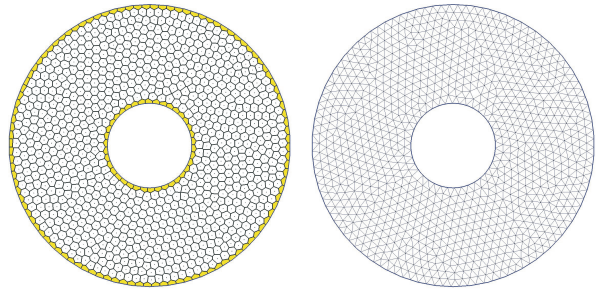


Figure 12: CVT-based 2D mesh generation of a ring.

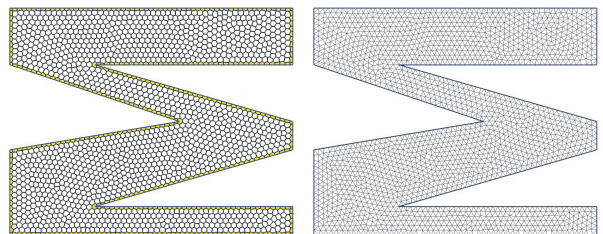


Figure 13: CVT-based 2D mesh generation. The boundary vertices of the input domain are used as constraints.

632 **Tetrahedral meshing.** The complete process of
 633 the proposed tetrahedral meshing framework is il-
 634 lustrated in Figure 6. Figure 14 (a)&(b) show two
 635 adaptive tetrahedral meshing examples, using lfs
 636 as the density function [30]. Figure 14 (c)&(d) give
 637 two examples with sharp features preserved. Our
 638 framework can generate high quality meshes effi-
 639 ciently and robustly. The running time for obtain-
 640 ing final results ranges from seconds to minutes, de-
 641 pending on the size of the input tetrahedral mesh
 642 and the desired number of sites.

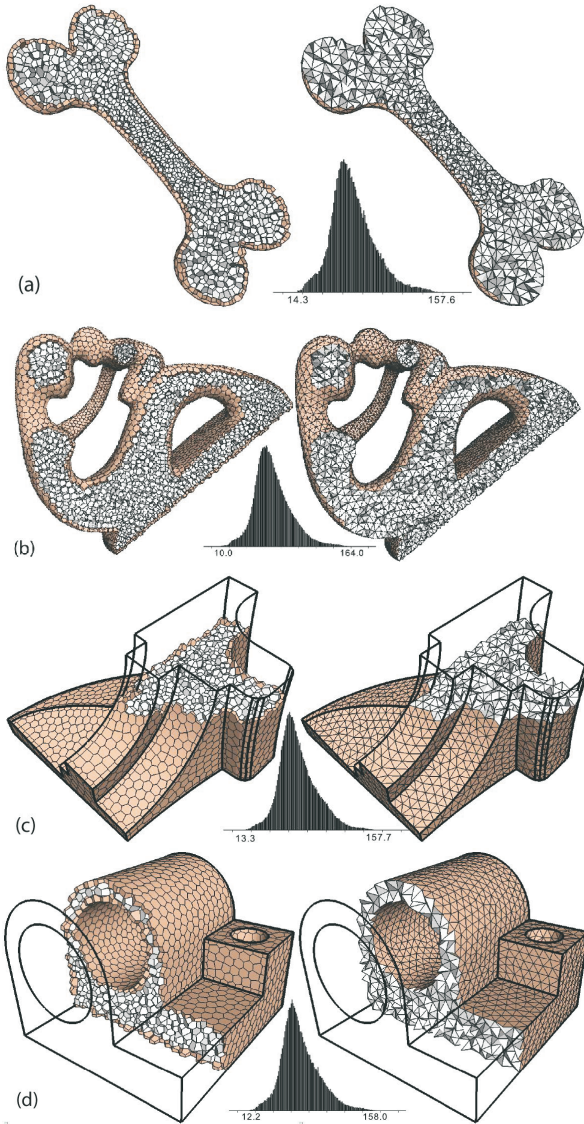


Figure 14: Tetrahedral mesh generation results. The histograms show the angle distribution of the results.

643 **Comparison.** We compare our meshing results
 644 with the Delaunay refinement approach provided
 645 by CGAL [4], as well as a recent work that used
 646 a discrete version of clipped Voronoi diagram for
 647 tetrahedral mesh generation [32]. Four shape qual-
 648 ity measurements are used as in [32], i.e.,

- 649 • $Q_1 = \theta_{min}$, the minimal dihedral angle θ_{min} of
 650 each tetrahedron;
- 651 • $Q_2 = \theta_{max}$, the maximal dihedral angle θ_{max}
 652 of each tetrahedron;
- 653 • $Q_3 = \frac{3r_{in}}{r_{circ}}$, the radius-ratio of each tetra-
 654 hedral, where r_{in} and r_{circ} are the in-
 655 scribed/circumscribed radius, respectively;
- 656 • $Q_4 = \frac{12\sqrt[3]{9V^2}}{\sum l_{i,j}^2}$, meshing quality of [39], where
 657 V is the volume of the tetrahedron, and $l_{i,j}$
 658 the length of the edge which connects vertices
 659 v_i and v_j .

660 Q_3 and Q_4 are between 0 and 1, where 0 denotes a
 661 silver and 1 denotes a regular tetrahedron.

662 We choose the sphere generated from an iso-
 663 surface as input domain. The Hausdorff distance
 664 (measured by Metro [40]) between the boundary
 665 of generated mesh and the input surface (normal-
 666 ized by dividing by the diagonal of bounding box)
 667 is 0.049%, which is 3 times smaller than 0.17% re-
 668 ported by [32]. The quality of the tetrahedral mesh
 669 is shown in Figure.15 and the comparison of each
 670 measurement is given in Table 2. Our approach
 671 produces better meshing quality, as well as smaller
 672 surface approximation error, attributed to the ex-
 673 act clipped Voronoi diagram computation.

method	$\overline{Q_1}$	$\overline{Q_4}$	$min(Q_1)$	$min(Q_4)$	HDist
[4]	48.11°	0.847	12.05°	0.339	0.054%
[32]	56.32°	0.911	16.31°	0.376	0.170%
ours	56.37°	0.932	24.23°	0.560	0.049%

Table 2: Comparison of meshing qualities. HDist is the Hausdorff distance between the boundary of generated mesh and the input discretized isosurface.

674 We also compare our result with an octree-based
 675 approach [27]. As shown in Figure 16, the CVT
 676 based approach exhibits much better element qual-
 677 ity than a standard approach. Our approach out-
 678 performs previous work in boundary approximation
 679 error (as shown in Figure 17), attributed to the ex-
 680 act clipped Voronoi diagram computation and si-
 681 multaneous surface remeshing [3].

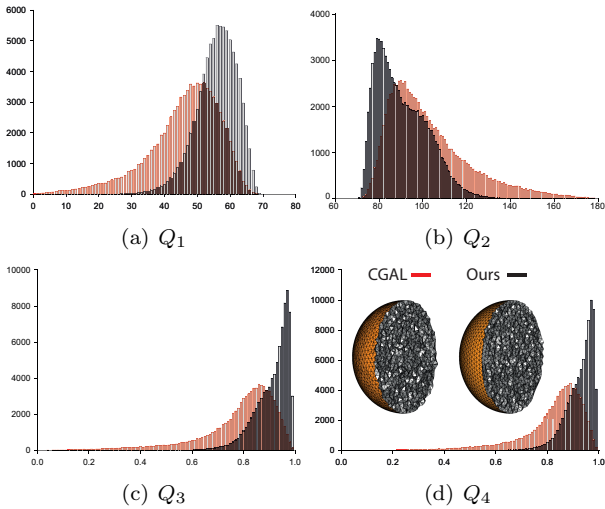


Figure 15: Comparison of the meshing qualities of the sphere with the Delaunay refinement approach implemented in CGAL [4].

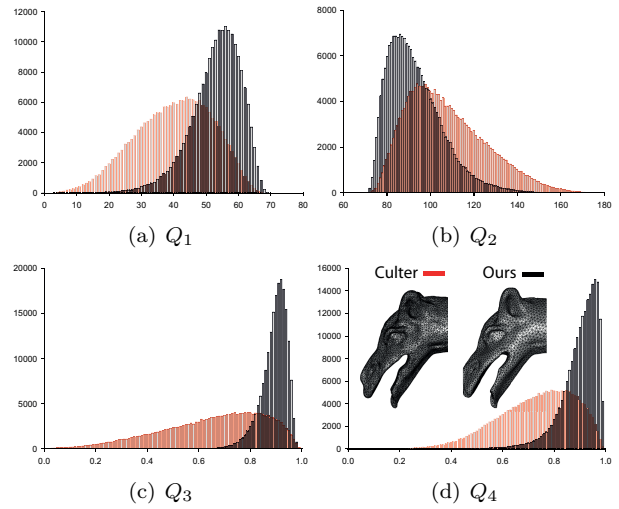


Figure 16: Comparison with the octree based approach [27]. The resulting tetrahedral mesh has 200k tetrahedra.

7. Conclusion

We have presented efficient algorithms for computing the clipped Voronoi diagram for closed 2D and 3D objects, which has been a difficult problem without an efficient solution. As an application, we present a new CVT-based mesh generation algorithm which combines the clipped VD computation and fast CVT optimization.

In the future, we plan to look for more interdisciplinary applications of the clipped Voronoi diagram, such as biology and architecture. Applying our meshing technique to physical simulation applications, and extending the clipped Voronoi diagram to a higher dimension are also interesting directions.

Acknowledgements

We would like to thank anonymous reviewers for their detailed comments and suggestions which greatly improve the manuscript. We also thank one reviewer who pointed out the reference [1]. This work is partially supported by the Research Grant Council of Hong Kong (project no.: 718209 and 718010), the State Key Program of NSFC project (60933008), European Research Council (GOODSHAPE FP7-ERC-StG-205693), and ANR/NSFC (60625202, 60911130368) Program (SHAN Project).

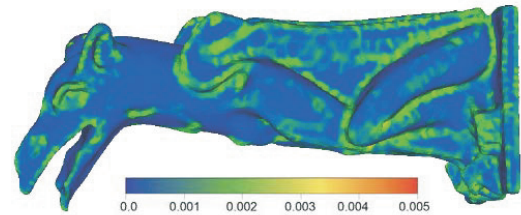


Figure 17: Approximation error of gargoyle model: 50K vertices, 256K tetrahedra, mean/max Hausdorff distance: 0.045%/0.37%. Our approach produces smaller approximation error compared with [30] (mean error: 0.053%) using the same number of vertices.

References

- [1] Timothy M. Y. Chan, Jack Snoeyink, and Chee-Keng Yap. Output-sensitive construction of polytopes in four dimensions and clipped Voronoi diagrams in three. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 282–291, 1995.
- [2] Yang Liu, Wenping Wang, Bruno Lévy, Feng Sun, Dong-Ming Yan, Lin Lu, and Chenglei Yang. On centroidal Voronoi tessellation: Energy smoothness and fast computation. *ACM Trans. on Graphics*, 28(4):Article No. 101, 2009.
- [3] Dong-Ming Yan, Bruno Lévy, Yang Liu, Feng Sun, and Wenping Wang. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. *Computer Graphics Forum (Proceedings of SGP 2009)*, 28(5):1445–1454, 2009.
- [4] CGAL, Computational Geometry Algorithms Library. <http://www.cgal.org>.
- [5] C. Bradford Barber, David P. Dobkin, and Hannu Huh-

- 727 danpaa. The quickhull algorithm for convex hulls. *ACM* 792
728 *Trans. Math. Software*, 22:469–483, 1996. 793
- 729 [6] Franz Aurenhammer. Voronoi diagrams: a survey of a 794
730 fundamental geometric data structure. *ACM Comput-* 795
731 *ing Surveys*, 23(3):345–405, 1991. 796
- 732 [7] Steven Fortune. Voronoi diagrams and Delaunay trian- 797
733 gulations. In *Computing in Euclidean Geometry*, pages 798
734 193–233, 1992. 799
- 735 [8] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, and 800
736 Sung Nok Chiu. *Spatial Tessellations: Concepts and* 801
737 *Applications of Voronoi Diagrams*. Wiley, 2nd edition, 802
738 2000. 803
- 739 [9] R. Kunze, F.-E. Wolter, and T. Rausch. Geodesic 804
740 Voronoi diagrams on parametric surfaces. In *Proceed-* 805
741 *ings of Computer Graphics International 1997*, pages 806
742 230–237, 1997. 807
- 743 [10] Gabriel Peyré and Laurent D. Cohen. Geodesic remesh- 808
744 ing using front propagation. *Int. J. of Computer Vision*, 809
745 69:145–156, 2006. 810
- 746 [11] Herbert Edelsbrunner and Nimish R. Shah. Triangu- 811
747 lating topological spaces. *Int. J. Comput. Geometry* 812
748 *Appl.*, 7(4):365–378, 1997. 813
- 749 [12] Qiang Du, Max. D. Gunzburger, and Lili Ju. Con- 814
750 strained centroidal Voronoi tessellations for surfaces. 815
751 *SIAM J. Sci. Comput.*, 24(5):1488–1506, 2003. 816
- 752 [13] Kelly A. Lyons, Henk Meijer, and David Rappaport. 817
753 Algorithms for cluster busting in anchored graph drawing. 818
754 *J. Graph Algorithms Appl.*, 2(1):1–24, 1998. 819
- 755 [14] Dong-Ming Yan, Kai Wang, Bruno Lévy, and Laurent 820
756 Alonso. Computing 2D periodic centroidal Voronoi tess- 821
757 sellation. In *Proc. of 8th International Symposium on* 822
758 *Voronoi Diagrams in Science and Engineering (ISVD)*, 823
759 pages 177 – 184, 2011. 824
- 760 [15] Benoit Hudson, Gary L. Miller, Steve Y. Oudot, and 825
761 Donald R. Sheehy. Topological inference via meshing. 826
762 In *Proceedings of the 2010 annual symposium on Com-* 827
763 *putational geometry (SOCG)*, pages 277–286, 2010. 828
- 764 [16] Kenneth E. Hoff III, John Keyser, Ming C. Lin, and 829
765 Dinesh Manocha. Fast computation of generalized 830
766 Voronoi diagrams using graphics hardware. In *Proceed-* 831
767 *ings of ACM SIGGRAPH 1999*, pages 277–286, 1999. 832
- 768 [17] Avneesh Sud, Naga K. Govindaraju, Russell Gayle, 833
769 Ilknur Kabul, and Dinesh Manocha. Fast proximity 834
770 computation among deformable models using discrete 835
771 Voronoi diagrams. *ACM Trans. on Graphics (Proc.* 836
772 *SIGGRAPH)*, 25(3):1144–1153, 2006. 837
- 773 [18] S. Owen. A survey of unstructured mesh generation 838
774 technology. In *Proceedings of 7th International Meshing* 839
775 *Roundtable*, pages 26–28, 1998. 840
- 776 [19] Pascal Jean FREY and Paul-Louis GEORGE. *Mesh* 841
777 *Generation: Application to Finite Elements*. Hermès 842
778 Science, 2000. 843
- 779 [20] Nina Amenta, Marsahll Bern, and Manolis Kamvys- 844
780 selis. A new Voronoi-based surface reconstruction algo- 845
781 rithm. In *Proceedings of ACM SIGGRAPH 1998*, 846
782 pages 415–421, 1998. 847
- 783 [21] Pierre Alliez, David Cohen-Steiner, Yiyang Tong, and 848
784 Mathieu Desbrun. Voronoi-based variational recon- 849
785 struction of unoriented point sets. In *Proceedings* 850
786 *of Symposium on Geometry Processing (SGP 2007)*, 851
787 pages 39–48, 2007. 852
- 788 [22] F. Leymarie and B. Kimia. The medial scaffold of 3D 853
789 unorganized point clouds. *IEEE Trans. Pattern Anal.* 854
790 *Mach. Intell.*, 29(2):313–330, 2007. 855
- 791 [23] S. Yamakawa and K. Shimada. Quad-layer: Layered 856
quadilateral meshing of narrow two-dimensional do-
main by bubble packing and chordal axis transforma-
tion. *Journal of Mechanical Design*, 124:564–573, 2002.
- [24] W. R. Quadros and K. Shimada. Hex-layer: Layered all-
hex mesh generation on thin section solids via chordal
surface transformation. In *Proc. of 11th International*
Meshing Roundtable, pages 169–180, 2002.
- [25] Peter Sampl. Semi-structured mesh generation based
on medial axis. In *Proceedings of the 9th International*
Meshing Roundtable, pages 21–32, 2000.
- [26] J. R. Shewchuk. What is a good linear element? inter-
polation, conditioning, and quality measures. In *Pro-*
ceedings of the 11th International Meshing Roundtable,
pages 115–126, 2002.
- [27] Barbara Cutler, Julie Dorsey, , and Leonard McMillan.
Simplification and improvement of tetrahedral models
for simulation. In *Proceedings of the Eurographics Sym-*
posium on Geometry Processing, pages 93–102, 2004.
- [28] Yi-Jun Yang, Jun-Hai Yong, and Jia-Guang Sun. An
algorithm for tetrahedral mesh generation based on con-
forming constrained Delaunay tetrahedralization. *Com-*
puters & Graphics, 29(4):606–615, 2005.
- [29] L. Chen and J. Xu. Optimal Delaunay triangulations.
Journal of Computational Mathematics, 22(2):299–308,
2004.
- [30] Pierre Alliez, David Cohen-Steiner, Mariette Yvinec,
and Mathieu Desbrun. Variational tetrahedral meshing.
ACM Transactions on Graphics (Proceedings of ACM
SIGGRAPH 2005), 24(3):617–625, 2005.
- [31] Jane Tournois, Camille Wormser, Pierre Alliez, and
Mathieu Desbrun. Interleaving Delaunay refinement
and optimization for practical isotropic tetrahedron
mesh generation. *ACM Trans. on Graphics (Proc. SIG-*
GRAPH), 28(3):Article No. 75, 2009.
- [32] J. Dardenne, S. Valette, N. Siauve, N. Burais, and
R. Prost. Variational tetrahedral mesh generation from
discrete volume data. *The Visual Computer (Proceed-*
ings of CGI 2009), 25(5):401–410, 2009.
- [33] Qiang Du, Vance Faber, and Max Gunzburger. Cen-
troidal Voronoi tessellations: applications and algo-
rithms. *SIAM Review*, 41(4):637–676, 1999.
- [34] Masao Iri, Kazuo Murota, and Takao Ohya. A fast
Voronoi diagram algorithm with applications to geo-
graphical optimization problems. In *Proceedings of the*
11th IFIP Conference on System Modelling and Opti-
mization, pages 273–288, 1984.
- [35] Ivan E. Sutherland and Gary W. Hodgman. Reen-
trant polygon clipping. *Communications of the ACM*,
17(1):32–42, 1974.
- [36] Jane Tournois, Rahul Srinivasan, and Pierre Alliez. Per-
turbating slivers in 3D Delaunay meshes. In *Proceedings*
of the 18th International Meshing Roundtable, pages
157–173, 2009.
- [37] Hang Si. TetGen: A quality tetrahedral mesh gener-
ator and three-dimensional Delaunay triangulator.
<http://tetgen.berlios.de>.
- [38] Andreas Meyer and Sylvain Pion. FPG: A code gen-
erator for fast and certified geometric predicates. *Real*
Numbers and Computers (RNC), pages 47–60, 2008.
- [39] Anwei Liu and Barry Joe. On the shape of tetra-
hedra from bisection. *mathematics of computation*,
63(207):141–154, 1994.
- [40] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Mea-
suring error on simplified surfaces. *Computer Graphics*
Forum, 17(2):167–174, 1998.