Computers & Graphics 114 (2023) 247–256

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on SMI 2023

Meshless power diagrams

Yanyang Xiao[a], Juan Cao[b], Shaoping Xu[a], Zhonggui Chen[c,*]

[a] *School of Mathematics and Computer Sciences, Nanchang University, Nanchang, 330031, China*
[b] *School of Mathematical Sciences, Xiamen University, Xiamen, 361005, China*
[c] *School of Informatics, Xiamen University, Xiamen, 361005, China*

ARTICLE INFO

Article history:
Received 17 May 2023
Received in revised form 5 June 2023
Accepted 14 June 2023
Available online 17 June 2023

Keywords:
Weighted Voronoi diagrams
Power diagrams
k-nearest neighbors

ABSTRACT

The computation of power diagrams (or weighted Voronoi diagrams) is a fundamental task in computational geometry and computer graphics. To accomplish the computation, we provide a different way from the existing ones for lifting the weighted seeds to a set of points in the space of one dimension higher, then the power cells can be directly obtained by computing the intersections of the Voronoi cells of these lifting points and the original space. This property enables us to apply the method based on the *k*-nearest neighbors query to the generation of power diagrams. Each power cell is obtained by sequentially clipping the input domain using the bisectors between its seed and the *k*-nearest neighbors. Experimental results demonstrate that our method outperforms the state-of-the-art one based on regular triangulation in terms of efficiency for general cases in the 2D and 3D spaces.

1. Introduction

A Voronoi diagram divides the space into regions composed of points closest to one of a set of points (referred to as seeds). Voronoi diagram is one of the most important geometric structures for region partition in computational geometry and computer graphics [1]. It has been extensively generalized using different metrics other than the Euclidean distance or replacing points with other geometric objects [2–6]. The power diagram [2] (or weighted Voronoi diagram) is one of the generalizations, with a wide range of applications, including sampling [7,8], adaptive remeshing [9,10], mesh generation [11,12], superpixels segmentation [13,14], and simulation [15,16]. Hence, the computation of power diagrams is a fundamental task, and improving its efficiency has considerable importance for downstream applications.

The additional degree of freedom makes the power diagram more complicated than the Voronoi diagram given that each seed is attached a real number, called weight, posing a challenge to its efficient computation. Only few methods are available for generating power diagrams of general cases. Aurenhammer 1987 [2] introduced an algorithm that uses a convex hull in a higher dimensional space to construct power diagrams. A more common approach is based on the duality between the power diagram and weighted Delaunay triangulation (or regular triangulation, RT) [17], and there have been robust and high-quality implementations, such as the computational geometry algorithms library (CGAL) [18]. Compared with an abundant number of studies on the computation of Voronoi diagrams in the literature, e.g., [19–29], the research on the computation of power diagrams lags far behind.

Several works have attempted to extend Voronoi generation methods to power diagram computation, such as the extension of the *k*-nearest neighbors (kNN) based method [16,30], also referred to meshless method. Based on the observation that each Voronoi cell is the intersection of half-spaces between the corresponding seed and all the other seeds, the kNN based method [26,28,31] sorts the bisectors associated with that seed in ascending order according to its distance to the other seeds, and then uses these bisectors to clip the given domain to produce its Voronoi cell. The clipping procedure can be terminated in advance once the distance from the seed to a bisector is greater than the maximum distance from the seed to the cell. Zhai et al. 2020 [16] and Basselin et al. 2021 [30] applied a similar strategy in the computation of power diagrams. However, their extensions have a strong precondition that the weights of adjacent seeds have close values, which evidently cannot be used for general cases.

Directly applying the kNN based method to the computation of power diagrams is inappropriate because the distance metric of power diagrams is not additive in the weight dimension, resulting in an inefficient sorting of the distances from a weighted seed to the bisectors. It has been proved that any power diagram actually corresponds to a Voronoi diagram in a higher dimensional space [2,32]. The weighted seeds can be lifted to a set of objects in a higher dimensional space, which are then used for

* Corresponding author.
E-mail addresses: xiaoyanyang@ncu.edu.cn (Y. Xiao), juancao@xmu.edu.cn (J. Cao), xushaoping@ncu.edu.cn (S. Xu), chenzhonggui@xmu.edu.cn (Z. Chen).

https://doi.org/10.1016/j.cag.2023.06.014
0097-8493/© 2023 Elsevier Ltd. All rights reserved.

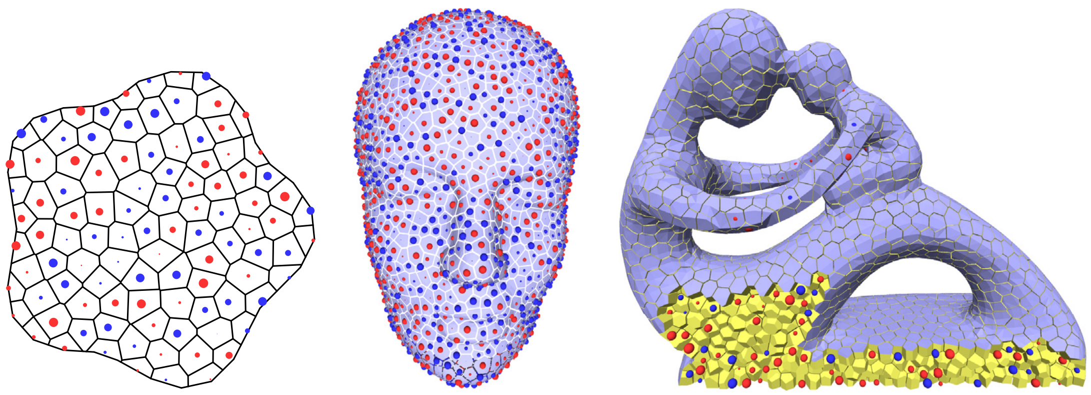


Fig. 1. Power diagrams on various domains (2D polygon, surface mesh and 3D volume, respectively) generated by our method. The seeds with positive and negative weights are rendered as red and blue disks/balls, respectively, and the magnitude of weight is proportional to the size of corresponding disk/ball.

the computation of power cells. In the Observation 7 of [33], Lévy presented a lifting way that the seed weight can be transformed as a new coordinate of the seed location, making the Voronoi generation methods directly applied to the computation of power diagrams.

In this paper, we further provide a variant of the lifting way in [33], the weighted seeds are mapped to a set of higher dimensional points, each of them has a twin point. Based on this, the power cells can be quickly obtained by computing the intersections of the Voronoi cells of these lifting points and the original space. And we develop an efficient method based on the kNN query to accomplish the computation. Our specific contributions can be summarized as follows:

- We provide a variant for lifting the weighted seeds to a set of points in a higher dimensional space, and propose an efficient method for power diagram generation based on the kNN query, which outperforms the state-of-the-art method based on regular triangulation in terms of efficiency for general cases in the 2D and 3D spaces.

The remainder of this paper is organized as follows: Section 2 reviews the computation methods of Voronoi and power diagrams. Section 3 introduces the lifting theory for weighted seeds. Section 4 proposes the kNN based method for generating power diagrams. Section 5 provides an acceleration technique for the proposed method. Section 6 presents the experimental results and applications. Section 7 concludes the paper.

2. Related work

2.1. Computation of Voronoi diagrams

Many robust and efficient methods for generating Voronoi diagrams have been proposed in the recent decades [19–29,34–36]. The divide-and-conquer method [19] recursively splits the given seeds into two halves and merges their Voronoi diagrams. Brown 1979 [21] transformed the original seeds to a higher dimensional space and computed their convex hull to generate Voronoi diagrams. The sweep-line method [22] uses line scanning to find the Voronoi corners and their connection. These methods only apply to planar Voronoi diagram generation because their key steps are so complex that they are challenging to implement in high-dimensional space. Similarly, the GPU-assisted Voronoi calculation methods are also limited to low-dimensional space [34–36].

The more popular methods for computing the Voronoi diagram is based on the duality of Voronoi and Delaunay tessellations. The main focus of these methods is the fast construction of Delaunay triangulation, which can be accomplished by various algorithms, including the Lawson algorithm [37], the Bowyer–Watson algorithm [38,39], the divide-and-conquer algorithm [40] and the incremental algorithm [41]. Parallel construction methods have also been developed to achieve higher efficiency (e.g., [42,43]).

In recent years, the meshless Voronoi generation methods have attracted remarkable attention [16,26,28,30,31]. This method has a great advantage in performance over the Delaunay triangulation-based method because it only relies on the seed locations. It performs effectively, especially when the seeds are uniform [28]. Once the neighboring relationship between seeds is established, the computation of each Voronoi cell is independent, hence is suitable for parallel implementation on GPUs [28,29].

2.2. Computation of power diagrams

The computation of power diagrams attracts much less attention than the Voronoi generation. Aurenhammer 1987 [2] presented an algorithm for the computation of d-dimensional power diagram using a (d+1)-dimensional convex hull. Nocaj and Brandes 2012 [44] reduced this algorithm to three steps for only the 2D cases. Zheng et al. 2019 [45] extended the jump flooding algorithm to compute planar power diagrams with the assistance of GPUs. In practice, the method of constructing a regular triangulation [17] and then calculating its duality to obtain the power diagram is more common. Zhai et al. 2020 [16] and Basselin et al. 2021 [30] have adopted the kNN based method to generate power diagrams. However, their methods are only applicable when the weights of the adjacent seeds are close to each other. In this paper, we develop an efficient method for the computation of general power diagrams by using a KD-tree.

3. From power diagram to restricted Voronoi diagram

Given n seeds $\{\mathbf{v}_i\}_{i=1}^n$ in the d-dimensional space $\mathcal{R}^d$, each of which is assigned with a weight $w_i, i = 1, \ldots, n$, the power diagram divides the space $\mathcal{R}^d$ into a set of cells $\{\Omega_i\}_{i=1}^n$ as follows

$$\Omega_i = \{\mathbf{x} \in \mathcal{R}^d | D(\mathbf{x}, \mathbf{v}_i, w_i) \le D(\mathbf{x}, \mathbf{v}_j, w_j), \forall j \neq i\}, \quad (1)$$

where $D(\mathbf{x}, \mathbf{v}, w) = \|\mathbf{x} - \mathbf{v}\|^2 - w$ is the distance function, see Fig. 2(a).

It has been shown in [2] that a d-dimensional power diagram can be constructed by lifting it to (d+1)-dimensional space. Specifically, each weighted seed $(\mathbf{v}_i, w_i)$ is mapped to a hyperplane h_i in the space $\mathcal{R}^{d+1}$, where

$$h_i : z = 2\mathbf{v}_i \cdot \mathbf{x} + w_i - \|\mathbf{v}_i\|^2, \mathbf{x} \in \mathcal{R}^d \quad (2)$$

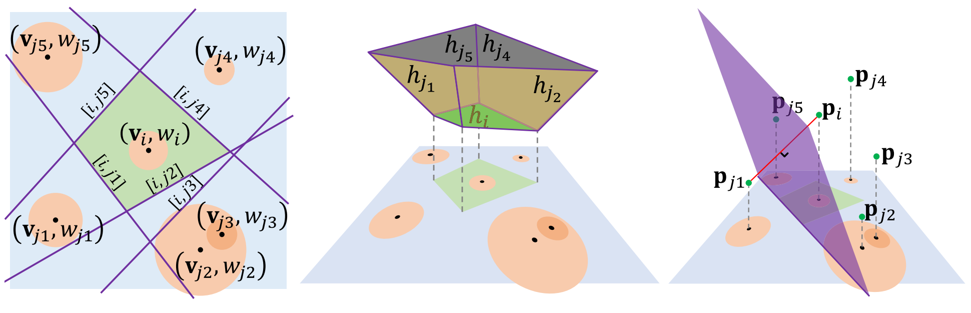


Fig. 2. Illustration of the equivalence between a power diagram and a higher dimensional restricted Voronoi diagram by a 2D example. The weighted seeds are shown as black dots and brown disks. (a) The power cell of seed $\mathbf{v}_i$ (light green) is obtained by clipping the domain (light blue) using the nearest bisectors (purple line). (b) In [2], the weighted seeds are mapped to a set of planes $\{h_j\}$, and the intersection of the halfspaces above $\{h_j\}$ is a polyhedron, whose facets projected back to the original space are the power cells. (c) We propose a different lifting way. The weighted seeds are lifted as 3D points $\{\mathbf{p_j}\}$ (green dots), and the power cell of seed $\mathbf{v}_i$ can be directly obtained by computing the intersection of the 3D Voronoi cell of $\mathbf{p}_i$ and the 2D domain.

and z is the $(d+1)$-th coordinate. The intersection Z^+ of the halfspaces above $h_1, \dots, h_n$ is a $(d+1)$-polyhedron, whose facets projected back to $\mathcal{R}^d$ are the power cells of the original seeds, see Fig. 2(b). Please refer to [2] for more details.

The above lifting from $\mathcal{R}^d$ to $\mathcal{R}^{d+1}$ for the weighted seeds does not allow us to directly apply Voronoi generation methods to the computation of power diagrams. We introduce a variant of the lifting way presented in [33] in the rest of this section.

The problem that $D(\mathbf{x}, \mathbf{v}, w)$ is not additive in the weight dimension can be solved by introducing a parameter $\eta \geq w_{max}$, where

$$w_{max} = \max\{w_1, w_2, \dots, w_n\} \tag{3}$$

is the maximum weight. The following inequality relation remains unchanged by introducing the parameter η into the distance function $D_\eta(\mathbf{x}, \mathbf{v}, w) = \|\mathbf{x} - \mathbf{v}\|^2 + \eta - w$:

$$D_\eta(\mathbf{x}, \mathbf{v}_i, w_i) \leq D_\eta(\mathbf{x}, \mathbf{v}_j, w_j), \forall j \neq i. \tag{4}$$

In other words, the partition controlled by the above modified distance function remains the same as the original power diagram. As $\eta - w_i \geq 0, \forall i$, the modified distance function can be further treated as follows:

$$D_\eta(\mathbf{x}, \mathbf{v}, w) = \|\mathbf{x} - \mathbf{v}\|^2 + (0 - \pm\sqrt{\eta - w})^2 = \|\mathbf{y} - \mathbf{p}\|^2, \tag{5}$$

where $\mathbf{y} = (\mathbf{x}, 0)$ and $\mathbf{p} = (\mathbf{v}, \pm\sqrt{\eta - w})$. $\mathbf{p} \in \mathcal{R}^{d+1}$ is referred to as the lifting point of $\mathbf{v} \in \mathcal{R}^d$. Eq. (5) is actually the squared Euclidean distance of two points in the $(d+1)$-dimensional space $\mathcal{R}^{d+1}$. Therefore, the power cell is equally defined by the following:

$$\Omega_i = \{\mathbf{x} \in \mathcal{R}^d \mid \|\mathbf{y} - \mathbf{p}_i\|^2 \leq \|\mathbf{y} - \mathbf{p}_j\|^2, \forall j \neq i\}, \tag{6}$$

which is the restriction of Voronoi cell of $\mathbf{p}_i$ in the $(d+1)$ dimension space to $\mathcal{R}^d$, as shown in Fig. 2(c). A hidden seed dominates a null region if the intersection between the Voronoi cell of the corresponding lifting point and $\mathcal{R}^d$ is empty.

The two options of lifting point for each seed produce the same result because $\mathcal{R}^d$ intersects all the possible bisectors of $\mathbf{p}_i$ and $\mathbf{p}_j$, $\forall i \neq j$, at the same location. For simplicity, we set all lifting points as $\{\mathbf{p}_i = (\mathbf{v}_i, \sqrt{\eta - w_i})\}_{i=1}^n$. The parameter η implies how far the lifting points are from the space $\mathcal{R}^d$. The lower bound of η cannot be determined until the weights are provided.

Therefore, we can directly obtain the power diagram for any weighted seeds by computing a higher dimensional restricted Voronoi diagram. The Voronoi generation methods are applicable to the computation of power diagrams given this property.

4. Meshless power diagrams

In practical applications, it is usually desired to compute the power cells inside a given bounded domain, i.e., the clipped or restricted power diagrams. Equipped with the above theory, we develop a method based on the kNN query to compute the power diagrams on different given domains in this section. The method is referred to meshless power diagrams, since we do not construct any triangulations. We initially describe the method in detail, and then discuss the choice of parameters.

4.1. Overview

Consider a domain $\mathcal{M} = \{\tau\}$ (2D polygons, surface mesh or 3D polytopes) and a set of weighted seeds $\{(\mathbf{v}_i, w_i)\}_{i=1}^n$. The intersection of the power cells of $\{(\mathbf{v}_i, w_i)\}_{i=1}^n$ and $\mathcal{M}$ can be computed by the following steps, refer to the pseudo-code in Algorithm 1.

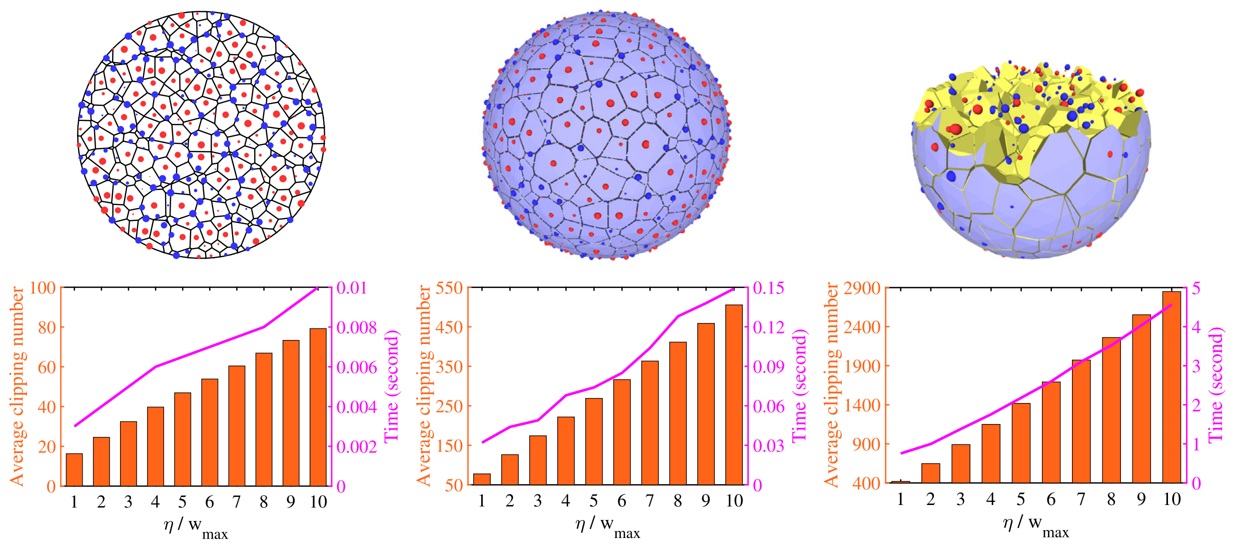
Lifting. The given weighted seeds $\{(\mathbf{v}_i, w_i)\}_{i=1}^n$ are lifted to the higher dimensional space as $\mathbf{p}_i = (\mathbf{v}_i, \sqrt{\eta - w_i}), i = 1, \dots, n$. The lifting points will be treated as the Voronoi seeds to generate bisectors for the subsequent clipping step. Note that the value of η highly affects the computation efficiency. We defer the discussion in Section 4.3.

KD-tree construction. The neighborhood search between lifting points is required in the next step. The KD-tree is a mature and widely used data structure to organize the given dataset and fast query the nearest neighbors of the target point. Thus, we construct a KD-tree for the lifting points once they have been generated.

Clipping. The bisector of points $\mathbf{p}_i$ and $\mathbf{p}_j$ is the plane satisfying $(\mathbf{p}_j - \mathbf{p}_i)^T \cdot (\mathbf{y} - (\mathbf{p}_j + \mathbf{p}_i)/2) = 0, \forall i \neq j$. We compute the intersection of any region and the half-space bounded by the bisector using Sutherland Hodgman clipping algorithm [46]. To obtain the restricted Voronoi cell Ω_i of $\mathbf{p}_i$, we initially set $\Omega_i = \mathcal{M}$, and sort the bisectors between $\mathbf{p}_i$ and all the other lifting points in ascending order according to their distances from $\mathbf{p}_i$, and then clip Ω_i gradually by these bisectors in sequence. In addition, sorting the bisectors associated with $\mathbf{p}_i$ is equivalent to sorting the lifting points except for $\mathbf{p}_i$, which can be achieved efficiently using the KD-tree. See more details in Section 4.2.

4.2. Clipping

A bisector has a contribution to the final cell if the result of its clipping is retained. There are only a few nearest bisectors that contribute to the cell result. Sorting all the lifting points at once is unnecessary because the clipping procedure can be terminated in



(a) 2D case, 300 weighted seeds (b) Surface case, 800 weighted seeds (c) 3D case, 1500 weighted seeds

Fig. 3. The influence of η on the computation efficiency. For all cases, the larger η results in more average clipping number of Voronoi cells and the running time. Thus, we suggest $\eta = w_{\max}$ for the kNN based computation of power diagrams.

Algorithm 1 Meshless Power Diagram

```
Input: domain $\mathcal{M} = \{\tau\}$, weighted seeds $\{(\mathbf{v}_i, w_i)\}_{i=1}^{n}$;
Output: power diagram $\{\Omega_i\}_{i=1}^{n}$;
 1: find the maximum weight $w_{\max}$ and set $\eta$;
 2: generate lifting points $\{\mathbf{p}_i = (\mathbf{v}_i, \sqrt{\eta - w_i})\}_{i=1}^{n}$;
 3: build a KD-tree of $\{\mathbf{p}_i\}_{i=1}^{n}$;
 4: set stack $\mathcal{S} \leftarrow \emptyset$;
 5: for each region $\tau$ of $\mathcal{M}$ do
 6:   push the nearest lifting point of the centroid of $\tau$ into $\mathcal{S}$;
 7:   while $\mathcal{S}$ not empty do
 8:     $\mathbf{p}_i \leftarrow \mathcal{S}$.top(), $\mathcal{S}$.pop();
 9:     reset $\tau$;
10:     calculate the farthest distance $D_{\max}$ from $\tau$ to $\mathbf{p}_i$;
11:     $k \leftarrow$ initial number of nearest neighbors;
12:     find indices $\mathcal{N}$ of $k$ nearest lifting points of $\mathbf{p}_i$;
13:     $j \leftarrow 0$;
14:     while $D_{\max} > \|\mathbf{p}_{\mathcal{N}[j]} - \mathbf{p}_i\|/2$ do
15:       clip $\tau$ using the bisector of $\mathbf{p}_i$ and $\mathbf{p}_{\mathcal{N}[j]}$;
16:       update $D_{\max}$, $j \leftarrow j + 1$;
17:       if $j \equiv k$ then
18:         $k \leftarrow k \times 2$ and enlarge $\mathcal{N}$;
19:       end if
20:     end while
21:     append final $\tau$ to $\Omega_i$;
22:     push unvisited neighbors' indices of $\mathbf{p}_i$ contributed to
        final $\tau$ to $\mathcal{S}$;
23:   end while
24: end for
```

advance if it satisfies the condition that the farthest distance from $\mathbf{p}_i$ to Ω_i is smaller than half of the distance from $\mathbf{p}_i$ to its neighbor. Therefore, it is reasonable to set up a dynamic container, to which we place a small number of nearest neighbors of $\mathbf{p}_i$ initially, and then enlarge it until the clipping procedure terminates.

The process to compute Ω_i is as follows. We denote L as the initial number of neighbors, and get L neighbors for $\mathbf{p}_i$ from the KD-tree. Then we visit the neighbors one by one and use the bisector between $\mathbf{p}_i$ and its neighbor to clip the domain. Once the end of the neighbors list is reached, we enlarge the number of neighbors twice and update the neighbors list from the KD-tree, as shown in the step 18 in Algorithm 1. When the termination condition is met, Ω_i is obtained. An appropriate value of L can reduce the times of neighborhood searches, thereby improving the efficiency, see more details in Section 4.4.

In practice, the input domain $\mathcal{M}$ is usually assembled by a set of smaller region elements (polygons in 2D and surface mesh, polytopes in 3D). Each Voronoi cell comprises parts of only a few nearest regions of the corresponding lifting point. Hence, testing whether all the regions intersect with each Voronoi cell is unnecessary. We visit each region and divide it using the bisectors between its nearest lifting points. Specifically, we identify the nearest lifting point $\mathbf{p}_i$ of the centroid of the given region τ, and use the k-nearest bisectors of $\mathbf{p}_i$ to clip it. Because τ is bounded and we enlarge the range of seed neighbors if the termination condition of clippings is not met (see step 18 in Algorithm 1), the clipped τ must be an accurate result, which is finally appended to Ω_i. The neighbors of $\mathbf{p}_i$ that contributed to the final τ are certainly the seeds whose Voronoi cell intersects with the original τ. We should compute their intersections further. Lastly, each power cell Ω_i consists of a set of small clipped regions, which is an accurate result, and no regions are missed.

4.3. Setting of η

As mentioned above, η controls the distance from the lifting points to the domain $\mathcal{M}$. Different values of η result in the same region partition. However, a larger η moves the lifting points further away from the domain, leading to a larger maximum distance from the lifting point to its restricted Voronoi cell. Therefore, the termination condition of the clipping procedure becomes more difficult to satisfy for each Voronoi cell, i.e., more clipping tests are required.

A comparison of different η values on the computation efficiency is shown in Fig. 3. The input domains are a unit disk, a sphere-shaped triangular mesh, and a tetrahedralized ball. The weighted seeds are randomly generated with weights in the range of $[-0.01, 0.01]$. With the growth of η $(1 \sim 10 \times w_{\max})$, the average clipping number of Voronoi cells and the running time increase dramatically, as shown in the orange bars and pink curves in Fig. 3, respectively. Therefore, we suggest a minimal value of $\eta = w_{\max}$ for the computation of power diagrams.

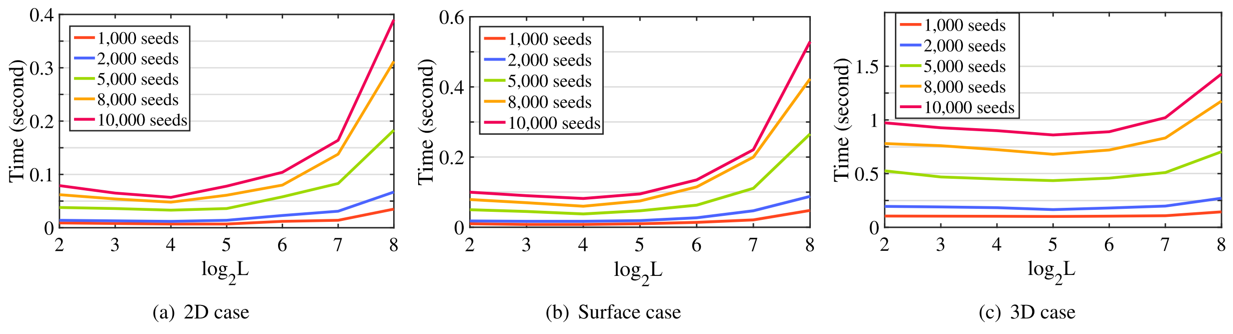


Fig. 4. The timing curves of generating power diagrams against to the different values of L. Five sets of random weighted seeds on each domain (2D unit square, the surface and the volume of a unit cube) are tested. The trend of each curve is similar, i.e., a gradual rise after a slow decline. We suggest $L = 14 \sim 18$ for the 2D and the restricted cases, while $L = 30 \sim 35$ for the 3D case.

4.4. Setting of L

L refers to the initial number of the nearest neighbors of each lifting point, which highly influences the computation efficiency. Given that a target region only intersects with some nearest bisectors, finding excessive nearest neighbors of the target lifting point is unnecessary. On the contrary, if L is too small, then the routine of querying k-nearest neighbors using the KD-tree has to be invoked more frequently, thereby consuming more time.

To find an appropriate value of L, we randomly generate five sets of weighted seeds with different numbers on three input domains, i.e., a 2D unit square, the surface and the volume of a unit cube. Their time costs are recorded for generating power diagrams at different values of L (ranging from 4 to 256). The timing curves are plotted in Fig. 4, where we can observe that the trend of all curves is similarly gradually rising after a slow decline. According to the lowest point of each curve, we suggest $L = 14 \sim 18$ for the 2D and the surface cases, whereas $L = 30 \sim 35$ for the 3D case.

5. Acceleration

The proposed algorithm computes the restricted Voronoi cells of lifting points. We can further speed up the computation of the cell of $\mathbf{p}_i$ by replacing $\mathbf{p}_i$ with its twin point $\mathbf{q}_i = \{\mathbf{v}_i, -\sqrt{\eta - w_i}\}$, $\forall i$ while fixing other lifting points.

As illustrated in Fig. 5, suppose that $\|\mathbf{p}_j - \mathbf{p}_i\|^2 < \|\mathbf{p}_k - \mathbf{p}_i\|^2$ and $\sqrt{\eta - w_j} > \sqrt{\eta - w_i} > \sqrt{\eta - w_k}$ ($\sqrt{\eta - w}$ is the height of the corresponding lifting point to the 2D plane (light blue)). If we compute the cell of $\mathbf{p}_i$, the bisector B_{ij} between $\mathbf{p}_j$ and $\mathbf{p}_i$ is used before the one B_{ik} between $\mathbf{p}_k$ and $\mathbf{p}_i$. However, in this situation, the distance from $\mathbf{p}_i$ to the intersection of B_{ij} and $\mathcal{R}^d$ is possibly larger than that from $\mathbf{p}_i$ to the intersection of B_{ik} and $\mathcal{R}^d$, resulting in the clipping launched by B_{ij} being meaningless. More importantly, such a situation is detrimental to reaching the termination condition of cell clippings as soon as possible.

Section 3 indicates that the cell result of $\mathbf{p}_i$ is equivalent to that of $\mathbf{q}_i$, $\forall i$. For computing the cell of $\mathbf{q}_i$, $i = 1, \ldots, n$ using the proposed algorithm, we keep the same set of $\{\mathbf{p}_i\}_{i=1}^n$ in the KD-tree, but perform kNN query for $\mathbf{q}_i$ instead of $\mathbf{p}_i$, then we simply skip $\mathbf{p}_i$ when it appears in the list of neighbors. In the situation in Fig. 5, $\|\mathbf{p}_k - \mathbf{q}_i\|^2$ is possibly smaller than $\|\mathbf{p}_j - \mathbf{q}_i\|^2$. Hence, the bisector between $\mathbf{p}_k$ and $\mathbf{q}_i$ clips the cell of $\mathbf{q}_i$ earlier, thereby producing smaller maximum distance from $\mathbf{q}_i$ to the cell, and allowing us to end the cell computation with fewer clipping tests.

This trick can effectively improve the efficiency of the meshless computation of power diagrams. A comparison result is shown in Fig. 6. The inputs are the same as the configuration set in Fig. 3 except for the number of seeds. We test 10 sets of random seeds for the 2D, surface, and 3D cases, respectively, and

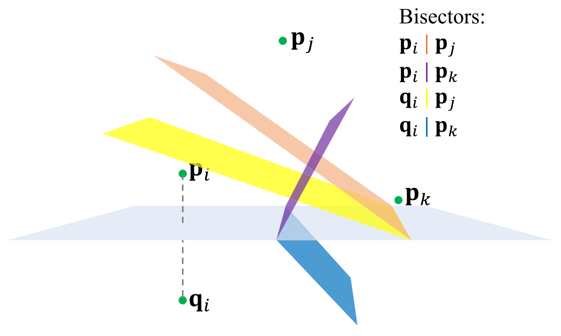

Bisectors:
$\mathbf{p}_i$ | $\mathbf{p}_j$
$\mathbf{p}_i$ | $\mathbf{p}_k$
$\mathbf{q}_i$ | $\mathbf{p}_j$
$\mathbf{q}_i$ | $\mathbf{p}_k$
$\mathbf{p}_j$
$\mathbf{p}_i$
$\mathbf{p}_k$
$\mathbf{q}_i$

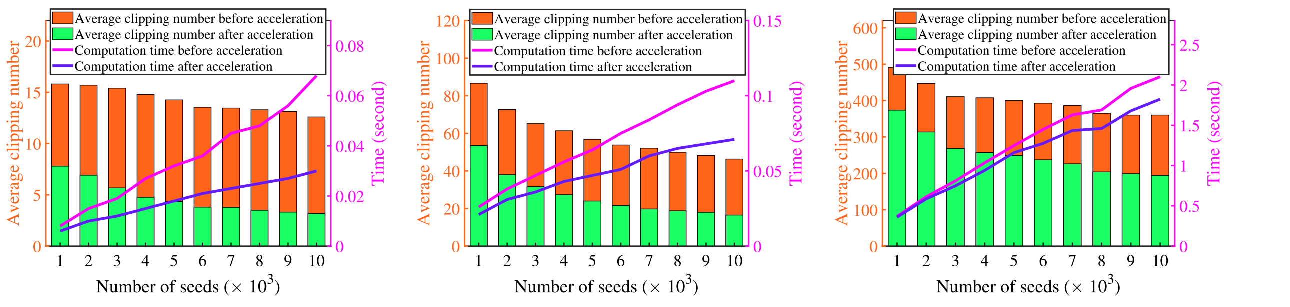
Fig. 5. The termination condition of computing the restricted Voronoi cell of $\mathbf{q}_i$ could be met earlier than that of computing restricted Voronoi cell of $\mathbf{p}_i$, thereby speeding up the computation of power diagrams.

then plot the average clipping number and computation time of the proposed algorithm before and after using the acceleration. It can be seen clearly that both the average clipping number and computation time decrease remarkably after the acceleration technique is embedded. In the remainder of this paper, we refer to the accelerated version as our method.

6. Results and applications

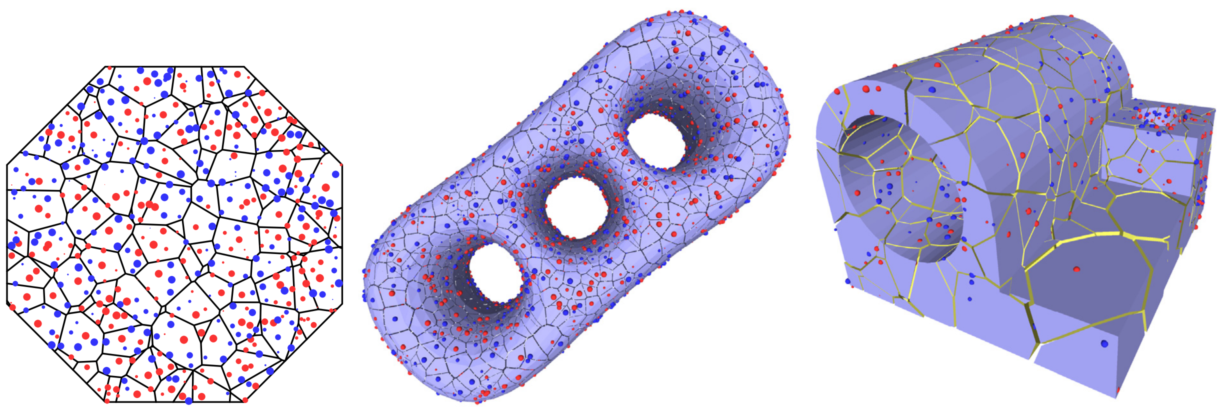
Our method can be applied to the computation of general power diagrams without restrictions on the locations and weights of the given seeds. We focus on its performance in the 2D and 3D spaces in this paper. In this section, we first show the power diagram results generated by our method. Then, we compare our method with the RT-based method in efficiency. Finally, we show the applications of our method in blue noise sampling, adaptive remeshing, and visualization. The proposed algorithm is realized using C++,[1] the KD-tree construction and kNN query are implemented by the NanoFLANN library [47]. The RT-based method is implemented by the CGAL library [18]. We feed a set of tetrahedra to the algorithm when computing the 3D power diagrams. For clipping a 3D cell with a bisector, we use it to clip all the boundary facets of the cell and connect the intersection points to form a new facet appending to the cell. All results are obtained using a laptop with a 1.6 GHz Intel Core I5 processor (4 cores) and 16 GB RAM.

[1] The implementation of our method is open-sourced on https://github.com/yanyangxiao/voronoi.git



(a) 2D case (b) Surface case (c) 3D case

Fig. 6. Comparisons of average clipping number and computation time before and after acceleration. The inputs are consistent with the configuration in Fig. 3 except for the number of seeds.



(a) 2D power diagram (b) Restricted power diagram (c) 3D power diagram

Fig. 7. Power diagram results of randomly generated seeds. Hidden seeds with empty cells are still shown.

6.1. Results and comparisons

Figs. 1 and 7 show our power diagram results on different domains (2D polygon, surface mesh, and 3D volume), respectively. The testing seeds in Fig. 1 are generated by uniform sampling from the domain and attaching random weights ranging $[-0.01, 0.01]$, whereas the seeds in Fig. 7 are generated randomly, and the weights are in the range $[-0.05, 0.05]$.

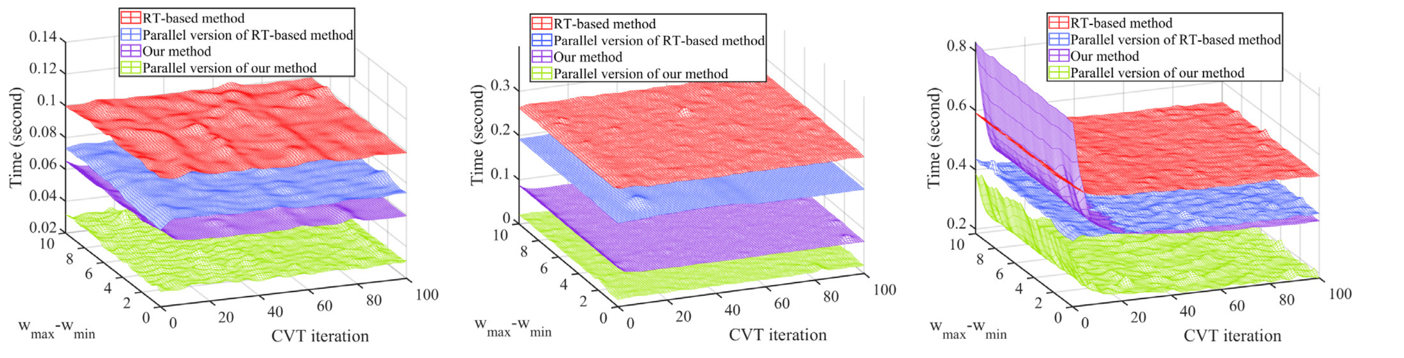
We now discuss the efficiency of our method. It has shown that the distribution of Voronoi seeds is an essential factor affecting the efficiency of the kNN based method, and the computation of the case with uniform distribution is faster than that of the one with random distribution [28], where the Voronoi seeds with uniform distribution means the seeds are distributed in the manner of blue noise sampling, while the seeds with random distribution means the seeds are distributed in white noise. The reason is that the termination condition of clippings can be satisfied earlier in the uniform case than that in the random case. Therefore, the efficiency of our method is sensitive to the distribution of the lifting points, depending on the seed locations and weights. In the next, we compare our method with the RT-based method. Since the method of Basselin et al. 2021 [30] requires similar weights of adjacent seeds, it cannot be used for the computation of power diagrams in general cases. Therefore, comparisons with Basselin et al. 2021 [30] are not provided.

For computing power diagrams in most cases, our method can provide higher speed than the RT-based method, as demonstrated by the comparisons of running time in Fig. 8. Given domain $[0, 1, 000]^d$, and from a random initialization with 10,000 seeds, we perform the centroidal Voronoi tessellation method [48,49] to obtain 100 sets of seeds. Each set is assigned random weights in 100 different ranges $[-w, w]$, where $w \in [0, 5]$. The total 10,000 sets of seeds are then tested by the RT-based method and our method. Each set of seeds is tested 10 times, and we plot their average running time in Fig. 8. The results of 1,000 seeds with four typical configurations (random locations with zero weights, random locations with random weights in $[-5, 5]$, uniform locations with zero weights, and uniform locations with random weights in $[-5, 5]$) in the 2D space are shown in Fig. 9.

Fig. 8 clearly shows that the computation time of our method drops with the improvement of seed uniformity, whereas the RT-based method remains stable, especially in the 3D case. However, our method outperforms the latter method for all the tests in the 2D and surface cases. In the 3D case, our method is more efficient only when the positions and weights of the seeds become uniform. The reason is that our method requires much more clipping tests between cell facets and bisectors than the RT-based method when the seeds are not uniform, especially in higher dimensional spaces.

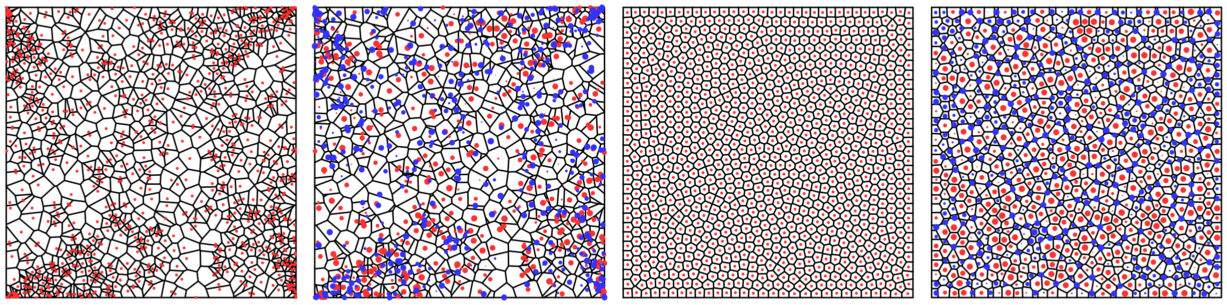
Fortunately, once the neighboring relationship between seeds is obtained, the clipping of each power cell is independent, which is suitable for parallel computation. We implement the parallel clipping for the RT-based method and ours and test the above data. The surfaces of running time are also drawn in Fig. 8. The figure shows that the parallel version of our method is more efficient than that of the RT-based method for all the cases. The reason is that the construction of KD-tree costs far less than that of regular triangulation, making our method have higher parallelism.

Table 1 shows a time comparison of the main steps between the RT-based method and ours on the examples in Figs. 1, 7, 9. As mentioned, our method costs most of its time in the clipping stage. By contrast, the RT-based method usually spends less time



(a) 2D case (b) Surface case (c) 3D case

Fig. 8. Time comparison with the RT-based method. The efficiency of our method is sensitive to the distribution of seed locations and weights, but it is still better than that of the RT-based method for most cases, except for the 3D cases with random seeds. We also compare the parallel versions of this two methods, and ours provides better efficiency for all the cases.



(a) 0.008 seconds (b) 0.008 seconds (c) 0.006 seconds (d) 0.007 seconds

Fig. 9. The time that our method computes power diagrams of 1,000 weighted seeds with different configurations. (a) random locations, zero weights; (b) random locations, random weights in [−5, 5]; (c) uniform locations, zero weights; (d) uniform locations, random weights in [−5, 5].

Table 1
Time comparison of the main steps between the RT-based method and our method (second).

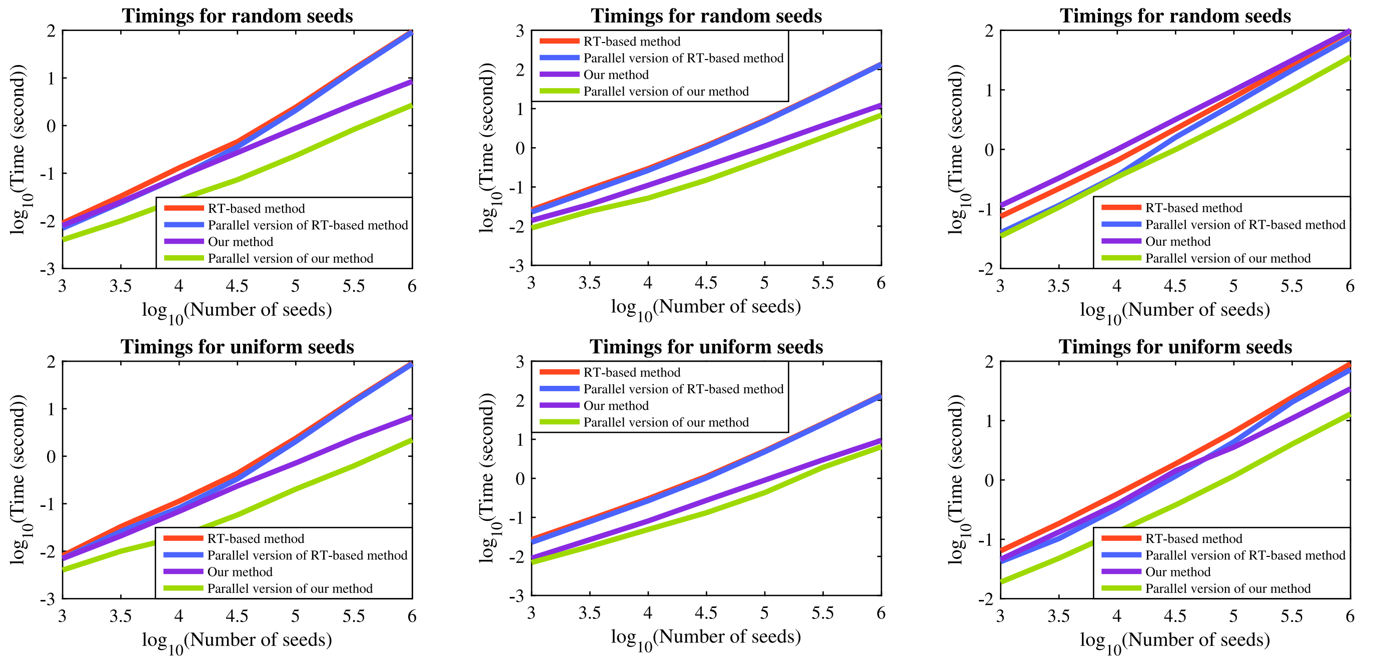
Figs.	#Region	#Seed	RT-based method					Our method				
			RT	Clipping		Total		KD-tree	Clipping		Total	
				Serial	Parallel	Serial	Parallel		Serial	Parallel	Serial	Parallel
1 (left)	1	100	0.001	0.001	0.001	0.002	0.002	0.0001	0.001	0.001	**0.0011**	**0.0011**
1 (middle)	5,000	1,000	0.011	0.059	0.026	0.07	0.037	0.001	0.04	0.019	**0.041**	**0.02**
1 (right)	15,663	3,000	0.039	1.273	0.676	1.312	0.715	0.001	1.057	0.475	**1.058**	**0.476**
7 (a)	1	500	0.002	0.001	0.001	0.003	0.003	0.0002	0.002	0.001	**0.0022**	**0.0012**
7 (b)	4,000	2,000	0.027	0.081	0.04	0.108	0.067	0.001	0.065	0.029	**0.066**	**0.03**
7 (c)	8,244	3,000	0.019	0.331	0.225	**0.35**	0.244	0.001	0.413	0.192	0.414	**0.193**
9 (a)	1	1,000	0.005	0.005	0.003	0.01	0.008	0.001	0.007	0.003	**0.008**	**0.004**
9 (b)	1	1,000	0.005	0.005	0.003	0.01	0.008	0.001	0.007	0.004	**0.008**	**0.005**
9 (c)	1	1,000	0.004	0.004	0.003	0.008	0.007	0.001	0.005	0.003	**0.006**	**0.004**
9 (d)	1	1,000	0.004	0.005	0.003	0.009	0.007	0.001	0.006	0.004	**0.007**	**0.005**

on clipping because the triangulation provides an explicit neighborhood for each seed. Their time of parallel computation is also given in the table, implying a better performance of the parallel version of our method.

Fig. 10 compares the computation time of the RT-based method, our method, and their parallel versions against the number of seeds with random and uniform locations, respectively. The input domains are the same as that in Fig. 8, and the weights are randomly generated in [0, 0.1] to reduce the impact of the hidden seeds. Our method is faster than the RT-based method and its parallel version for the 2D and surface cases, while the parallel version of our method outperforms the rest for all the cases. As the number of seeds increases, the RT-based method spends a larger part of its time constructing regular triangulation, resulting in a lower acceleration ratio of its parallel computation.

6.2. Applications

Our method and its parallel implementation can provide much higher computational efficiency than the state-of-the-art method. It is suitable for applications requiring the frequent computation of power diagrams.



(a) 2D case (b) Surface case (c) 3D case

Fig. 10. The computation time of the RT-based method, our method and their parallel versions against the number of seeds.

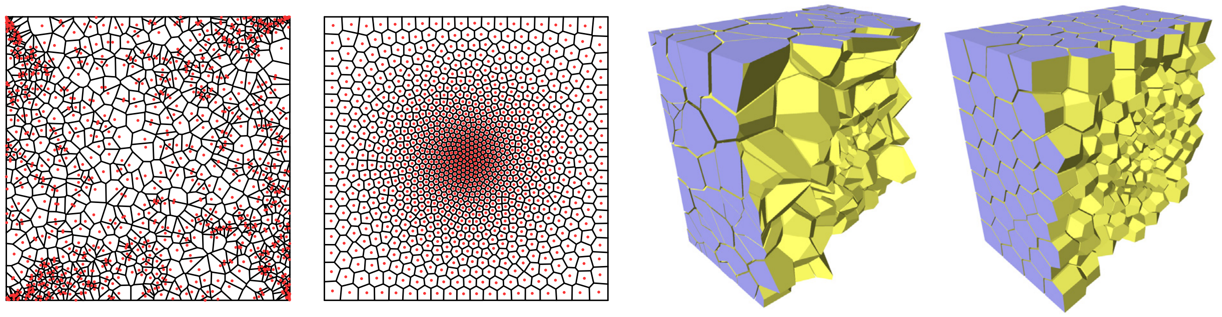


Fig. 11. Blue noise sampling with 1,000 points in 2D and 3D spaces. Left: 2D random initialization; middle left: 2D result after 100 iterations; middle right: 3D random initialization; right: 3D result after 100 iterations. Our method saves 7.97 s (2D) / 51.61 s (3D) in total over the RT-based method.

6.2.1. Blue noise sampling

Blue noise sampling is a core problem in computer graphics, and has been studied for many years [7,8,50]. de Goes et al. 2012 [7] and Xin et al. 2016 [8] presented capacity-constrained centroidal power diagrams to generate density-adapted uniform point sets. The results are obtained by minimizing the following energy function:

$$E(\{\mathbf{v}_i, w_i\}_{i=1}^{n}) = \sum_{i=1}^{n} \int_{\Omega_i} \rho(\mathbf{x}) \|\mathbf{x}-\mathbf{v}_i\|^2 d\mathbf{x} - \sum_{i=1}^{n} w_i \left(\int_{\Omega_i} \rho(\mathbf{x}) d\mathbf{x} - m \right),$$

where $\rho(\mathbf{x})$ is the density function, Ω_i the power cell, and m the capacity constraint. We adopt the optimization algorithm of Xin et al. 2016 [8] to search the minimum. Fig. 11 shows a blue noise sampling result after 100 iterations, where $\rho(\mathbf{x}) = 1/(\|\mathbf{x}\|^2 + 0.1)$. The result is obtained in 8.61 s (2D) / 258.82 s (3D) using our method for the computation of power diagrams in the algorithm. By contrast, using the RT-based method requires 16.58 s (2D) / 310.43 s (3D). With the improved uniformity of the seed, our method can significantly speed up the algorithm.

6.2.2. Adaptive remeshing

Given an input mesh, surface remeshing aims to produce an approximating mesh with higher quality. Power diagrams have been successfully applied to adaptive remeshing by Yan et al. 2014 [10]. The algorithm presented in [10] extends the farthest point optimization (FPO) method [51]. After initial sampling on the input surface and computing their restricted power diagram, in each iteration, FPO deletes each point and updates the power diagram, and then inserts a new point at the location that is farthest from the other points, where the power distance defines the distance between two points. Once the algorithm is completed, the resulting mesh can be extracted from the power diagram based on its duality. The computation and local update of the power diagram runs through the entire algorithm. Our method with an incremental KD-tree is applied to the power diagram computation in this algorithm. Note that the propagation strategy [27] is embedded in our method for the local update of the power diagram. An adaptive remeshing result with 5,000 points after 20 iterations is generated in 84.56 s, as shown in Fig. 12, where the mean curvature is used to decide the weights of points. In comparison, the algorithm using the RT-based method to update the power diagram consumes 0.93 s more per iteration.

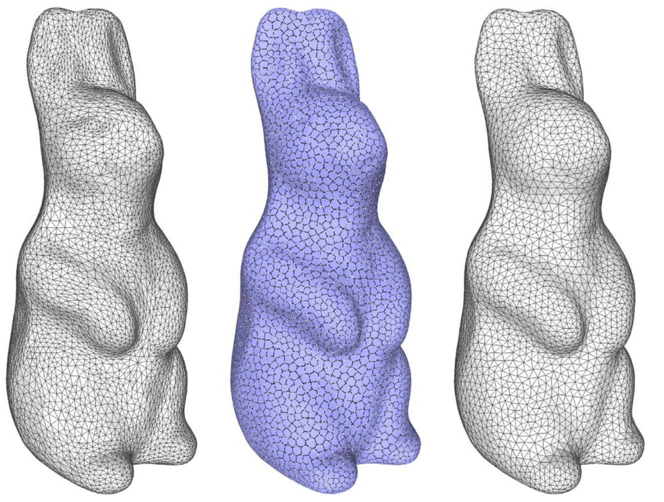


Fig. 12. Adaptive surface remeshing using farthest point optimization method [10], where the power diagram is updated by using our method. The weights of samples are determined by the mean curvature. Left: input surface; middle: result power diagram; right: duality of the power diagram.

6.2.3. Voronoi treemap generation

Treemap is a well-known structure for the visualization of attributed hierarchical data. Balzer and Deussen 2005 [52] and Nocaj and Brandes 2012 [44] proposed Voronoi treemap approaches to represent a polygonal shape to each node of the data. For a single layer of the data in a polygonal domain, they aim to generate a power diagram where, for each cell, the ratio of area to total area is the same as that of the associated attribute to total attribute. In each iteration, the Voronoi treemap approach [44] moves the seeds to the cell centroids and increases or decreases the weights based on the difference between the area share and the attribute share. Our method can also accelerate the generation of Voronoi treemaps. Computing the power diagrams using our method, the Voronoi treemap approach generates a result for 53,558 weighted nodes in 27.41 s, as shown in Fig. 13. The original algorithm, which relies on the convex hull computation, costs 36.25 s.

7. Conclusions and future work

We provide a variant for lifting the weighted seeds to a set of points in a higher dimensional space, then the power cells can be directly obtained by computing the intersections of the Voronoi cells of these lifting points and the original space. This property allows us to tailor a Voronoi generation method to compute the power diagrams. The proposed method initially generates the lifting points of the given weighted seeds and gradually clips the given domain to produce each power cell based on the sorted k-nearest neighbors of the corresponding lifting point. Experimental results show that our method and its parallel version perform better than the standard triangulation-based method for general cases in the 2D and 3D spaces.

The termination of the clipping procedure for each power cell is highly affected by the distribution of the lifting points. Thus, the efficiency of our method is sensitive to the seed locations and weights. We plan to modify the clipping strategy to avoid as many invalid cell-bisector clipping tests as possible to improve efficiency and weaken the influence of the distribution of the weighted seeds. We are also interested in extending the algorithm on the GPUs and exploring more applications.

CRediT authorship contribution statement

Yanyang Xiao: Conceptualization, Methodology, Investigation, Software, Visualization, Writing – original draft, Writing – review & editing. **Juan Cao:** Methodology, Validation, Formal analysis, Writing – review & editing. **Shaoping Xu:** Validation, Supervision, Writing – review & editing. **Zhonggui Chen:** Methodology, Validation, Formal analysis, Supervision, Writing- review & editing.

Declaration of competing interest

The authors declared no potential conflicts of interest with respect to the research, authorship, and publication of this article.

Data availability

No data was used for the research described in the article.

Acknowledgments

The research of Yanyang Xiao was supported by the National Natural Science Foundation of China (No. 62102174). The research of Juan Cao was supported by the National Natural Science Foundation of China (No. 62272402), the Xiamen Youth Innovation Funds (No. 3502Z20206029) and the Fundamental Research Funds for the Central Universities, China (No. 20720220037). The research of Zhonggui Chen was supported by the National Natural Science Foundation of China (No. 61972327) and Natural Science Foundation of Fujian Province, China (No. 2022J01001). The research of Shaoping Xu was supported by the National Natural Science Foundation of China (No. 62162043).

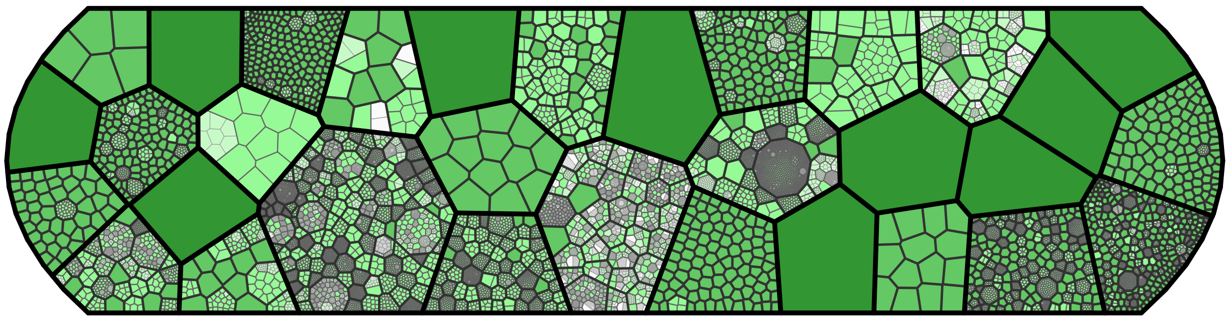


Fig. 13. Voronoi treemap of a hierarchy data with 53,558 weighted nodes. It costs 27.41 s to generate the result by the algorithm of Nocaj and Brandes [44], in which the power diagrams are computed by our method (implemented in C++), while the original algorithm takes 36.25 s (implemented in Java). A brighter color indicates a lower hierarchy level.

References

[1] Aurenhammer F. Voronoi diagrams—A survey of a fundamental geometric data structure. ACM Comput Surv 1991;23(3):345–405.
[2] Aurenhammer F. Power diagrams: Properties, algorithms and applications. SIAM J Comput 1987;16(1):78–96.
[3] Labelle F, Shewchuk JR. Anisotropic Voronoi diagrams and guaranteed-quality anisotropic mesh generation. In: Proceedings of the nineteenth annual symposium on computational geometry. New York, NY, USA: Association for Computing Machinery; 2003, p. 191–200.
[4] Boissonnat JD, Nielsen F, Nock R. Bregman Voronoi diagrams. Discrete Comput Geom 2010;(44):281–307.
[5] Gezalyan AH, Mount DM. Voronoi diagrams in the Hilbert metric. 2021, arXiv e-prints, arXiv:2112.03056.
[6] Papadopoulou E, Zavershynskyi M. The higher-order Voronoi diagram of line segments. Algorithmica 2016;74:415–39.
[7] de Goes F, Breeden K, Ostromoukhov V, Desbrun M. Blue noise through optimal transport. ACM Trans Graph 2012;31(6).
[8] Xin SQ, Lévy B, Chen Z, Chu L, Yu Y, Tu C, et al. Centroidal power diagrams with capacity constraints: Computation, applications, and extension. ACM Trans Graph 2016;35(6).
[9] Yan DM, Wonka P. Gap processing for adaptive maximal Poisson-disk sampling. ACM Trans Graph 2013;32(5).
[10] Yan DM, Guo J, Jia X, Zhang X, Wonka P. Blue-noise remeshing with farthest point optimization. Comput Graph Forum 2014;33(5):167–76.
[11] Budninskiy M, Liu B, de Goes F, Tong Y, Alliez P, Desbrun M. Optimal Voronoi tessellations with Hessian-based anisotropy. ACM Trans Graph 2016;35(6).
[12] Xiao Y, Chen Z, Cao J, Zhang YJ, Wang C. Optimal power diagrams via function approximation. Comput Aided Des 2018;102:52–60.
[13] Fiedler M, Alpers A. Power-SLIC: Fast superpixel segmentations by diagrams. 2020, arXiv e-prints, arXiv:2012.11772.
[14] Ma D, Zhou Y, Xin S, Wang W. Convex and compact superpixels by edge-constrained centroidal power diagram. IEEE Trans Image Process 2021;30:1825–39.
[15] de Goes F, Wallez C, Huang J, Pavlov D, Desbrun M. Power particles: An incompressible fluid solver based on power diagrams. ACM Trans Graph 2015;34(4).
[16] Zhai X, Hou F, Qin H, Hao A. Fluid simulation with adaptive staggered power particles on GPUs. IEEE Trans Vis Comput Graphics 2020;26(6):2234–46.
[17] Edelsbrunner H, Shah N. Incremental topological flipping works for regular triangulations. Algorithmica 1996;15(3):223–41.
[18] The CGAL Project. CGAL user and reference manual. 5.5 ed.. CGAL Editorial Board; 2022, URL https://doc.cgal.org/5.5/Manual/packages.html.
[19] Shamos MI, Hoey D. Closest-point problems. In: 16th Annual symposium on foundations of computer science 1975. 1975, p. 151–62.
[20] Green PJ, Sibson R. Computing Dirichlet tessellations in the plane. Comput J 1978;21(2):168–73.
[21] Brown KQ. Voronoi diagrams from convex hulls. Inform Process Lett 1979;9(5):223–8.
[22] Fortune S. A sweepline algorithm for Voronoi diagrams. Algorithmica 1987;2(1).
[23] Fortune S. Voronoi diagrams and Delaunay triangulations. In: Computing in Euclidean geometry. 1995, p. 225–65.
[24] Ledoux H. Computing the 3D Voronoi diagram robustly: An easy explanation. In: 4th International symposium on voronoi diagrams in science and engineering. 2007, p. 117–29.
[25] Yan DM, Lévy B, Liu Y, Sun F, Wang W. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. Comput Graph Forum 2009;28(5):1445–54.
[26] Lévy B, Bonneel N. Variational anisotropic surface meshing with Voronoi parallel linear enumeration. In: Jiao X, Weill J-C, editors. Proceedings of the 21st international meshing roundtable. Berlin, Heidelberg: Springer Berlin Heidelberg; 2013, p. 349–66.
[27] Yan DM, Wang W, Lévy B, Liu Y. Efficient computation of 3D clipped Voronoi diagram. In: Mourrain B, Schaefer S, Xu G, editors. Advances in geometric modeling and processing. Berlin, Heidelberg: Springer Berlin Heidelberg; 2010, p. 269–82.
[28] Ray N, Sokolov D, Lefebvre S, Lévy B. Meshless Voronoi on the GPU. ACM Trans Graph 2018;37(6).
[29] Liu X, Ma L, Guo J, Yan D-M. Parallel computation of 3D clipped Voronoi diagrams. IEEE Trans Vis Comput Graphics 2022;28(2):1363–72.
[30] Basselin J, Alonso L, Ray N, Sokolov D, Lefebvre S, Lévy B. Restricted power diagrams on the GPU. Comput Graph Forum 2021;40(2):1–12.
[31] Rycroft CH. VORO++: A three-dimensional Voronoi cell library in C++. Chaos 2009;19(4).
[32] Ash PF, Bolker ED. Generalized Dirichlet tessellations. Geom Dedicata 1986;20:209–43.
[33] Lévy B. A numerical algorithm for L_2 semi-discrete optimal transport in 3D. ESAIM: Math Model Numer Anal - Modél Math Et Anal Num 2015;49(6).
[34] Hsieh HH, Tai WK. A simple GPU-based approach for 3D Voronoi diagram construction and visualization. Simul Model Pract Theory 2005;13(8):681–92.
[35] Rong G, Tan TS. Jump flooding in GPU with applications to Voronoi diagram and distance transform. In: Proceedings of the 2006 symposium on interactive 3D graphics and games. I3D '06, New York, NY, USA: Association for Computing Machinery; 2006, p. 109–16.
[36] Majdandzic I, Trefftz C, Wolffe G. Computation of Voronoi diagrams using a graphics processing unit. In: 2008 IEEE international conference on electro/information technology. 2008, p. 437–41.
[37] Lawson CL. Transforming triangulations. Discrete Math 1972;3(4):365–72.
[38] Bowyer A. Computing Dirichlet tessellations. Comput J 1981;24(2):162–6.
[39] Watson DF. Computing the n-dimensional Delaunay tessellation with application to Voronoi polytopes. Comput J 1981;24(2):167–72.
[40] Dwyer RA. A faster divide-and-conquer algorithm for constructing Delaunay triangulations. Algorithmica 1987;2:137–51.
[41] McCullagh MJ, Ross CG. Delaunay triangulation of a random data set for isarithmic mapping. Cartogr J 1980;17(2):93–9.
[42] Kolingerová I, Kohout J. Optimistic parallel Delaunay triangulation. Vis Comput 2002;18:511–29.
[43] Lo S. Parallel Delaunay triangulation in three dimensions. Comput Methods Appl Mech Engrg 2012;237–240:88–106.
[44] Nocaj A, Brandes U. Computing Voronoi treemaps: Faster, simpler, and resolution-independent. Comput Graph Forum 2012;31:855–64.
[45] Zheng L, Gui Z, Cai R, Fei Y, Zhang G, Xu B. GPU-based efficient computation of power diagram. Comput Graphics 2019;80:29–36.
[46] Sutherland IE, Hodgman GW. Reentrant polygon clipping. Commun ACM 1974;17(1):32–42.
[47] Blanco JL, Rai PK. NanoFLANN: A C++ header-only fork of FLANN, a library for nearest neighbor (NN) with KD-trees. 2014, https://github.com/jlblancoc/nanoflann.
[48] Du Q, Faber V, Gunzburger M. Centroidal Voronoi tessellations: Applications and algorithms. SIAM Rev 1999;41(4):637–76.
[49] Liu Y, Wang W, Lévy B, Sun F, Yan DM, Lu L, et al. On centroidal Voronoi tessellation-energy smoothness and fast computation. ACM Trans Graph 2009;28(4):101:1–101:17.
[50] Chen Z, Yuan Z, Choi YK, Liu L, Wang W. Variational blue noise sampling. IEEE Trans Vis Comput Graphics 2012;18(10):1784–96.
[51] Schlömer T, Heck D, Deussen O. Farthest-point optimized point sets with maximized minimum distance. In: Proceedings of the ACM SIGGRAPH symposium on high performance graphics. New York, NY, USA: Association for Computing Machinery; 2011, p. 135–42.
[52] Balzer M, Deussen O. Voronoi treemaps. In: Proceedings of the proceedings of the 2005 IEEE symposium on information visualization. Washington, DC, USA: IEEE Computer Society; 2005, p. 7–14.