Graph-based Computation of Voronoi Diagrams on Large-scale Point-based Surfaces

A. Bletterer[†1], F. Payan[1] M. Antonini[1]

[1]Université Côte d'Azur, CNRS, I3S, France

Abstract

We present an original algorithm to construct Voronoi tessellations on surfaces from a set of depth maps. Based on a local graph-based structure, where each local graph spans one depth map, our algorithm is able to compute partial Voronoi diagrams (one per scan), and then to merge/update them into a single and globally consistent Voronoi diagram. Our first results show that this algorithm is particularly promising for improving the sampling quality of massive point clouds or for reconstructing very large-scale scenes, with low and manageable memory consumption.

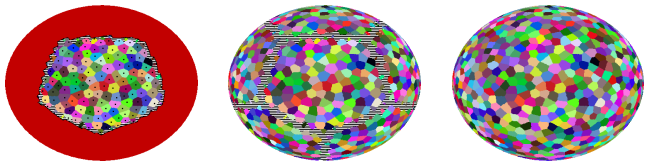
CCS Concepts

• ***Computing methodologies*** → ***Point-based models; Parametric curve and surface models;***

1. Introduction

3D point clouds produced during building or large site scanning campaigns can contain billions of points. To obtain them, dozens of acquisitions are registered and merged. Processing and manipulation of surfaces modeled by these point clouds (called *underlying surfaces* from now on) are tricky. One reason is the lack of structure, which complicates any processing requiring a local neighborhood on the sampled surfaces. A second reason is the volume of data which can be prohibitive, and thus requires on-the-fly processing and/or out-of-core implementations. Recently, a data structure based on interconnected local graphs was proposed [BPA20] for processing point clouds obtained by aggregation of multiple acquisitions. By taking advantage of depth map connectivity, of local processings and of an out-of-core implementation, this structure handles massive 3D point clouds on any computer, regardless of its RAM capacity. We present a novel algorithm based on this structure, which constructs Voronoi diagrams on the underlying surface of point clouds. The overall principle (Figure 1) is to construct one partial Voronoi diagram per local graph, and then to merge/update them into a single and globally coherent Voronoi diagram, with a special attention on overlapping regions. By adding a relaxation stage, centroidal Voronoi tessellations can also be provided. Many applications can benefit from this algorithm. In the experimental section, we show for example that the sampling quality of 3D scenes containing hundreds of millions of points can be improved with a low and controllable memory consumption, but also to obtain high quality meshes of scanned surfaces.

† This work was supported by a grant from Région Sud (France).



(a) Partial Voronoi diagram of one graph. (b) Merging of all the partial diagrams. (c) Updating in the common zones.

Figure 1: *Local graph-based computation of a global Voronoi diagram on a point-based surface.*

2. Basics of Centroidal Voronoi diagrams

Let us define the Voronoi tessellation (VT) of a metric space (M,d) as the partition of the space M constructed from a set of sites $S \subseteq M$, such that each Voronoi cell (VC) C_i contains the set of points $x \in M$ closer to s_i than to any other site: $C_i = \{x \in M | d(x,s_i) < d(x,s_j), \forall s_j \in S, s_j \neq s_i\}$. A VT is called *Centroidal* (CVT) if each site s_i corresponds to the centroid of its cell. One simple method to obtain a CVT is called *Lloyd relaxation* [Llo82]. Starting from an initial set of sites, two steps are alternated until convergence: i) the VT is computed; ii) each site is moved to the centroid of its cell generated at the previous step.

3. Structuration of underlying surfaces with local graphs

Figure 2 gives an overview of the workflow proposed in [BPA20] for structuring an underlying surface sampled by a point cloud obtained from N depth maps. For each depth map D_j (D_j, $j \in \{0,1,..,N\}$) representing a specific point of view (Figure 2 (a)), a local undirect graph $G_j = (V_j, E_j)$ is constructed (Figure 2 (b)). V_j is a set of vertices, where each vertex is associated to one 3D point

DOI: 10.2312/egs.20221030

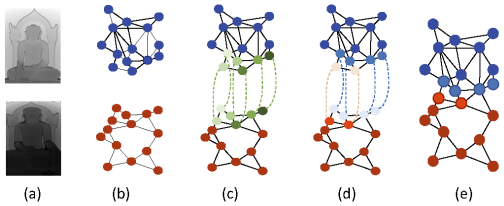


Figure 2: *Workflow to structure aggregated point clouds with interconnected local graphs [BPA20].*

of the cloud, and E_j a set of edges linking the vertices, according to the depth map connectivity. By segmentation, some edges are removed in each graph to identify distinct elements in the scanned surfaces. The local graphs are then interconnected (Figure 2 (c)) by matching the vertices present in overlapping regions (green dots). To avoid redundant operations in these regions, each redundant point is finally assigned to one graph only (Figure 2 (d)). After that, V_j is composed of two subsets: V_j^+, containing the *active vertices* on which the operations/computations are done when G_j is processed, and V_j^- the *passive vertices* that will retrieve the results of operations/computations made on the active vertices associated to them in other graphs. Hence, a set of interconnected and non-redundant local graphs (Figure 2 (e)) spans the entire point cloud.

4. Computation of CVT from one depth map

Considering only one acquisition, a graph $G(V,E)$ is first computed with [BPA20]. We now explain how a CVT can be computed on the underlying surface by using this graph.

Sites initialization The initial position of sites can be determined in different ways, but it is known that a uniform pattern onto the surface limits the number of iterations during relaxation. As a Poisson-disk distribution provides an "almost uniform" sampling pattern, we use the graph-based sampling method proposed in [BPA20] to get the initial position of sites.

VC computation To determine the VCs, we were inspired by [PPA16] that proposes a parallelized algorithm on GPU for computing VTs on stereoscopic images. We generalized this algorithm to graphs, which allows considering any kind of connectivity. Our algorithm consists in computing each VC by region growing around each site in parallel. As the space M to partition is a surface, the metric d considered is the geodesic distance, here implemented as a Dijkstra propagation on the graph with weights on the edges. Figure 3 illustrates our graph-based partitioning technique, by showing the VT computed on one part of the point cloud *Templo Mayor* (courtesy of CyArk/Google Open Heritage Program [Goo]) by using only one depth map. As the partition is computed in the parameterization domain defined by the depth map, we can see that the VT is faithful to the underlying surface of the point cloud, and not to the ambient 3D space in which the point cloud is embedded.

Centroids computation Once the VT is obtained, the centroid is computed for each cell, and then the closest vertex of V from that centroid is selected.

5. Computation of CVT from several depth maps

We now extend this technique to point clouds constructed from a set of acquisitions. As input, a set of graphs $G = \{G_1, G_2, ..., G_N\}$

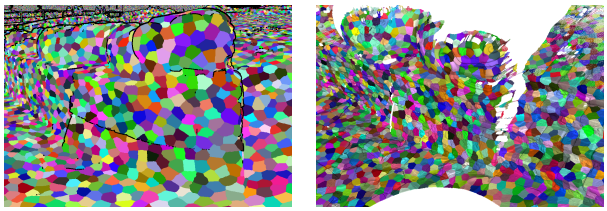


Figure 3: *VT of one region of* Templo Mayor *(courtesy of [Goo]) corresponding to one depth map. The partition is shown on the depth map (left), and in 3D space (right).*

is given, each graph representing one patch of the surface sampled by the point cloud. We remind that these graphs are interconnected in the overlapping regions, as explained in Section 3. This structure describes the entire point cloud piecewisely, it was thus natural to develop an algorithm that first computes one local CVT per graph, before merging and updating all these CVTs to get a global CVT.

5.1. Theoretical foundations

Such an approach is possible because of the local properties of a VT. Indeed, a VC can be constructed from only a subset of sites $S' \subseteq S$ surrounding it [Küh98]. As a consequence, all the sites out of its *Voronoi basin* do not affect its shape (See Figure 4). By exten-

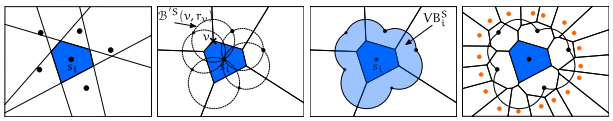


Figure 4: *(a) The dark blue VC is determined by intersecting the half-spaces of the subset of sites $S' \subseteq S$ surrounding the site s_i; (b) set of balls associated to these sites; (c) resulting Voronoi basin; (d) the orange sites, out of the Voronoi basin, do not affect the cell.*

sion, considering only a subspace $M' \subseteq M$ and the subset of sites $S' = S \cap M'$, a subset of VCs can be defined *globally*, *i. e.,* as if those cells were defined on the entire space M with the knowledge of all the sites S. The cells concerned are those whose basin is entirely in M' [Ble18]. *Partial cells* [For87] can also be defined in M' for sites which are between the cells globally defined and the borders of M' (see Figure 5). Such a cell contains all the points closer to its respective site than to all other sites of S' and also to the borders of M'. The resulting *partial VT*, composed of globally defined cells and of partial cells will be updated to provide the global VT, once the space is entirely known.

5.2. Proposed workflow

Inspired by these theoretical foundations, we develop an algorithm to compute a global VT on the underlying surface of a point cloud structured by several interconnected local graphs. Figure 6 details the concept with a textbook case, by considering only 2 neighboring graphs G_1 and G_2. By using the algorithm presented in Section 4 on each graph G_i, a partial VT can be computed on a subspace M_i'

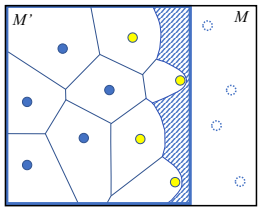


Figure 5: *Partial VT computed on $M' \subseteq M$ [For87]. Some VCs are globally defined (blue sites), while others are partially defined (yellow sites). The global VT will be obtained as the entire space M and the exterior sites are known.*

defined only by the active vertices V_i^+. The border of M_i' is depicted by the vertices of V_i^+ adjacent to at least one vertex of V_i^-. Figure 6(a) and Figure 6(b) show the partial VT obtained on G_1 and G_2 respectively, composed of cells globally defined (containing blue nodes) and cells partially defined (containing yellow nodes). The hatched nodes are active vertices closer to the border of M_i' than to its sites (similar to the hatched zone in Figure 5 that requires the knowledge of $M - M'$ to be partitioned). The white nodes represent the passive vertices V_i^- that will be processed in another graph. In a second round, each active vertex of each graph indicates to its associated passive vertices (in the other graphs) the site to which it is the closest, as well as the distance from this site. All the graphs are thus informed that some vertices in the common zones already belong to specific cells (see Figure 6(c)). In a third round, this information is considered to finalize the global partition in the areas that were previously closer to the border than from any site (Figure 6(d)). This approach can be generalized to N graphs, as shown by Figure 1. CVTs can be also easily constructed with this workflow, by alternating them with centroid computations which can be done similarly to the case of one depth map, as described in Section 4.

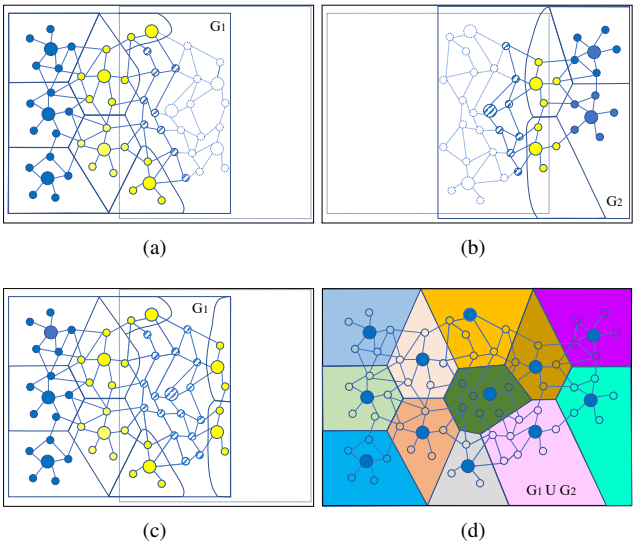


Figure 6: *Proposed approach to compute a global VT on a graph combining two interconnected local graphs.*

6. Experimentations

Figure 7 shows VTs obtained on popular models, *Egea* (courtesy of [AV]) and *Armadillo* (courtesy of [LGCP]), from 12 depth maps (1024x1024 pixels), generated from 12 virtual points of view positioned regularly all around each model. The VTs are globally satisfactory, and do not suffer from artifacts in the overlapping regions, thanks to the distinction between active and passive vertices during computations.

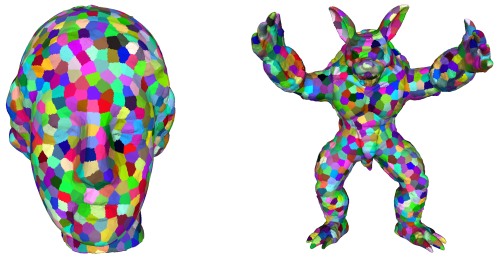


Figure 7: *VT computed on* Egea *and* Armadillo *with our approach.*

To further appreciate the efficiency of our workflow, Figure 8 shows the Poison-disk uniform distribution obtained with [BPA20] on *Dragon* (courtesy of [LGCP]) on the left, and the same distribution after convergence of our CVT on the right. We can notice that the relaxation leads to a more uniform distribution, as expected. Figure 9 now shows close views of adaptive distributions (the graph edges are weighted by the L_2 norm and the local curvature, as in [BPA20]) on real life data (*Eim Ya Kyaung* (1.337 billion points, 58 acquisitions, courtesy of [Goo]) generated with [BPA20] on the left, and with our CVT approach on the right. Both methods tend to attract the sites in the most curved regions, which emphasizes the fine details of surfaces, but our CVT technique also allows to distribute the vertices more uniformly all over the surface, which is interesting to improve sampling quality of point clouds.

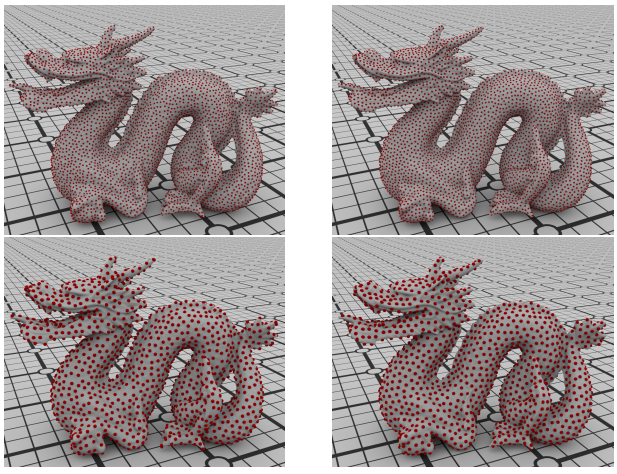


Figure 8: *Poisson-disk distributions [BPA20] on* Dragon *(left), and the same distributions after convergence of our CVT technique. Top/bottom: two distinct sampling densities.*

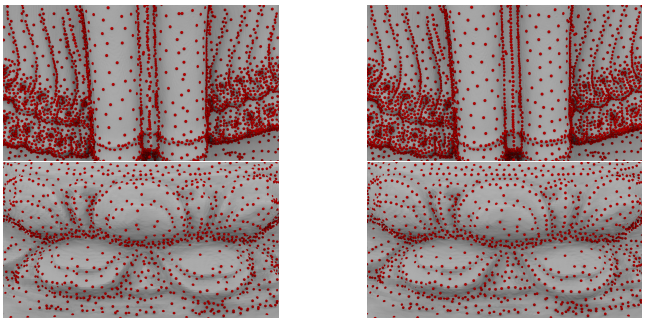


Figure 9: *Left: adaptive Poisson-disk distributions [BPA20]; right: same distributions after convergence of the CVT, on parts of* Eim Ya Kyaung *(1.337 billion points, 58 acquisitions).*

All those sampling patterns have been analysed with [WW11]. The results, which can be seen in [Ble18], show that the sampling distributions are improved by our relaxation algorithm. Once a VT is generated, one can obtain a triangulation from the computed partition by connecting the sites of neighboring cells. Figure 10 shows the reconstruction of *Meeting House* (courtesy of [Goo]) from an adaptive Poisson-disk distribution [BPA20], and the same reconstruction after convergence of the CVT. The histograms, showing the respective angle distribution, attest that the mesh quality is globally improved by our CVT. Finally, we present computation times

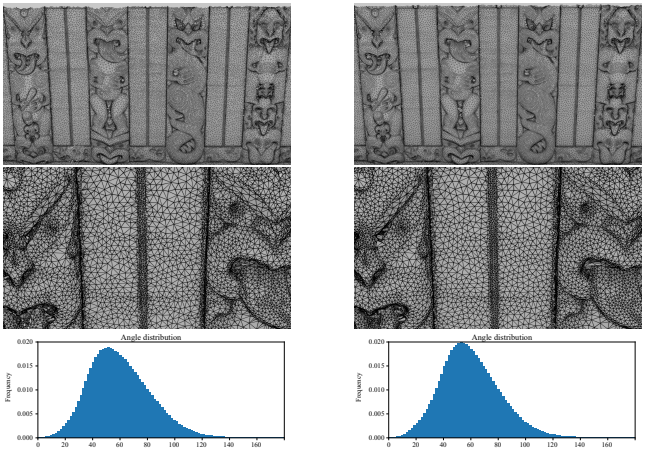


Figure 10: *Left:* Meeting House *(457 million points, 22 acquisitions) reconstructed from an adaptive Poisson-disk distribution [BPA20]; right : after convergence of the CVT. The histograms show the angle distribution of each reconstruction.*

and peak memories reached when constructing the CVT from a given set of sites, of two real life data: *Meeting House (interior)*, which contains 457 million points for 22 acquisitions, and *Templo Mayor*, which contains 349 million points for 17 acquisitions. Despite hundreds of millions of points as input, our algorithm requires less than 5GiB (2.3GiB and 4.4 GiB respectively), to construct a global CVT from the original depth maps (8192×8192 pixels). The corresponding computation times are $37mn$ and $1h05mn$, respectively. Additionally, as shown in [BPA20], depth maps can be subdivided before structuring the data, to further reduce the peak memory. For example, with depth maps cut into tiles of dimension (4096×4096), similar CVTs can be obtained with less than 2GiB (1.3GiB and 2GiB respectively), at the expense of longer computation times ($1h56mn$ and $2h39mn$, respectively). This is one interesting feature of our algorithm: the memory peak can be managed, so that it runs even on computers with limited RAM capacities.

7. Conclusion

We introduce a new method for computing (centroidal) Voronoi diagrams on large-scale point-based surfaces, generated from sets of depth maps. The originality of our approach comes from the use of interconnected local graphs allowing to create partial Voronoi diagrams, and then to merge/update them to finally get a globally consistent Voronoi diagram. An out-of-core implementation makes the processing of massive point clouds possible with an extremely low memory cost, even on point clouds composed of hundreds of millions of samples. This work is in progress, but our first results are really encouraging. Nevertheless, our method can be improved. For example, some triangulations, in particular with complex scenes, may suffer from artifacts (non-manifold elements, holes, etc.), due to the discrete aspect of our method and variable sampling densities between neighboring graphs. Our next step is thus to integrate validity tests on local VT configurations [Gus07], and/or to add a post-processing technique to fill remaining holes [BL17].

References

[AV] AIM@SHAPE-VISIONAIR: Aim@shape-visionair shape repository. URL: http://visionair.ge.imati.cnr.it. 3

[BL17] BOLTCHEVA D., LEVY B.: Surface reconstruction by computing restricted voronoi cells in parallel. *Computer-Aided Design 90* (2017), 123–134. 4

[Ble18] BLETTERER A.: *Une approche basée graphes pour la modélisation et le traitement de nuages de points massifs issus d'acquisitions de LiDARs terrestres*. Theses, Université Côte d'Azur, Dec. 2018. 2, 4

[BPA20] BLETTERER A., PAYAN F., ANTONINI M.: A local graph-based structure for processing gigantic aggregated 3D point clouds. *IEEE Transactions on Visualization and Computer Graphics* (Dec. 2020), 1–1. 1, 2, 3, 4

[For87] FORTUNE S.: A sweepline algorithm for voronoi diagrams. *Algorithmica 2*, 1-4 (1987), 153. 2, 3

[Goo] GOOGLE/CYARK: Open heritage. URL: https://artsandculture.google.com/project/cyark. 2, 3, 4

[Gus07] GUSKOV I.: Manifold-based approach to semi-regular remeshing. *Graphical Models 69*, 1 (jan 2007), 1–18. 4

[Küh98] KÜHN U.: Local calculation of voronoi diagrams. *Information processing letters 68*, 6 (1998), 307–312. 2

[LGCP] LEVOY M., GERTH J., CURLESS B., PULL K.: The stanford 3d scanning repository. URL: http://graphics.stanford.edu/data/3Dscanrep. 3

[Llo82] LLOYD S.: Least squares quantization in pcm. *IEEE Transactions on Information Theory 28*, 2 (1982), 129–137. 1

[PPA16] PEYROT J.-L., PAYAN F., ANTONINI M.: From stereoscopic images to semi-regular meshes. *Signal Processing: Image Communication 40* (2016), 97 – 110. 2

[WW11] WEI L.-Y., WANG R.: Differential domain analysis for non-uniform sampling. *ACM Transactions on Graphics 30* (2011), 50:1–50:10. 4