Advances in Engineering Software 132 (2019) 7–17

Contents lists available at ScienceDirect

Advances in Engineering Software

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

Research paper

Approximation of Voronoï cell attributes using parallel solution

Jan Mašek*, Miroslav Vořechovský

Institute of Structural Mechanics, Brno University of Technology, Veveří 331/95, Brno 602 00, Czech Republic

ARTICLE INFO

Keywords:
A. Voronoï diagram
B. approximation by rasterization
C. parallel solution
D. error of hyper-dimensional raster

ABSTRACT

This paper concerns an algorithm for fast parallel approximation of selected attributes of hyper-dimensional Voronoï cells in a unit hypercube. The presented algorithm does not require the construction of a corresponding Voronoï diagram (usually by employing the Quick Hull algorithm) which typically is a highly computationally demanding task, especially when performed in higher dimensions. The algorithm is suitable for both the clipped and periodic variants of Voronoï tessellation and provides a significant speedup in a convenient range of practical usage. For the purposes of approximation of selected scalar properties of Voronoï cells, only the distances of points in sample are evaluated using an adequately fine underlying orthogonal mesh. The algorithm estimates e.g. the volumes and centroids of Voronoï cells, radii and coordinates of centers of the corresponding Delaunay hyper-circles. The paper also provides quite accurate explicit error estimators for the extracted attributes due to rasterization and thus the user is given a control over the accuracy by selecting an appropriate discretization density. Among numerous fields in which Voronoï diagrams are being utilized, the authors are concerned with optimization of point samples for the Design of Experiments and also with weighting of integration points in Monte Carlo type integration. In these applications, selected scalar topological descriptors of Voronoï diagrams are being repeatedly computed. As the full Voronoï diagram is not of interest but the resulting cell volumes or shape descriptions have to be repeatedly computed, the presented parallel solution seems highly suitable for these applications.

1. Introduction

The Voronoï diagram has become a classical construct of computational geometry, see [2,67]. Consider a domain of a general integer dimension, D, containing N_s points ("seeds", "sites" or "generators"). The task of the Voronoï tessellation is to divide the given space into $i = 1, \ldots, N_s$ convex polyhedrons of dimension D that encompass regions where all points are closer to ith point than to any other of N_s *generating* points. A Voronoï tessellation emerges by uniform radial growth from seeds outwards.

Applications of Voronoï diagrams can be found in numerous fields of research and industry (for example biology [42], Thiessen polygons in hydrology [12,59], chemistry, astrophysics, fluid dynamics [62], computational physics [35], epidemiology [34], medical diagnosis [57], engineering, geometry [70], informatics [46], etc.) The geometric sense of Voronoï diagrams is used for solving tasks like searching for the nearest neighbor, the largest empty circle or estimating the roundness of an object. Technical applications include e.g. construction of meshes in bordered or periodically repeated domains, see [73] and [72].

This paper is motivated by applications of Voronoï tesselation in the field of Design of Experiments. There, the design domain (usually transformed into a convex unit hypercube $[0, 1]^D$) is being filled by a set of N_s distinct points (a "sample" or "design"). The selection of optimal sampling points from a unit hypercube is an old problem that finds many applications in science and industry [5,11,18,19,21,24,25,29,36,40,43,50]. A Voronoï diagram can be used for evaluation of uniformity of a point sample and possibly for further optimization of the point layout, see [33,52,53]. Additionally, it has been also proposed to use the volume of each cell in the Voronoï diagram as the weight of the corresponding integration point as used in the weighted numerical integration of Monte Carlo type [68,69]. There is a large potential in these applications to utilize an accelerated extraction of selected scalar descriptors of Voronoï diagrams.

The construction of the exact Voronoï diagram is a computationally demanding task. In the worst cases, the solution complexity approaches $O(N_s^2)$, see e.g. the QuickHull algorithm [4]. Moreover, the computational time and the storage demands grow exponentially with the dimension of the design domain, D. On top of these computations, the eventual enumeration of volumes of all D-dimensional cells, see [10], has to be executed.

For many applications, the exact geometrical description of Voronoï cells is not needed; only selected attributes of the regions are of interest.

* Corresponding author. Institute of Structural Mechanics, Brno University of Technology, Veveří 331/95, 602 00 Brno, Czech Republic.
E-mail addresses: jan.masek1@vut.cz (J. Mašek), vorechovsky.m@vut.cz (M. Vořechovský).

https://doi.org/10.1016/j.advengsoft.2019.03.012
Received 1 November 2018; Received in revised form 28 February 2019; Accepted 31 March 2019

Nomenclature

- Generating points/sites (point sample): a set of points that are an input for construction of its corresponding Voronoï diagram,
- Voronoï diagram: a set of identified Voronoï regions/cells belonging to each generating points,
- Voronoï region/cell: a continuous region that encapsulates all points within the design domain that, according to a specific distance metric, are closer to the corresponding generating point than to any other of generating points,
- Grid cells/pixels: a square-shaped elements that are utilized for discretization of Voronoï regions,
- Grid node: the centroid of a grid cell that is used for comparison of the distance between generating points and each grid cell,
- Tiles: chunks of data that are processed by blocks of threads on GPU in parallel.

The authors of [48] proposed to solve the largest empty sphere problem (a byproduct of Voronoï tessellation) in the multidimensional space by using a popular stochastic search approach, evolutionary algorithm (EA). They followed the path presented in [39]. In the present paper, a different approach to approximation is selected. The presented contribution proposes a fast parallelized implementation of approximation of volumes of the Voronoï cells without the need of constructing the Voronoï diagram itself. This approach turns out to be useful in cases where the actual shape of the diagram is not of interest but its certain scalar properties have to be repeatedly computed for optimization purposes.

The presented contribution proposes to conduct a rasterization of the hyper-dimensional design space using an underlying mesh of nodes. Because a mesh of equal and independent nodes is highly suitable for a kind of massively parallel execution, the Nvidia CUDA [51] platform is utilized for implementation of the solution.

2. Discretization approach

The essence of the solution method is based on division of the entire design space into evenly distributed sub-regions. Further used is the division using a regular orthogonal grid of cells. The side of a single cell of such a hyper-dimensional orthogonal grid is considered as $1/n_g$, where n_g is number of grid cells along each axis of the hypercubical domain. The total number of grid cells N_g is then simply $N_g = (n_g)^D$.

Cells of a regular orthogonal grid exhibit advantageous properties of equal hyper-volumes and center of gravity located in the middle of each cell. Such a grid of equal cells is a convenient option in conjunction with parallel execution. The center of gravity of each *grid cell* is further considered to be a *grid node* within a regular orthogonal grid used in the approximation process, see e.g. [56]. All of N_g created nodes are defined with their spatial coordinates.

From now on, we follow the definition of the Voronoï diagram combined with Euclidean metric. Position of each node $\boldsymbol{u}_j$ can be compared with position of each site $\boldsymbol{x}^{(i)}$ from the N_s input sites of the Voronoï diagram. Their mutual Euclidean distance $d(\boldsymbol{x}^{(i)}, \boldsymbol{u}_j)$ is computed as follows:

$$d(\boldsymbol{x}^{(i)}, \boldsymbol{u}_j) = \sqrt{\sum_{v=1}^{D} [\Delta_v(\boldsymbol{x}^{(i)}, \boldsymbol{u}_j)]^2}, \tag{1}$$

where $\Delta_v(\boldsymbol{x}^{(i)}, \boldsymbol{u}_j)$ is the projection of the mutual distance between the site $\boldsymbol{x}^{(i)}$ and the grid node (center of the "pixel") $\boldsymbol{u}_j$ along axis v:

$$\Delta_v(\boldsymbol{x}^{(i)}, \boldsymbol{u}_j) = x_v^{(i)} - u_{j,v}. \tag{2}$$

The particular node $\boldsymbol{u}_j$ is deemed to belong to the Voronoï cell $C^{(i)}$ associated with its corresponding site $\boldsymbol{x}^{(i)}$ iff this is the closest of all sites in the sample. The following condition is then met:

$$C^{(i)} = \{\boldsymbol{u} \in \mathbb{R}^D | \ \forall \ k \neq i: d(\boldsymbol{x}^{(i)}, \boldsymbol{u}_j) \leq d(\boldsymbol{x}^{(k)}, \boldsymbol{u}_j)\}. \tag{3}$$

Note that the square root operation in Eq. (1) can be omitted to save computational time as it does not influence the result of distance comparison. By conducting such a comparison of distances between each of N_g grid nodes with each of N_s Voronoï sites, an array of N_g integers is obtained, containing indexes of the closest site to each grid node. Such a grid of nodes, if *sufficiently dense*, may approximate the analytic description of the desired scalar properties of the Voronoï diagram.

Fig. 1illustrates the results of approximation of a Voronoï diagram of a Latin Hypercube (LH) point sample containing $N_s = 16$ sites. The first three diagrams from the left show an approximated tessellation in a bordered domain. This is sometimes called the *clipped tessellation* and it is achieved by mirroring the sites with respect to all boundaries of the design domain. The boundary Voronoï cells are then being bordered by the boundaries of the domain.

Another option is the *periodic tessellation*: while comparing distances between the grid nodes and Voronoï sites, it is easily possible to switch to periodic boundary conditions, if desired. Then, the shortest distance $\overline{d}(\boldsymbol{x}^{(i)}, \boldsymbol{u}_j)$ within the periodically repeated domain is measured, using the *minimum image convention*:

$$\overline{d}(\boldsymbol{x}^{(i)}, \boldsymbol{u}_j) = \sqrt{\sum_{v=1}^{D} [\overline{\Delta}_v(\boldsymbol{x}^{(i)}, \boldsymbol{u}_j)]^2}, \tag{4}$$

where $\overline{\Delta}_v(\boldsymbol{x}^{(i)}, \boldsymbol{u}_j)$ is the projection of the distance between grid node $\boldsymbol{u}_j$ and the closest of periodically repeated images of site $\boldsymbol{x}^{(i)}$ along axis v:

$$\overline{\Delta}_v(\boldsymbol{x}^{(i)}, \boldsymbol{u}_j) = \min[\Delta_v(\boldsymbol{x}^{(i)}, \boldsymbol{u}_j), 1 - \Delta_v(\boldsymbol{x}^{(i)}, \boldsymbol{u}_j)]. \tag{5}$$

The approximated Voronoï diagram using periodic boundary conditions is illustrated in the rightmost plot in Fig. 1.

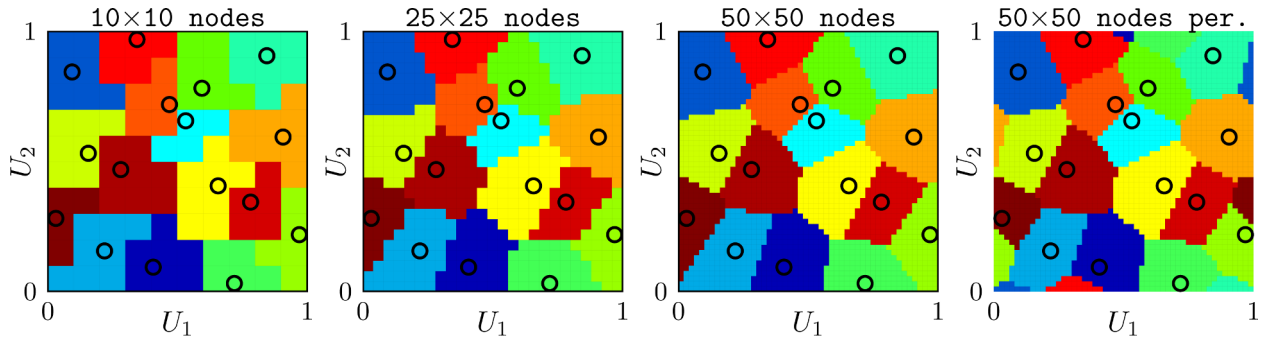

Fig. 1. The convergence of a rasterized Voronoï diagram towards an exact solution.

3. Attributes extracted from the rasterized Voronoï diagram

As soon as the index of the closest site is obtained for each of the grid nodes, it is possible to extract certain desired attributes of such a rasterized Voronoï diagram. For purposes of the use case of sample optimization employed by the authors, attributes discussed below are approximated.

3.1. Cell volumes

The volumes of Voronoï cells are the most convenient scalar property that can be extracted from the rasterized diagram. These are of interest for many use cases. For example in *weighted Monte Carlo integration* [3,71], it has been proposed recently [68,69] to use volumes of Voronoï cells as weights of individual integration points. Also, it is known that equal ("uniform") volumes of Voronoï cells may help in selecting sites (integration points) for each cell in order to obtain "uniformly" distributed sites in a domain.

The actual approximated volume $V^{(i)}$ of each Voronoï cell $C^{(i)}$ is then a result of a simple summation of known volumes v_j of all respective sub-regions deemed to belong to each cell $C^{(i)}$. Moreover, since the grid cell volumes are equal, $v = (1/n_g)^D$, the actual computation shrinks merely to a sum of grid cell volumes of all $N^{(i)}$ grid cells belonging to the Voronoï cell $C^{(i)}$:

$$V^{(i)} \approx \hat{V}^{(i)} = \sum_{j=1}^{N^{(i)}} v = \frac{N^{(i)}}{{n_g}^D}. \tag{6}$$

The error introduced by such an approximation is estimated in Section 4.1.

3.2. Cell centroids

Another desired characteristic of a Voronoï diagram are the coordinates of centroids, or centers of gravity, of all Voronoï cells. There are many applications in which cell centroids are needed, e.g. in Centroidal Voronoï tessellation (CVT) [20]. CVT finds applications in optimal mesh generation, optimal quadrature, optimal quantization, clustering and also in data compression. Also, many patterns seen in nature are closely approximated by a CVT (cells of the cornea or the breeding pits of the male tilapia). Along the lines of optimal sampling, it has been proposed in [55] to employ CVT as a sampling method: in the iterative Lloyd's algorithm, the centroids of individual cells are used as new sites for the next Voronoï tessellation. The iterations are halted when the positions of sites and the respective cell centroids coincide. In this application, repeated identification of Voronoï cells and computation of their centroids is necessary.

Coordinates of the centers of gravity of Voronoï cells, $\mathbf{cg}^{(i)} = \{\cdots, \mathrm{cg}_v^{(i)}, \cdots\}$, $v = 1, \ldots, D$, can be rather efficiently approximated. Because all grid cells are of equal volume (weight), in a bordered domain (clipped tessellation), the coordinate in the vth dimension of a center of gravity of the ith Voronoï cell V_i is simply estimated by the average of coordinates of nodes belonging to the cell V_i:

$$\mathrm{cg}_v^{(i)} \approx \hat{\mathrm{cg}}_v^{(i)} = \frac{1}{N^{(i)}} \sum_{j=1}^{N^{(i)}} u_{j,v}^{(i)}. \tag{7}$$

In the case of periodically repeated design domain, such plain averaging is not possible. However, the desired centroid coordinates can be still computed by a one-pass algorithm. The approximated coordinates of Voronoï cell centroids need to respect the periodic boundary conditions. This is ensured by evaluating *provisional* (current) centroid coordinates after each of $N^{(i)}$ steps of averaging.

If there are $N^{(i)}$ grid cells belonging to the ith Voronoï cell, the computation of the coordinate of the centroid of the ith Voronoï cell in dimension v has to be conducted sequentially in $N^{(i)}$ steps in each of which, one has to check whether there is a closer periodic distance between the current approximated Voronoï cell centroid and the particular grid node. The (j)th step $(1 < j \leq N^{(i)})$ of averaging must be conducted as follows.

First, a decision must be made about possible update of the position of the newly considered (current, (j)th) grid node $u_{j,v}^{(i)}$ depending on its relative position to the provisional centroid of all preceding $(j-1)$ nodes, i.e. $\mathrm{cg}_v^{(i)}(j-1)$:

$$\begin{aligned} &\text{if} \quad (|(u_{j,v}^{(i)} + 1) - \mathrm{cg}_v^{(i)}(j-1)| < 0.5)\; u_{j,v}^{(i)} \mathrel{+}= 1, \\ &\text{else if}\, (|(u_{j,v}^{(i)} - 1) - \mathrm{cg}_v^{(i)}(j-1)| < 0.5)\; u_{j,v}^{(i)} \mathrel{-}= 1. \end{aligned} \tag{8}$$

Simply, if the newly considered node is far from the provisional centroid, a peridic image that is closer ($\pm$ 1) is considered instead. Then, the current centroid coordinate $\mathrm{cg}_v^{(i)}(j)$ in dimension v is obtained using the following recurrence formula for average:

$$\mathrm{cg}_v^{(i)}(j) = \frac{u_{j,v}^{(i)} - \mathrm{cg}_v^{(i)}(j-1)}{j} + \mathrm{cg}_v^{(i)}(j-1). \tag{9}$$

After the last step of such an averaging, it is needed to check whether the resulting centroid coordinates appear within the original design domain or their periodically repeated image has been obtained. If so, such a coordinate is simply translated back into the design domain by either adding of subtracting 1.

3.3. Center and radius of the largest delaunay hyper-circle

The search for the largest empty hyper-circle within a set of sites is a classical task of computational geometry [1,14,16,17,22,32,45,65]. Its position (coordinates of center) and radius are in applications used for description of the emptiest region within a point sample. Particularly, the miniMax optimization criterion in the field of Designs of Experiments, see e.g. [33,52,53], utilizes the largest empty hyper-circle to unveil the group of $D+1$ points that shall be brought closer to each other (towards the center of the circle), see Fig. 2 left.

A typical approach to obtain the position and radius of the largest

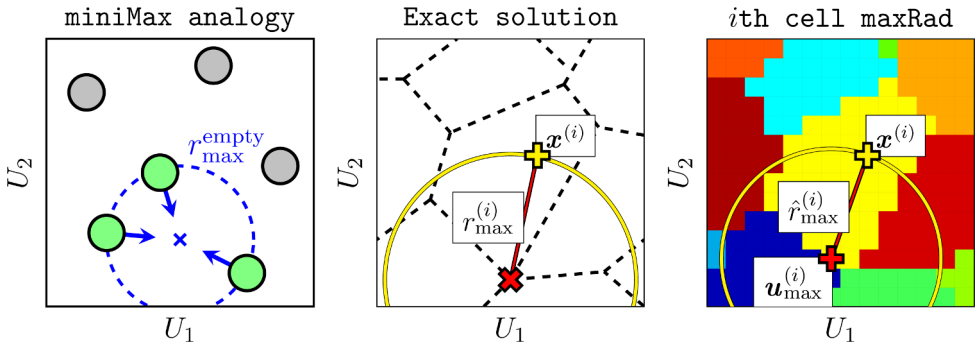

Fig. 2. Approximation of the center of the largest empty Delaunay circle and its radius.

empty hyper-circle is by executing either the construction of Voronoï diagram or the Delaunay triangulation that is its *dual graph*. Another possibility is to find the largest from all $\binom{N_s}{D+1}$ empty hyper-circles determined by any unique group of $D+1$ sites.

For approximation of the largest of empty Delaunay circles belonging to each site, see Fig. 2 center, the algorithm analyses the distances between the site and its respective nodes. Based on the knowledge of indices of nodes that form each Voronoï cell, their respective coordinates are generated and mutual distances to the cell centroid are compared. The furthermost node of all is then considered to be an approximated center of the largest empty Delaunay circle and the distance between the centroid and the furthermost node to be its radius, as shown in Fig. 2 right. For each generating point i, the radius is computed as the most distant grid cell centroid:

$$\hat{r}_{\max}^{(i)} = \max_{1\le j\le N^{(i)}} d(\boldsymbol{x}^{(i)}, \boldsymbol{u}_j^{(i)}). \tag{10}$$

In the periodic version, metric d is replaced by $\bar{d}$. The grid cell that maximizes this radius is the approximated vertex, $\boldsymbol{u}_{\max}^{(i)}$, see Fig. 2 right for illustration. The value of the miniMax criterion, ϕ_{mM}, is then approximated by the largest radius of all N_s estimated radii:

$$\phi_{\mathrm{mM}} \approx \max_{1\le i\le N_s} \hat{r}_{\max}^{(i)}. \tag{11}$$

The errors in these distance approximations are discussed in Section 4.2.

4. Analysis of approximation errors

The proposed approximation of Voronoï tessellation by "rasterization" or "pixelization" of the domain introduces errors in the approximated properties. It is important to know the loss of information and accuracy inherent in the grid, and its relationship to the grid cell size. The grid cells form hypercubes of edge length $a = 1/n_g$ and their volume is $v = a^D = 1/{n_g}^D$. The maximum distance within a single grid cell (sometimes referred to as the "pixel") is its spatial diagonal:

$$l_d = a\sqrt{D} = \frac{\sqrt{D}}{n_g}. \tag{12}$$

4.1. Errors in cell volumes

The error in the ith cell volume is defined as:

$$\epsilon_V = V^{(i)} - \hat{V}^{(i)} \qquad i = 1, \ldots, N_s, \tag{13}$$

where $V^{(i)}$ is the exact cell volume obtained using the QuickHull algorithm (either clipped or periodic Voronoï tessellation) and $\hat{V}^{(i)}$ is the approximated solution obtained with a certain rasterization using Eq. (6).

The problem of error in estimation of the area of 2D regions using the dot-grid method (point-counting method) is known from the vector-to-raster conversion of maps and it has received an attention mainly in the70′s and80′s, [6,7,12,13,15,23,27,28,38,41,49,60,61]. In all these publications, it was shown that the rasterization error in 2D is proportional to the cell size, a, and therefore is inversely proportional to the number of cells per edge, n_g. Various approaches were employed to estimate the influence of rasterization method, size of raster cell and map complexity on this error. However, the solutions are limited only to rasterization of 2D maps. The error in estimation of a volume of a convex polyhedron by point counting in a fine orthogonal grid in a general dimension, D, is somewhat more complicated. The proposed derivation follows.

The volume of ith Voronoï cell is estimated as a sum of grid cells that "occupy" the Voronoï region, see Eq. (6). The centroid rule is applied, i.e. "occupied by" is interpreted such that the cell is counted if its centroid falls within the region. The estimated volume then involves both (i) points belonging to grid cells (pixels) that are entirely inside the Voronoï cell and, (ii) points that represent cells intersected by the exact Voronoï boundary. The error in cell volume is affected only by the boundary pixels. There are two possibilities: either the grid cell is counted as a whole but part of it lies outside the Voronoï cell (it contributes to overestimation of the volume) or, the grid cell is not counted even though a part of it lies inside the Voronoï cell (it contributes to underestimation). In the following, we disregard the rare cases where the vertex connecting two or more boundaries of a Voronoï cell is inside the grid cell (pixel). If one passes a random boundary through a grid cell (e.g. a random line through a small square), then the average misclassified volume is zero. The maximum absolute error (overestimated volume or underestimated volume) by a single grid cell j equals half of its volume, i.e.

$$\epsilon_{\mathrm{b,max}} = \frac{v}{2} = \frac{1}{2\,{n_g}^D}. \tag{14}$$

For a single grid cell, the distribution is generally a bimodal distribution symmetrical with respect to zero and ranging between $\pm\,\epsilon_{\mathrm{b,max}}$. In the present analysis, though, this distribution of volume error in a single grid cell is not sufficient to capture the behavior of a cut through an orthogonal grid of cells. The real Voronoï cell boundary is straight and crosses *many grid cells under the same angle*. Therefore the error contribution coming from a single boundary, b, is composed of many *dependent* contributions from individual grid cells (pixels).

Indeed, it is plausible to assume that each Voronoï cell (a polyhedron) is bounded by a finite number, N_b, of boundary hyper-planes (lines in 2D, planes in 3D, etc.). We also assume that these N_b boundaries are about the same extent (length, area, etc.), meaning that the polyhedra are relatively regular geometrical objects resembling circles, spheres, hyperspheres etc. Assume also that the number of pixels crossing all the identical N_b boundaries is known, denoted as n_b, see Fig. 3. Then assume that each boundary crosses roughly the same number of grid cells: n_b/N_b. We now assume that each boundary is sufficiently large (a long line in 2D, plane in 3D, etc.) to achieve asymptotic properties. The error from such a single boundary is dependent on the angle between the boundary and the grid. In the worst possible situation, the boundary is aligned with the grid (see boundary No. 1 in Fig. 3) and the error is proportional to the length of a boundary. Then the error of a single boundary of a single Voronoï cell is just a multiple of the error arising by such a single grid cell:

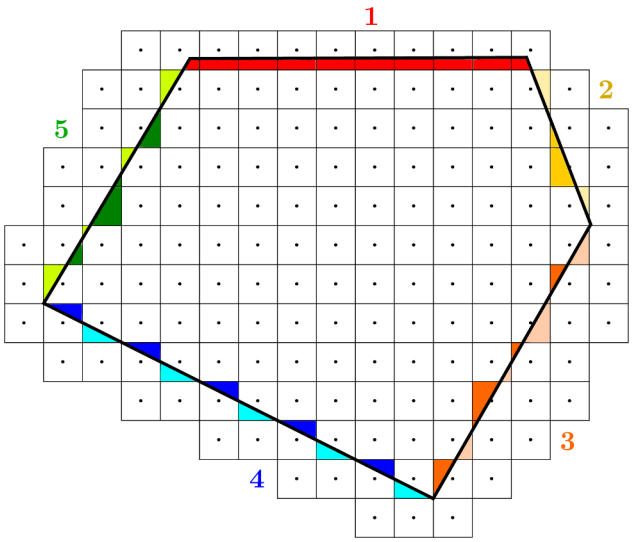

Fig. 3. Illustration of the volume error contributors for a 2D Voronoï cell with $N_b = 5$ boundaries, $n_b = (10 + 5 + 10 + 10 + 8) = 43$ boundary grid cells.

$$\frac{n_b}{N_b}\cdot\epsilon_b, \tag{15}$$

where ϵ_b is a random variable with an asymptotically *uniform distribution* varying between $-\epsilon_{b,max}$ and $+\epsilon_{b,max}$ (see Eq. 14). The mean value of ϵ_b equals zero and the standard deviation equals:

$$\sigma_{max} = \frac{\epsilon_{b,max}}{\sqrt{3}} = \frac{1}{{n_g}^D}\frac{\sqrt{3}}{6}. \tag{16}$$

If, however, the boundary is not aligned with the grid edge, the error can also become zero, see boundary No. 4 in Fig. 3 where the same amount of volume erroneously taken as the interior is balanced by the rasterization in the cells that are considered outside the Voronoï cell boundary. We assume that, in an average situation, the standard deviation is one half of the extreme standard deviation σ_{max}, i.e:

$$\sigma_{ave} = \epsilon_{b,max}\frac{\sqrt{3}}{6} = \frac{1}{{n_g}^D}\frac{\sqrt{3}}{12}. \tag{17}$$

One could argue that the error should rather be described as a sum of contributions from individual cells. However, the fact that the boundary is "straight" (a line, plane,...) gradually modifies the probability density of ϵ_b from a symmetrical bimodal distribution into uniform distribution for large enough boundaries because the contributions become highly *dependent*.

The total error in volume of a single Voronoï cell can be calculated as:

$$\epsilon_V = \sum_{b=1}^{N_b}\frac{n_b}{N_b}\cdot\epsilon_b, \tag{18}$$

i.e. the error composes of N_b contributions weighted by boundary "lengths" (n_b/N_b is the number of boundary pixels per single boundary). Taking the boundary "lengths" as equal for a Voronoï cell allows to write:

$$\epsilon_V = n_b\cdot\underbrace{\frac{1}{N_b}\sum_{b=1}^{N_b}\epsilon_b}_{V_b} = n_b\cdot V_b. \tag{19}$$

In this formula, averaging over N_b random variables is performed. We can assume these variables to be independent as the real Voronoï cell samples well all possible orientations of the boundary with respect to the approximating grid. The result of this averaging is a random variable V_b with the mean value and standard deviation given by:

$$\mu_b = 0, \tag{20}$$

$$\sigma_b = \frac{\sigma_{ave}}{\sqrt{N_b}} = \frac{\sqrt{3}}{{n_g}^D\cdot 12\cdot\sqrt{N_b}}. \tag{21}$$

The probability density of variable V_b is a generalized Bates distribution varying between $\pm\,\epsilon_{b,max}$. With an increasing number of Voronoï cell boundaries, N_b, the variance decreases proportionally to $1/N_b$ and the density gradually changes from uniform (for a single boundary) through triangular (two boundaries) to a bell-shaped distribution (ultimately Gaussian for infinite N_b).

We remark that in the worst possible situation of a Voronoï cell having *all* boundaries aligned with a grid (a rectangle, hypercube, etc.), and such a cell being randomly shifted with respect to the grid, the standard deviation σ_b in Eq. (21) is doubled, i.e. one must plug in σ_{max} from Eq. (16) instead of σ_{ave} from Eq. (17).

The number of boundaries of a single Voronoï cell, N_b, is, for random positions of the generators, also a random variable. The reason is that Voronoï tessellation may fill the space with various types of convex polyhedra. The boundaries in 2D tessellations are formed by "edges", in 3D by "faces", in 4D by "cells", etc. The average count of these boundary objects for Voronoï tessellation with randomly distributed generating points (Poisson point process) have been, along with statistical distributions of other attributes, extensively studied by many authors [2,8,9,26,30,31,37,44,47,54,58,63,64]. Based on these studies, we take the average number of edges of polygons in 2D as $N_b = 6$, and the number of faces in 3D as $N_b = 48\pi^2/35 + 2 \approx 15.5$ (this comes out from the Euler–PoincarÃ©characteristic stating that the number of vertices plus faces minus edges equals two). The average number of "cells" for 4-polytopes from Voronoï tessellation of random point set is not known. There are known *regular convex 4-polytopes* that have their numbers of "cells" varying between 5 (5-cell, pentachoron or pentatope), to 8, 16, 24, 120 or even 600 "cells" (hexacosichoron or tetraplex). We have made our own analysis of the average number of boundaries per Voronoï region in QuickHull and we confirm the above results for 2D and 3D domains. In 4D, we have found $N_b \approx 39$ and, in 5D $N_b \approx 86$. Linear regression suggests $N_b(D) \approx N_b(D-1)\cdot 2.3 + 1.7$.

The last step in the evaluation of the error is the determination of the number of grid cells, n_b, that intersect the N_b boundaries of a typical Voronoï cell, $C^{(i)}$. The key is that the extent of a boundary can be estimated as a function of the total Voronoï cell volume. For simplicity, we approximate the convex polyhedra by D-dimensional hyper-spheres (balls) of radius R. It is known that topologically, a D-simplex is equivalent to a D-ball – it is an D-dimensional manifold with corners. The simplification helps to estimate the ratio between the D-dimensional volume and $(D-1)$-dimensional boundary of the Voronoï cell. It is known that the volume of D-ball with radius R equals

$$V(R) = \frac{\pi^{\frac{D}{2}}}{\Gamma\left(\frac{D}{2}+1\right)}R^D, \tag{22}$$

where Γ is the Gamma function. The boundary of such an D-ball (a "surface") reads simply:

$$S(R) = D\frac{V(R)}{R}. \tag{23}$$

This boundary intersects n_b grid cells. We say that, on average, a grid cell is cut by such a boundary in the half-way between the largest possible cut (a diagonal of a 2D cell, a diagonal plane in 3D, etc.) and just a vertex. Such an average cut has the size (length, area, volume, ···) equal to $(l_d/2)^{D-1}$, see e.g. the various cut lengths appearing in 2D in Fig. 3. The number of cut cells is therefore estimated as

$$n_b \approx \frac{S(R)}{(l_d/2)^{D-1}} = D\frac{V(R)}{R}\left(\frac{2\,n_g}{\sqrt{D}}\right)^{D-1}. \tag{24}$$

The only unknown remaining in this equation is the radius of an D-ball. We know that the unit volume of $[0,1]^D$ is partitioned into N_s Voronoï cells. Therefore, on average, each Voronoï cell occupies a volume of $1/N_s$. Taking this volume equal to the volume of a sphere in Eq. (22) yields the estimate:

$$R = \frac{\sqrt[D]{\Gamma\left(\frac{D}{2}+1\right)}}{\sqrt[D]{N_s}\,\sqrt{\pi}} = \ell_{char}\frac{\sqrt[D]{\Gamma\left(\frac{D}{2}+1\right)}}{\sqrt{\pi}}, \tag{25}$$

which is in accordance with the intuition that the radius must be proportional to the characteristic length, ℓ_{char}. The *characteristic length* is a kind of typical distance between two closest points out of N_s uniformly distributed points in a unit hypercube of dimension D [56]:

$$\ell_{char} = \frac{1}{\sqrt[D]{N_s}}. \tag{26}$$

This characteristic distance is the exact distance of neighboring sites forming a regular orthogonal grid of N_s points in $[0,1]^D$.

Substitution of Eqs. (25) and (22) into Eq. (24) yields the final estimation of the number of boundary grid cells/pixels:

$$n_b \approx (n_g\cdot\ell_{char})^{D-1}\sqrt{\pi}\,\frac{D^{\left(\frac{3-D}{2}\right)}\cdot 2^{D-1}}{\sqrt[D]{\Gamma\left(\frac{D}{2}+1\right)}}. \tag{27}$$

Note the scaling expressed through the first parenthesis $(n_g\cdot\ell_{char})$: if the

characteristic length (the distance between neighboring points) gets multiplied by the same number as the pixel length, $a = 1/n_g$, the number of intersected cells remains identical.

Finally, the error in volume, ϵ_V, can be estimated by evaluating Eq. (19). The error in volume is a random variable (asymptotically Gaussian). Its mean value is zero because the mean value of the random variable V_b is zero:

$$\mu_{\epsilon_V} = 0. \tag{28}$$

The standard deviation of the error is obtained simply by multiplication of n_b and the standard deviation of V_b:

$$\sigma_{\epsilon_V} = n_b \cdot \sigma_b = \frac{1}{n_g} \cdot \ell_{char}^{D-1} \cdot M(D). \tag{29}$$

where the third term is a function depending only on the dimension of the problem:

$$M(D) = \frac{\sqrt{3\pi} \cdot D^{\left(\frac{3-D}{2}\right)} \cdot 2^{D-1}}{12 \cdot \sqrt{N_b} \cdot \sqrt[D]{\Gamma\left(\frac{D}{2}+1\right)}}. \tag{30}$$

Despite the fact that Eq. (29) was derived as a rough approximation based on several simplifying assumptions, the standard deviation of the error fits very well with simulated data, see the blue dashed lines in Fig. 4. If the standard deviation in Eq. (29) is multiplied by two, one obtains the error for Voronoï cells that have all edges aligned with the orthogonal grid (see Eq. 16). Such an extreme case is depicted by black dotted lines in Fig. 4.

A closer look at Eq. (29) reveals three important facts: (i) the error agrees with the intuition that it is inversely proportional to the number of cells per edge, n_g, independently of the dimension of the problem, (ii) an increase in the number of sites, N_s, decreases the characteristic length and therefore it decreases the absolute error in volumes, and (iii) for a given grid density and number of points, the *error decreases with increasing dimension, D*.

One might argue that a relative error in ith cell volume, defined as $\left|V^{(i)} - \hat{V}^{(i)}\right| / V^{(i)}$, is more suitable as the error is standardized by the actual volume. The error in this measure is easy to evaluate: the mean value remains zero and the standard deviation in Eq. (29) gets simply multiplied by N_s.

4.2. Distance errors

We now study the relative error in maximum radius for cell number i defined as:

$$\epsilon_r = \frac{|r_{max}^{(i)} - \hat{r}_{max}^{(i)}|}{r_{max}^{(i)}}, \quad i = 1, \ldots, N_s, \tag{31}$$

where $r_{max}^{(i)}$ is the exact solution obtained using QuickHull algorithm (either clipped or periodic Voronoï tessellation) and $\hat{r}_{max}^{(i)}$ is the approximated solution obtained with a certain rasterization. The absolute error is standardized by the exact value to get an idea about the relative error.

Numerical simulations presented in Fig. 5 show the errors of estimated radii of empty hyper-spheres. The thickness of the lines increases with the exact maximum radius, i.e. the most relevant line to obtain estimation of ϕ_{mM}, recall Eq. (11), is the thickest line. The top row of Fig. 5 studies the decrease in error for a sample of $N_s = 16$ sites in dimensions $D = 2$, 3 and 4. Similarly, Fig. 5 bottom row studies the errors for $N_s = 50$ sites. It can be seen that for an increasing number of pixels, n_g along each edge, the error quickly decreases and so does the scatter in the errors.

There are two lines in Fig. 5 that represent plot of two formulas for the error. A notable error in radius corresponds to the grid cell diagonal; it is the maximum error that can occur for regular-shaped Voronoï cell:

$$\frac{l_d}{\ell_{char}} = \frac{1}{n_g} \sqrt{D} \cdot \sqrt[D]{N_s}. \tag{32}$$

The numerator is the "diagonal error" in resolution, i.e. the deviation between two pixels measured along the spatial diagonal, see the illustration in Fig. 6 left for the meaning in 2D. Eq. (32) does not represent the maximum possible error in radius, because the error is theoretically unbounded: Fig. 6 right depicts the situation with an acute angle between the two edges where $l_e > l_d$. There the cell diagonal error can be exceeded. The exceedance probability grows with the number of sites in

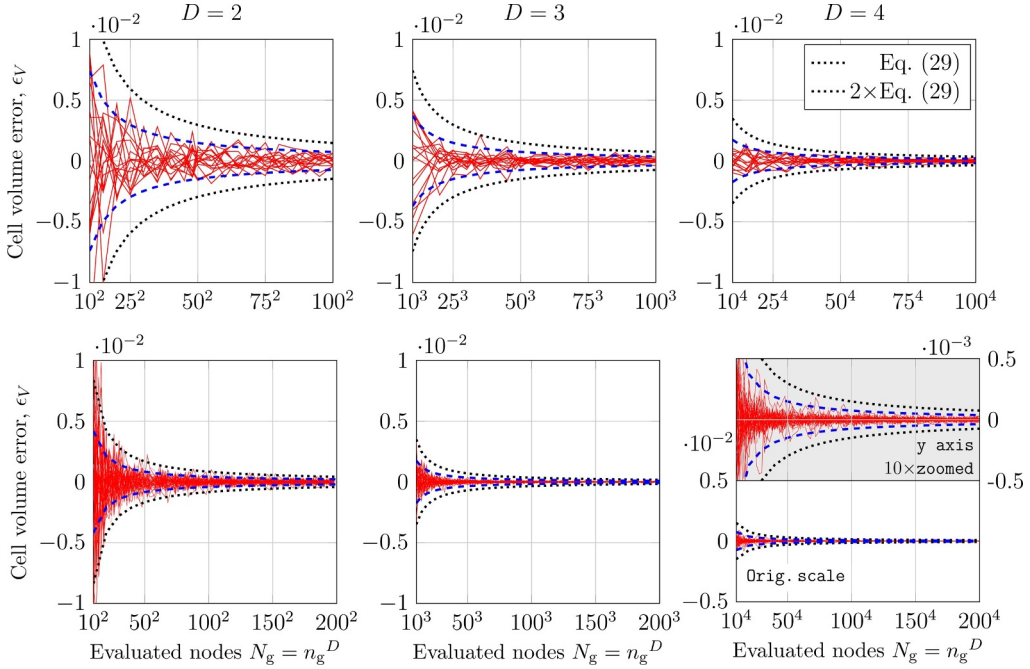

Fig. 4. Convergence of volumes of Voronoï cells to the exact solution for $D = 2$, 3 and 4 dimensions. Evaluated for a single LH sample with $N_s = 16$ (top row) and $N_s = 50$ (bottom row).

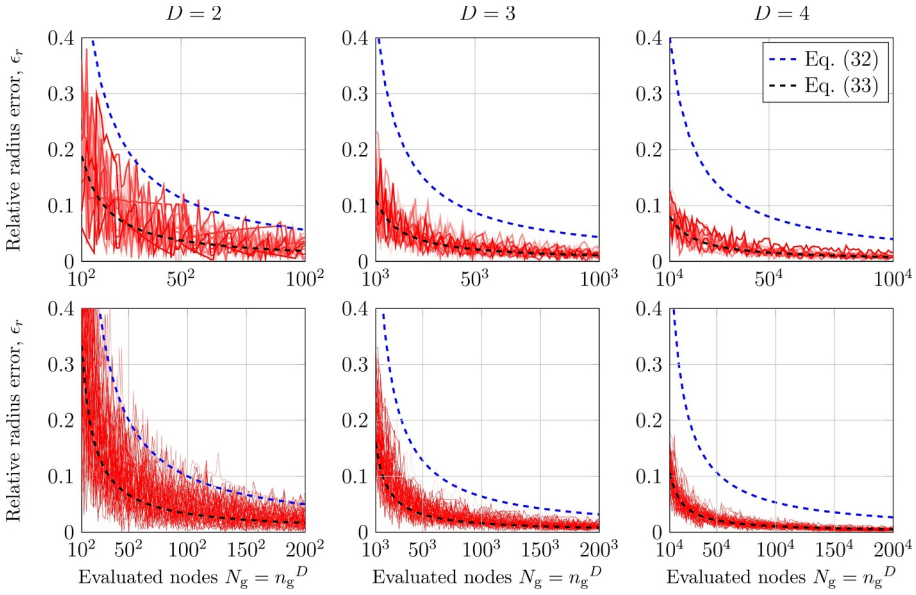

Fig. 5. Convergence of maxRads of Voronoï cells to the exact solution for $D = 2$, 3 and 4 dimensions. Evaluated for a single LH sample with $N_s = 16$ (top row) and $N_s = 50$ (bottom row).

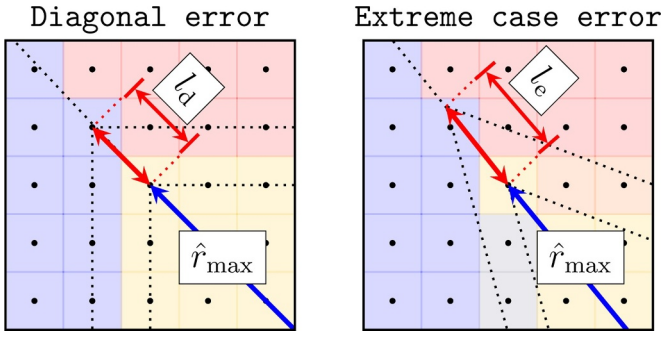

Fig. 6. Illustration of the errors in radius approximation when rasterizing 2D Voronoï diagram (two possible extreme mutual positions of the grid with respect to the boundaries of Voronoï cells denoted by the dotted lines).

a domain, see the peaks over the blue dashed line (Eq. 32) in Fig. 5 bottom left.

The average relative error in radii, depicted by the black dashed lines, has been found to read:

$$\mu_{\epsilon_r} = \frac{l_d/(D+1)}{\ell_{\text{char}}} = \frac{1}{n_g}\frac{\sqrt{D}}{D+1}\cdot\sqrt[D]{N_s}\,. \tag{33}$$

The normalizing denominator in both Eqs. (32) and (33) is the *characteristic length*.

The comparison between the numerically obtained relative errors and the suggested formulas in Fig. 5 show that the average error for a given grid density (n_g) does not depend strongly on the dimension, D. Indeed, for small to moderate dimensions (1 to 20), the product $\sqrt{D}\sqrt[D]{N_s}$ does not vary considerably for reasonable pairs N_s and D, while the numerically obtained errors seem to be well captured by Eqs. (32) and (33).

By comparing the errors in volumes, ϵ_V, with the errors in radii, ϵ_r, it becomes clear that for the same discretization density, the approximation of volumes is more accurate. The reason is that ϵ_V is a kind of *average* property while ϵ_r is related to the *extremes*. Therefore, to estimate ϵ_r with a desired accuracy necessitates a finer discretization grid. Similarly, one can also estimate the rasterization error in the estimated centroid for a cell number i as the Euclidean distance between the exact centroid and the approximated centroid: $\epsilon_g = \|\text{cg}^{(i)} - \hat{\text{cg}}^{(i)}\|$. The nature of the error is similar to the volume error: the only source is in the boundary cells that might under-represent or over-represent the volumes of grid cells intersected by the true Voronoï cell boundary. Again, it is a kind of average property and the errors are smaller than ϵ_r.

In order to measure the speedup of the proposed parallel solution with respect to the exact solution of Voronoï diagram, a certain fineness of the rasterization grid must be selected. Eq. (12) may suggest that the maximum absolute error in distance increases with dimension and to alleviate this dependency on the dimension, one should select the number of pixels per edge as $n_g = cN_s\sqrt{D}$, where c is a constant independent of the dimension. However, for the sites selected from the class of LH designs, the error analysis suggests that it is sufficient to use $n_g = cN_s$. Based on the knowledge of the boundaries on maximum and usual approximation errors, the user is provided with the option of adjusting the mesh density accordingly. We suggest to take the number of segments along each edge:

$$n_g = 3N_s, \tag{34}$$

as sufficient with respect to the errors. The implementation of the parallel solution and the associated speedup is discussed in what follows.

5. Implementation of parallel solution

The actual parallel implementation consists of several steps for which the approach to parallel execution differs to some extent. Generally, however, solution of all tasks described in Section 3 does exhibit a rather insignificant arithmetic workload. The dominant scale of the problem is clearly the grid of nodes. The number of nodes in grid rises steeply with (i) the side of grid, n_g, that is considered as $n_g = 3N_s$, and more so with (ii) the dimension of the domain, D.

With the size of grid being $N_g = (3N_s)^D$ nodes, the description of the grid consists of DN_g coordinates. It is surely not convenient to store these coordinates in global memory and load them each time these are of interest. Instead, coordinates of grid nodes are generated on-the-fly in GPU registers based on the knowledge of the index of each desired grid node. This approach brings several advantages:

- does save the global memory from storing a large array that may grow beyond control. Also, allocating such a large array would take considerable time at the beginning of the solution,

- because this major amount of data is never loaded from global memory, the L2 cache hit rate is increased during all steps of solution,
- does not overly stress GPU registers for there is hardly any register pressure induced by arithmetic instructions.

Nevertheless, the grid of nodes still remains to dictate the scale of the problem. Because its extent is substantially larger than the number of sites and, moreover, grid nodes are also independent entities, it is desirable to select the grid to be the parallelized dimension of the problem wherever possible.

In the numerical studies later presented, the required number of threads reaches up to about 10^{14} threads (=grid nodes). A question might arise about whether the hardware itself is capable of executing such a number of threads. The currently used Nvidia GPU, GTX 1080Ti, is theoretically able to execute a maximum number of $(2^{31} \times 2^{16} \times 2^{16}) \approx 10^{18}$ thread blocks, each of which may contain up to 1024 threads. The theoretical maximum number of executed threads then may reach up to about 10^{21} threads. This theoretical value of threads executed in maximum efficiency is typically decreased accordingly by requirements on register resources induced by each specific kernel function. Once the kernel requirements on register resources exceed the hardware capabilities, the on-chip L1/L2 caches start to be utilized. Ultimately, the local memory of the GPU would be used for storing of the register data. During the presented simulations, however, no such limitations have been encountered. Following is the discussion of implementation of individual solution steps as motivated in Section 3.

5.1. Comparison of distances between nodes and sites

The first task of the solution is also the most extensive. Initially, it is always required to obtain the index of the closest site to each of N_g grid nodes, recall Fig. 1. The actual implementation approach proposes to understand the nodes of the mesh as *target points* which interact with the *source points* which are here represented by the N_s sites of the approximated Voronoï diagram. As expected, the parallelized dimension here are the grid nodes, i.e. each node is serviced by its own thread. In total, the number of active threads is equal to the number of grid nodes, N_g.

Coordinates of the grid nodes are held in registers, being generated according to the node index (thread global index). Then, tiles of the source points are consecutively loaded into shared memory and a simple task of comparing mutual distances of target and source points (nodes and sites) is executed. For the sake of distance comparison, it is not necessary to evaluate the square root to get the actual Euclidean distance. It is only sufficient to sum the squares of projections of the distance along all dimensions as the square root is a monotonous operation.

All threads (=nodes) compare their distance to the same site at each time. This ensures that all threads access the same shared memory bank simultaneously, thus avoiding bank conflicts for any D. For each target point, index of its closest source point is stored. The result of such an algorithm is an array `gridClstIds` of N_g unsigned short integer (2 byte) indices of the closest site to each grid node.

5.2. Summation of cell volumes

Right after the indices of closest sites to grid nodes are known, it is possible to compute the approximated volumes of Voronoï cells as a simple sum of volume of grid cells belonging to each site according to Eq. (6). It is worth noting that the volumes of grid cells belonging to each Voronoï cell are scattered all across the `gridClstIdxs` array in global memory. Summing these by a parallel reduction for each Voronoï cell is therefore not feasible at this point. Instead, an inverse implementation approach to the preceding step is applied.

The N_s sites are here the parallelized dimension (*target points*), serviced by N_s active threads. Threads represent the sites that correspond with their global thread index. Tiles of N_g indices of closest sites from `gridClstIds` (*source points*) are consecutively loaded into shared memory. The threads proceed through the loaded closest indices and check whether each particular index of the closest site is equal to any of global thread indices in block. If so, the respective thread adds one to the integer count of grid cells that belong to its respective Voronoï cell.

Again, all threads access the same `gridClstIds` element at each time, thus avoiding shared memory bank conflicts. The result of the described procedure is an array `vorCellVols` of N_s integers that represent the number of grid cells belonging to each Voronoï cell. The resulting approximated Voronoï cell volume is then simply obtained by multiplication of elements of this array by the constant grid cell volume, $v = (1/n_g)^D$, recall Eq. (6).

5.3. Averaging coordinates of nodes belonging to each Voronoï cell

During the single parallelized pass through tiles of `gridClstIds` described above, it is also possible to compute the approximated Voronoï cell centroids right along. Each site (=thread) maintains D float coordinates of its centroid in shared memory.

In the intersite (non-periodic) variant, each time a thread recognizes its respective grid cell, apart from adding one to its count of own grid cells, it generates the coordinates of the grid node and adds these to cell centroid coordinates in shared memory. At the very end of the routine, centroid coordinates are merely divided by element in the `vorCellVols` array respective to each site. This way, the averaging in Eq. (7) is done.

If periodic boundary conditions are considered, Eqs. (8) and (9) must be used instead of a plain average. However, as the parallel execution is constructed so far, this does not represent a significant drawback. If a thread recognizes its respective grid cell in a tile of `gridClstIds`, it ensures that the closest image of the generated grid node to the provisional centroid is considered in each dimension according to Eq. (8). After that, the recurrence formula (9) for provisional average is employed. The number j of the jth iteration step is simply represented by the current integer count of respective grid cells in the provisional integer Voronoï cell volume held in registers.

5.4. Keeping track of the furthermost node discovered

The last of the solution steps is the search for the furthermost grid node to each site. As described in Section 3.3, such a most distant grid node will be then considered as the center of the largest empty Delaunay hyper-circle of each site. Such a grid node can also be identified during the single pass through the tiles of the `gridClstIds` array.

When a thread (=site) recognizes its respective grid cell, the generated coordinates of its node are used not only for computation of the Voronoï cell centroid but the distance to the site is also computed. If this distance is greater than the provisional maximum distance held in registers, the node becomes the new provisional furthermost node. Again, only the sum the squares of projections of the distance along all dimensions is computed.

The output of this routine is an array `maxRadIds` of integer indices of the furthermost node of each site. The actual distances to these nodes (maximum radii) are computed afterwards in parallel.

6. Performance and speedup

The execution performance of the presented parallel approximation is compared to the standard QuickHull algorithm [4]. The execution times are compared in Fig. 7 left. The solution speedup attained by the parallel solution is shown in Fig. 7 right.

The results are displayed in dependence on the number of sites, N_s, and dimensions, D. In case of the parallel approximation, the side of

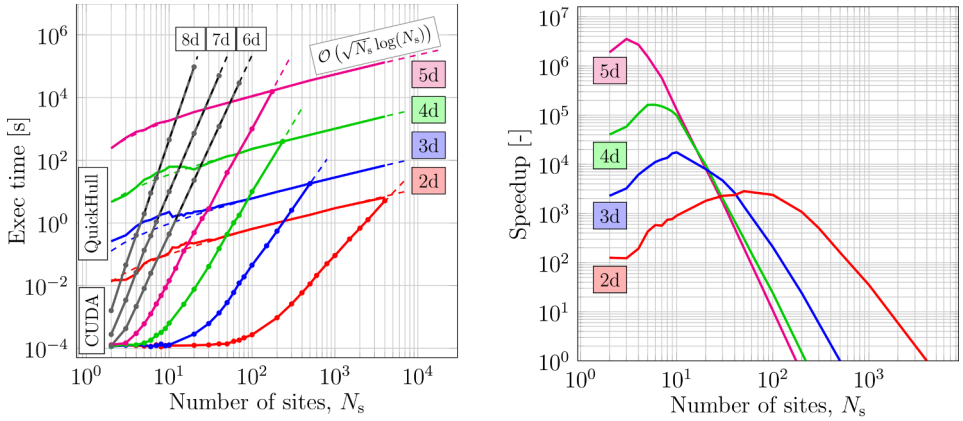

Fig. 7. Left: Comparison of execution time of the QuickHull algorithm and the presented parallel approximation ($N_g = (3N_s)^D$). Right: solution speedup attained by the parallel approximation.

grid is set as $n_g = 3N_s$. Therefore the grid contains $N_g = (3N_s)^D$ cells.

In an average case, the complexity of the QuickHull algorithm is known to be $O(N_s \log(N_s))$. In its worst case, QuickHull approaches $O(N_s^2)$. In this contribution, LH point samples are exclusively considered. Therefore, the attained complexity of QuickHull solution is lower than in an average case, reaching $O(\sqrt{N_s}\log(N_s))$ for LHS random samples (see Fig. 7 left).

As far as the presented parallel solution is considered, its complexity is mainly guided by the grid size, $N_g = (3N_s)^D$, which depends on the number of sites, N_s, as well as the dimension D. The observed asymptotic complexity of the parallel solution is about $O(N_s^D)$, see the "fan" of steep straight lines in Fig. 7 left. The curves obtained with the proposed rasterization algorithm eventually intersect the corresponding curve obtained with the QuickHull algorithm. From that number of sites on, it becomes cheaper to switch to QuickHull algorithm. Note, however, that the performance of QuickHull depends on the type of the sample. In the case of studied LH designs, the QuickHull algorithm performs above standard. The performance of the proposed parallel algorithm depends on the required precision. For a random sample, it might be necessary to use a finer discretization based on the shortest distance between sites. In this work, the shortest mutual distance between sites is controlled by assuming a certain regularity of the sample.

In the present case of LH samples, the peak solution speedup reaches between 3×10^3 to about 3×10^6 depending on dimension, D, and begins to fade down as the execution time of parallel solution approaches QuickHull curves. Nevertheless, for a solid range of practical scenarios (frequent solution for small point samples up to moderate dimension), the presented approximation algorithm does provide a significant solution speedup. This solution speedup often becomes decisive, especially when a repetitive execution of the solution in thousands or millions of steps of sample optimization is needed (such as in the combinatorial optimization algorithms [66]).

We remark that the algorithm still allows for a subsequent refinement to achieve sub-pixel resolution, if needed. As the rasterization errors stem only from the grid cells intersected by the boundaries of Voronoï cells, the refinement may be limited to the boundary grid cells, i.e. cells that are neighbored by at least one cell belonging to another Voronoï region. The total number of boundary cells, roughly $N_s \cdot n_b/2$ (see Eq. 27), is considerably lower than the total number of the initial grid cells, N_g.

7. Conclusion

The presented paper introduced a parallelized implementation of a numerical approximation of selected scalar properties of a hyper-dimensional Voronoï diagram. The developed approximation algorithm is motivated by the field of interest of the authors that is optimization of uniformity of point samples for Design of Experiments.

The notion of parallel processing of an underlying raster of nodes crucially relies on the use case of the algorithm. In the case of Latin Hypercube (LH) point samples, one can estimate a lower bound on mutual point distance. In this paper, a mild assumption regarding the regularity of point distribution is exploited for prediction of errors of approximated scalar descriptors of Voronoï diagram. The scalar properties approximated by the current algorithm are the Voronoï cell volumes, coordinates of Voronoï cell centroids and the maximum radius of Delaunay hyper-circle associated with each Voronoï cell. These are the values of interest when conducting sample optimization using the miniMax optimization criterion [33]. Nevertheless, there are various other characteristics that might be approximated, such as higher moments of Voronoï cells (moments of inertia and the principal moments) that can be used for description of shape and directional distortion of Voronoï cells.

A massively parallel implementation employing the Nvidia CUDA platform is presented, aiming for an efficient processing of such a grid of nodes that grows fast with its dimension. This disadvantage disqualifies the presented algorithm to be a universally utilized approach.

This paper was motivated by the field of interest of the authors that is optimization of uniformity of small sample point designs in low to moderate dimension. The criteria of such an optimization utilize scalar descriptors of Voronoï cells. In the optimization algorithms used, these scalar descriptors have to be evaluated for many trial point configurations (millions of steps or more). In these applications, the proposed approach yields great savings in execution times. Should the number of sites exceed the range of observed solution speedup, switching to e.g. the QuickHull solution is advisable.

However, for a convenient range of the number of sites, N_s, and dimensions, D, the parallel approximation offers a significant speedup of solution. When employed as a part of a kind of a repetitive algorithm such as in statistical optimization, Centroidal Voronoï tesselation or weighted Monte Carlo estimation, the benefit of using such a fast approximation becomes even more pronounced.

Acknowledgement

This work has been supported by the Czech Science Foundation under project No. GA16-22230S. Additionally the work has also been supported by Specific university research project MSMT No. FAST-J-18-5254 (The Ministry of Education, Youth and Sports of Czech Republic). The authors thank Dr. Václav Sadílek for performing the analysis of the average number of facets of Poisson–Voronoï tessellation.

References

[1] Asano T, Mulzer W, Rote G, Wang Y. Constant-work-space algorithms for geometric problems. J Comput Geom 2011;2. https://doi.org/10.20382/jocg.v2i1a4.

[2] Aurenhammer F. Voronoi diagrams—a survey of a fundamental geometric data structure. ACM Comput Surv (CSUR) 1991;23(3):345–405. https://doi.org/10.1145/116873.116880.

[3] Avellaneda M, Buff R, Friedman C, Grandechamp N, Kruk L, Newman J. Weighted monte carlo: a new technique for calibrating asset-pricing models. Int J Theor Appl Finance 2001;4(01):91–119.

[4] Barber CB, Dobkin DP, Huhdanpaa H. The quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software (TOMS) 1996;22(4):469–83.

[5] Bates S, Sienz J, Langley D. Formulation of the Audze–Eglais uniform Latin Hypercube design of experiments. Adv Eng Softw 2003;34(8):493–506. https://doi.org/10.1016/S0965-9978(03)00042-5.

[6] Bellhouse DR. Area estimation by point-counting techniques. Biometrics 1981;37. https://doi.org/10.2307/2530419.

[7] Bregt AK, Denneboom J, Gesink HJ, van Randen Y. Determination of rasterizing error: a case study with the soil map of the netherlands. Geogr Inf Syst 1991;5. https://doi.org/10.1080/02693799108927861.

[8] Brostow W, Chybicki M, Laskowski R, Rybicki J. Voronoi polyhedra and delaunay simplexes in the structural analysis of molecular-dynamics-simulated materials. Phys Rev B 1998;57(21):13448–58. https://doi.org/10.1103/PhysRevB.57.13448.

[9] Buchta C, Müller J, Tichy RF. Stochastical approximation of convex bodies. Mathematische Annalen 1985;271. https://doi.org/10.1007/bf01455988.

[10] Büeler B, Enge A, Fukuda K. Exact volume computation for polytopes: a practical study. Polytopes–combinatorics and computation. Springer; 2000. p. 131–54.

[11] Bulik M, Liefvendahl M, Stocki R, Wauquiez C. Stochastic simulation for crash-worthiness. Adv Eng Softw 2004;35(12):791–803.

[12] Burrough P. Principles of geographical information systems for land resources assessment. Monographs on Soil and Resources Survey (Book 12). Oxford University Press978-0198545637; 1986.

[13] Carver CF, Brunsdon SJ. Vector to raster conversion error and feature complexity: an empirical study using simulated data. Geog Inf Syst 1994;8. https://doi.org/10.1080/02693799408901999.

[14] Chan TM. Semi-online maintenance of geometric optima and measures. SIAM J Comput 2003;32. https://doi.org/10.1137/s0097539702404389.

[15] Yang Liao, Biao Qin, ChengHu Zhou, Yang Ou. An equal area conversion model for rasterization of vector polygons. SCIENCE CHINA Earth Sci 2007;50. https://doi.org/10.1007/s11430-007-5013-6.

[16] Chew LP, Dyrsdale RL. Voronoi diagrams based on convex distance functions. Proceedings of the first annual symposium on Computational geometry - SCG '85, held in Baltimore, Maryland, USA. 19850897911636. https://doi.org/10.1145/323233.323264.

[17] Fabri A. Dehne C, Kenyon F. Scalable and architecture independent parallel geometric algorithms with high probability optimal time. Proceedings of 1994 6th IEEE Symposium on Parallel and Distributed Processing, held in Dallas, TX, USA (26–29 Oct. 1994). 19940-8186-6427-4. https://doi.org/10.1109/spdp.1994.346119.

[18] Deng Z, Guo Z. Interval identification of structural parameters using interval overlap ratio and monte carlo simulation. Adv Eng Softw 2018;121:120–30. https://doi.org/10.1016/j.advengsoft.2018.04.006.

[19] Dong H, Li C, Song B, Wang P. Multi-surrogate-based differential evolution with multi-start exploration (mdeme) for computationally expensive optimization. Adv Eng Softw 2018;123:62–76. https://doi.org/10.1016/j.advengsoft.2018.06.001.

[20] Vance Faber, Max Gunzburger, Qiang Du. Centroidal Voronoi tessellations: Applications and algorithms. SIAM Rev 1999;41. https://doi.org/10.1137/S0036144599352836.

[21] Eliáš J, Vořechovský M. Modification of the Audze–Eglājs criterion to achieve a uniform distribution of sampling points. Adv Eng Softw 2016;100:82–96.

[22] Filipe L, Vieira M, Augusto M, Vieira M, Ruiz LB, Alfredo A, et al. Efficient incremental sensor network deployment algorithm. In Brazilian Symposium on Computer Networks. 2004.

[23] Frolov YS, Maling DH. The accuracy of area measurement by point counting techniques. Cartograph J 1969;6. https://doi.org/10.1179/caj.1969.6.1.21.

[24] Fuerle F, Sienz J. Formulation of the Audze–Eglais uniform latin hypercube design of experiments for constrained design spaces. Adv Eng Softw 2011;42(9):680–9.

[25] Fuerle F, Sienz J. Decomposed surrogate based optimization of carbon-fiber bicycle frames using optimum latin hypercubes for constrained design spaces. Comput Struct 2013;119:48–59. https://doi.org/10.1016/j.compstruc.2012.11.014.

[26] Galashev AE, Rakhmanova OR, Chukanov VN. Computer simulation of the absorption of CO_2 molecules by water cluster: 2. the microstructure. Colloid J 2005;67. https://doi.org/10.1007/s10595-005-0093-5.

[27] Goodchild MF, Moy WS. Estimation from grid data: the map as a stochastic process. In: Tomlinson RF, editor. Proceedings of the Commission on Geographical Data Sensing and Processing, Moscow, 1976. Ottawa: International Geographical Union, Commission on Geographical Data Sensing and Processing; 1977.

[28] Goodchild MF. Fractals and the accuracy of geographical measures. Math Geosci 1980;12. https://doi.org/10.1007/bf01035241.

[29] Graf W, Götz M, Kaliske M. Computing permissible design spaces under consideration of functional responses. Adv Eng Softw 2018;117:95–106. https://doi.org/10.1016/j.advengsoft.2017.05.015. Special Section - CIVIL-COMP 2017

[30] Hilhorst HJ, Lazar EA. Many-faced cells and many-edged faces in 3D poisson–voronoi tessellations. J Stat Mech: Theory Exp 2014;2014(10):1–22. https://doi.org/10.1088/1742-5468/2014/10/P10021.

[31] Yu AB, Xu JQ, Zou RP. Analysis of the packing structure of wet spheres by voronoi–delaunay tessellation. Granular Matter 2007;9. https://doi.org/10.1007/s10035-007-0052-4.

[32] Jeong CS, Lee DT. Parallel geometric algorithms on a mesh-connected computer. Algorithmica 1990;5. https://doi.org/10.1007/bf01840383.

[33] Johnson M, Moore L, Ylvisaker D. Minimax and maximin distance designs. J Stat Plan Inference 1990;2(26):131–48. https://doi.org/10.1016/0378-3758(90)90122-B.

[34] Johnson SB. The ghost map: the story of London's most terrifying epidemic - and how it changed science, cities, and the modern world. Riverhead1429501316; 2007.

[35] Kasim MF, Ceurvorst L, Ratan N, Sadler J, Chen N, Sävert A, et al. Quantitative shadowgraphy and proton radiography for large intensity modulations. Phys Rev E 2017;95:023306. https://doi.org/10.1103/PhysRevE.95.023306.

[36] Kim H, Kim S, Kim T, Lee TH, Ryu N, Kwon K, et al. Efficient design optimization of complex system through an integrated interface using symbolic computation. Advances in Engineering Software 2018;126:34—45. https://doi.org/10.1016/j.advengsoft.2018.09.006.

[37] Kumar S, Kurtz SK. Properties of a two-dimensional Poisson–Voronoi tesselation: A monte-carlo study. Mater Charact 1993;31(1):55–68. https://doi.org/10.1016/1044-5803(93)90045-W.

[38] Lawrence RL, Ripple WJ. Determining patch perimeters in raster image processing and geographic information systems. Int J Remote Sens 1996;17. https://doi.org/10.1080/01431169608949084.

[39] Lee J-S, Cho T-S, Lee J, Jang M-K, Jang T-K, Nam D, et al. A stochastic search approach for the multidimensional largest empty sphere problem. 2004https://pdfs.semanticscholar.org/9a47/72553eee0eff6c6e44e40a663be98129a6e0.pdf.

[40] Lehký D, Slowik O, Novák D. Reliability-based design: artificial neural networks and double-loop reliability-based optimization approaches. Adv Eng Softw 2018;117:123–35. https://doi.org/10.1016/j.advengsoft.2017.06.013. Special Section - CIVIL-COMP 2017

[41] Lloyd PR. Quantisation error in area measurement. Cartogr J 1976;13. https://doi.org/10.1179/caj.1976.13.1.22.

[42] Jan-Ulrich Kreft Wolfgang Alt Martin Bock Amit Kumar Tyagi. Generalized Voronoi tessellation as a model of two-dimensional cell tissue dynamics. Bull Math Biol 2010;72. https://doi.org/10.1007/s11538-009-9498-3.

[43] Mašek J, Vořechovský M. Parallel implementation of hyper-dimensional dynamical particle system on CUDA. Adv Eng Softw 2018;125:178–87. https://doi.org/10.1016/j.advengsoft.2018.03.009.

[44] Meijering JL. Interface area, edge length, and number of vertices in crystal aggregates with random nucleation. Philips Research Reports 1953;8:270–90.

[45] Miller R, Stout QF. Geometric algorithms for digitized pictures on a mesh-connected computer. IEEE Trans Pattern Anal Mach Intell 1985;PAMI-7. https://doi.org/10.1109/tpami.1985.4767645.

[46] Mitchell TM. Machine learning. McGraw-Hill series in computer science. 1 McGraw-Hill978-0070428072; 1997.

[47] Montoro JCG, Abascal JLF. The Voronoi polyhedra as tools for structure determination in simple disordered systems. J Phys Chem 1993;97(16):4211–5. https://doi.org/10.1021/j100118a044.

[48] Myšáková E, Lepš M. Evaluation of minimax criterion in constrained design domains. ESCO 2014: European Seminar on Computing, Pilsen, Czech Republic. 2014. p. 132http://www.esco2014.femhub.com/docs.

[49] Müller J. Map gridding and cartographic errors: a recurrent argument. Can Cartographer 1977;14:152–67.

[50] Novák D, Vořechovský M, Teplý B. FReEt: software for the statistical and reliability analysis of engineering problems and FReET-D: degradation module. Adv Eng Softw 2014;72:179–92. https://doi.org/10.1016/j.advengsoft.2013.06.011.

[51] David K. others. NVIDIA CUDA software and GPU parallel computing architecture. ISMM 2007;7:103–4.

[52] Pronzato L. Minimax and maximin space-filling designs: some properties and methods for construction. Journal de la Société Française de Statistique 2017;158(1):7–36.

[53] Pronzato L, Müller WG. Design of computer experiments: space filling and beyond. Stat Comput 2012;22(3):681–701. https://doi.org/10.1007/s11222-011-9242-3.

[54] Reitzner M. Stochastical approximation of smooth convex bodies. Mathematika 2004;51. https://doi.org/10.1112/s0025579300015473.

[55] Romero VJ, Burkardt JV, Gunzburger MD, Peterson JS. Comparison of pure and 'Latinized" centroidal voronoi tessellation against various other statistical sampling methods. Reliability Eng Syst Safety 2006;91(10–11):1266–80. https://doi.org/10.1016/j.ress.2005.11.023.

[56] Sadílek V, Vořechovský M. Evaluation of pairwise distances among points forming a regular orthogonal grid in a hypercube. J Civil Eng Manage 2018;24(5):410–523. https://doi.org/10.3846/jcem.2018.5189.

[57] Sánchez-Gutiérrez D, Tozluoglu M, Barry JD, Pascual A, Mao Y, Escudero LM. Fundamental physical cellular constraints drive self-organization of tissues. EMBO J 2016;35(1):77–88. https://doi.org/10.15252/embj.201592374.

[58] Sega M, Jedlovszky P, Medvedev N, Vallauri R. Free volume properties of a linear soft polymer: a computer simulation study. J Chem Phys 2004;121. https://doi.org/10.1063/1.1763840.

[59] Sen Z. Spatial modeling principles in earth sciences. Springer International Publishing978-3-319-41758-5; 2016. https://doi.org/10.1007/978-3-319-41758-5.

[60] Shortridge AM. Geometric variability of raster cell class assignment. Int J Geograph Inf Sci 2004;18. https://doi.org/10.1080/13658810410001702012.

[61] Shortridge Goodchild Michael F, Ashton M. Geometric probability and gis: some applications for the statistics of intersections. International Journal of Geographical Information Science 2002;16. https://doi.org/10.1080/13658810110099099.

[62] Springel V. E pur si muove: Galilean-invariant cosmological hydrodynamical

simulations on a moving mesh. Mon Not R Astron Soc 2010;401(2):791–851. https://doi.org/10.1111/j.1365-2966.2009.15715.x.
[63] Banavar Jayanth R, Sharma MG, Kumar Susmit, Kurtz Stewart K. Properties of a three-dimensional Poisson–Voronoi tesselation: a monte carlo study. J Stat Phys 1992;67. https://doi.org/10.1007/bf01049719.
[64] Takada A. Voronoi tessellation analysis of SiO_2 systems based on oxygen packing. J Non Cryst Solids 2018;499. https://doi.org/10.1016/j.jnoncrysol.2018.07.037.
[65] Toussaint GT. Computing largest empty circles with location constraints. Int J Parallel Program 1983;12. https://doi.org/10.1007/bf01008046.
[66] Vořechovský M, Novák D. Correlation control in small sample Monte Carlo type simulations i: a simulated annealing approach. Probab Eng Mech 2009;24(3):452–62. https://doi.org/10.1016/j.probengmech.2009.01.004.
[67] Voronoï G. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les parallélloèdres primitifs. Journal für die reine und angewandte Mathematik 1908;134:198–287.
[68] Vořechovský M, Sadílek V, Eliáš J. Application of Voronoï weights in Monte Carlo integration with a given sampling plan. In: Freitag S, Muhanna RL, Mullen RL, editors. Proceedings of the 7th International Workshop on Reliable Engineering Computing (REC), Bochum, Germany. 2016. p. 441–52http://rec2016.rub.de/papers.html.
[69] Vořechovský M, Sadílek V, Eliáš J. Voronoï weighting of samples in Monte Carlo integration. Proceedings of 2nd International Conference on Uncertainty Quantification in Computational Sciences and Engineering. 2017.
[70] Wolfram S. A new kind of science. 1 Wolfram Media1-57955-008-8; 2002.
[71] Yakowitz S, Krimmel J, Szidarovszky F. Weighted Monte Carlo integration. SIAM J Numer Anal 1978;15(6):1289–300.
[72] Yan D-M, Wang K, Lévy B, Alonso L. Computing 2d periodic centroidal Voronoï tessellation. Voronoï Diagrams in Science and Engineering (ISVD), 2011 Eighth International Symposium on. IEEE; 2011. p. 177–84.
[73] Yan D-M, Wang W, LéVy B, Liu Y. Efficient computation of clipped voronoï diagram for mesh generation. Comput-Aided Des 2013;45(4):843–52.