A local graph-based structure for processing gigantic aggregated 3D point clouds

Arnaud Bletterer, Frédéric Payan, Marc Antonini

A local graph-based structure for processing gigantic aggregated 3D point clouds

Arnaud Bletterer, Frédéric Payan, and Marc Antonini, *IEEE member*

***Abstract*—We present an original workflow for structuring a point cloud generated from several scans. Our representation is based on a set of local graphs. Each graph is constructed from the depth map provided by each scan. The graphs are then connected together via the overlapping areas, and careful consideration of the redundant points in these regions leads to a piecewise and globally consistent structure for the underlying surface sampled by the point cloud. The proposed workflow allows structuring aggregated point clouds, scan after scan, whatever the number of acquisitions and the number of points per acquisition, even on computers with very limited memory capacities. To show that our structure can be highly relevant for the community, where the gigantic amount of data represents a real scientific challenge per se, we present an algorithm based on this structure capable of resampling billions of points on standard computers. This application is particularly attractive for simplifying and visualizing gigantic point clouds representing very large-scale scenes (buildings, urban scenes, historical sites...), which often require a prohibitive number of points to describe them accurately.**

***Index Terms*—Computational Geometry and Object Modeling, Three-Dimensional Graphics and Realism, 3D Point Clouds, Data Structure, Graphs, Out-Of-Core Algorithms, Poisson-disk sampling.**

I. Introduction

3D points collected by depth cameras or LiDAR scanners can be seen as *structured* (or organized) data because the depth maps (or range images) provided by these devices bring implicit connectivity between points. Processing a structured point cloud is thus relatively easy, comparably to processing a 2D image, as the spatial connectivity is known.

Regarding the capture of a complex object or a large-scale scene, the latter must be scanned from multiple points of view. The incoming sets of 3D points are then registered and aggregated, to obtain the desired point cloud. The aggregation inevitably produces an *unstructured* point cloud, which complicates many operations because of the loss of sampling regularity and spatial connectivity.

To overcome this problem, one prevalent approach is to partition the ambient 3D space with an embedded structure, so that the points are contained in *3D cells*. Structures based on bounding spheres are proposed in [3], [4]. In [5], [6], a partitioning of the 3D space into voxels is proposed, where each voxel represents one point in the final structure. Many approaches based on octrees have also been proposed in the two last decades: [7], [8], [9], [10], [11]. The advantage of these methods is to offer a structure to point clouds, as the incidence information of the 3D cells is known. Although efficient for compression or visualization, for instance, these data structures may be inappropriate because they structure the ambient 3D space, and not the underlying surface described by the acquired points. This can be problematic with scenes composed of small elements, where spatial structures are, in essence, unable to distinguish neighboring points in space and neighboring points onto the surface.

Recently, a graph-based structure has been proposed as an alternative for processing 3D point clouds [12]. This structure has the advantage to offer incidence information to the points themselves, which allows processing the underlying surface directly. However, regarding the gigantic point clouds daily created with 3D sensors of higher and higher resolutions, sometimes from hundreds of acquisitions, this graph-based structure is inappropriate if the data size exceeds the memory capacity of the user's machine. Actually, this happens rather quickly, as graph-based structures are a particularly expensive representation.

In this paper, we propose an alternative to [12] for structuring point clouds with graphs. Our motivation is to benefit from the strength of a graph-based representation that naturally offers topology information to a set of points onto a surface while being able to manage any point cloud, whatever the number of points it contains. Our data structure is dedicated to point clouds resulting from the aggregation of multiple scans, in particular when the data size exceeds the RAM capacity of the user's machine.

By contrast to [12] that makes a unique graph for an entire cloud, our structure consists of a finite set of *local graphs*. Each local graph is associated with one scan, and describes one part of the point cloud, from a specific point of view. One specificity of the proposed approach lies in the usage of the structured depth map associated with each scan to construct large local graphs.

As it is, our structure would be inefficient, due to multiple distinct descriptions of the surface in the overlapping regions. To avoid redundant operations in these areas, but also to have a structure globally consistent all over the surface despite our "piecewise" representation, our structure links together the local graphs by interconnecting them, and prioritizing computations in the overlapping regions.

Our structure has major advantages: it can be constructed iteratively, scan after scan. It also enables efficient local

The authors are affiliated to Université Côte d'Azur, CNRS, I3S
This work was supported by a grant from Région Sud (France).
This work was partly presented in [1] and [2].

computations on gigantic point clouds without loading the whole structure into memory. It is thus particularly convenient for processing point clouds in an out-of-core manner.

To summarize, the contributions of this paper are:

- an original data representation based on local graphs constructed from depth maps that permits to structure aggregated point clouds with a low memory footprint, whatever the number of acquisitions and the number of points per acquisition;
- the specific management of the overlapping areas to explicitly interconnect the local graphs, which allows having a structure globally consistent all over the surface;
- a new resampling algorithm based on this structure that allows us to improve the sampling quality of gigantic point clouds reaching several billions of points, with direct control on the RAM capacity needed.

The rest of the paper is organized as follows. Section II presents the workflow and the proposed structure based on local graphs. Section III presents details of implementation and memory consumption. To show the potential of the proposed structure, Section IV presents an efficient resampling algorithm based on it. Finally, Section V summarizes the paper and proposes several promising perspectives.

II. Proposed data structure

Input data Let us consider a scene/object scanned from N known points of view. This procedure provides a set of N depth maps, and the entire set of parameters permitting to generate the corresponding *local* point clouds independently, and also to merge (or aggregate) them together in a global frame. If positioning information is not provided with the files, a vast literature about rigid registration of 3D point clouds exists, and the interested reader can refer to [13] for a survey. Such sets are the typical data provided by terrestrial LiDAR systems, for instance. They can be also synthetically generated from 3D models (Figure 1) with an approach similar to [14] .

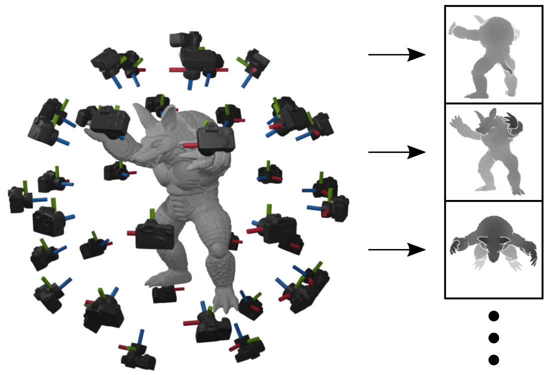


Fig. 1. Generation of synthetic depth maps from surface meshes.

Overview Figure 2 gives a schematic overview of our solution for structuring an aggregated point cloud.

- **Construction of local graphs** From each of N depth maps D_i $i \in \{0, 1, .., N\}$ (Figure 2 (b)), and their embedding function $e_i : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ (giving the position in 3D space of the points associated with the pixels; the inverse function is called *projection*), a local graph G_i is constructed (Figure 2 (c)). This graph structures one part of the point cloud, describing one part of the scene acquired from a specific point of view (Figure 2 (a)).

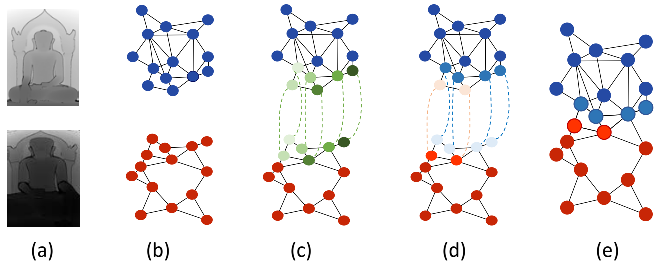


Fig. 2. Schematic overview of our workflow for structuring an aggregated point cloud. From each depth map (a) provided from a specific point of view, a *local graph* (b) is constructed to structure the associated 3D points; the local graphs are then interconnected (c) by matching the graph vertices describing redundant points in real world (depicted as green dots); these vertices are then assigned to only one graph (d). The output is a set of interconnected local graphs (e), giving a piecewise and globally consistent structure for the entire point cloud.

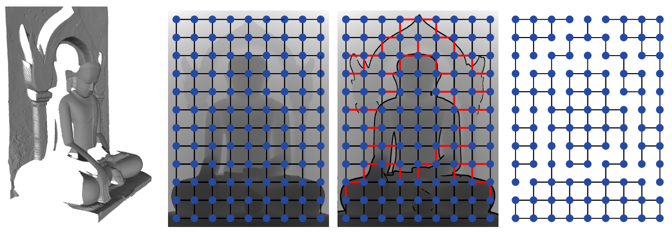
- **Interconnection of local graphs** To get a global structure of the whole aggregated point cloud, these local graphs are interconnected. This is achieved by detecting and matching graph vertices that describe similar regions of the scene (in other words, the overlapping areas) through embeddings/projections between the different depth maps. This stage results in a set of vertices *common* to several graphs, depicted by green dots in Figure 2 (d).
- **Management of matching vertices** The common vertices require specific handling to avoid redundant operations. Indeed, a local process in overlapping areas can be done in various ways, as those regions are modeled by several graphs of varying sampling density. The third stage consists in choosing for each set of redundant points the *best representatives*, *i.e.,* the vertices that will be actively used during computations. Hence, each redundant point is finally assigned to only one graph (Figure 2 (e)).

This workflow produces a set of interconnected and non-redundant local graphs that offers a piecewise and globally consistent structure to the underlying surface sampled by the point cloud (Figure 2 (f)). This structure can then be used as a support for different algorithms, such as the resampling technique presented in Section IV.

A. Construction of local graphs

LiDAR-like systems generate for each scan a depth map D. By dint of its embedding function e computed from the intrinsic and extrinsic sensor parameters, this map can be seen as a 2D parameterization of the local point cloud, that regularly samples one part of the scanned object/scene.

Structuring a local point cloud with a graph is natural, by taking advantage of the implicit connectivity information of its associated depth maps. Our procedure for constructing a graph G from a single depth map D is illustrated in Figure 3. The first stage (Figure 3(b)) consists in creating an undirected



(a) Point cloud. (b) Depth map and initial graph. (c) Segmentation. (d) Final graph.

Fig. 3. Overview of our method to construct a topology-aware graph from a depth map, to structure a 3D point cloud. Each graph vertex points out the 3D coordinates of the associated point in the cloud.

graph $G = (V, E)$ with, respectively, V and E the set of vertices and the set of edges linking the vertices. One vertex $v \in V$ is created for each non-zero pixel $d \in D$ (zero pixels represent directions in which the laser did not intersect the scanned surface, leading to pixel values equal to zero), and the coordinates (x, y, z) of the corresponding 3D point p in the real world is affected to it. Each vertex v is linked by an edge with the vertices in its k-neighborhood. Depending on the subsequent applications, different connectivities can be considered, which directly influences the accuracy and the complexity of the computations.

To structure the scanned surface correctly, one graph must also conform to its topology. Concretely, adjacent vertices associated with 3D points that are not close on the surface (in terms of geodesic distance) must be disconnected. This arises when two adjacent pixels are associated with 3D points belonging to two distinct elements in a scene. To detect such configurations, we use morphological gradients [15] to threshold the high depth variations in the depth map. Those gradients define *borders* in the depth map, depicting distinct elements of the scanned surface. The edges of the graph crossing those borders (in red in Figure 3(c)) are removed from E, and the final graph (Figure 3(d)) is obtained.

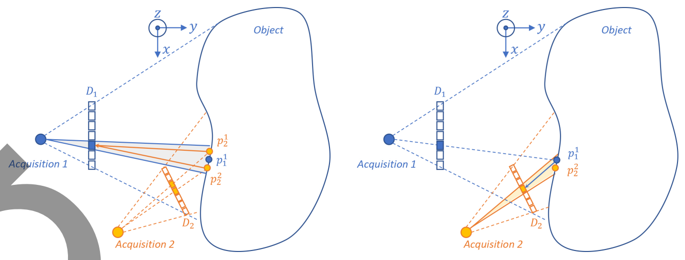
The threshold must be chosen wisely. If its value is too small, unacceptable over-segmentations occur in regions close to scanners. On the other hand, a large value could produce under-segmentation in regions far from scanners. But, in practice, this is less severe as these distant regions will be most probably processed using other graphs associated with scans closer to these regions (see Section II-C). Considering this, the threshold has been determined empirically, and currently follows an increasing function that depends on the depth values. Though perfectible, this solution gives satisfactory results with all the data tested during this project.

Regarding now a set of N depth maps, the same procedure can be followed for each, resulting in a set of N local independent graphs $\{G_1, G_2, ...G_N\}$ describing an entire point cloud from different points of view.

B. Interconnection of local graphs

To get a unique and global representation of the entire point cloud, we could merge all the graphs $\{G_1, G_2, ...G_N\}$ in a representation similar to the one proposed in [12], by using for instance a global parameterization based on transition functions [14]. Nevertheless, such approaches are unsuitable in the context of gigantic point clouds: a global representation could lead to a memory usage outreaching by far the RAM capacity of most computers. Instead, we propose to keep as a base representation the set of local graphs constructed at the previous step and to enhance it with additional data that explicitly describe interconnections between them.

The local graphs can be interconnected via the overlapping areas. Figure 4 illustrates the concept of our approach, in the case of two depth maps. Let us consider the point p_1^1 assigned to the vertex $v_1^1 \in G_1$, and the two points p_2^1 and p_2^2 assigned to the vertices $v_2^1 \in G_2$ and $v_2^2 \in G_2$ (Figure 4(a)). If one looks at these two points from the point of view of the acquisition 1, because of the discrete nature of the 3D sensors, these two points would be projected onto the same pixel of D_1. From this point of view, p_2^1 and p_2^2 are thus *duplications* of p_1^1. Regarding the graphs, the vertices v_2^1 and v_2^2 thus match the vertex v_1^1, creating two interconnections from G_2 to G_1: $(v_2^1, v_2^2) \to v_1^1$. Reversely, from G_1 to G_2, v_1^1 matches the vertex v_2^2, creating one interconnection: $v_1^1 \to v_2^2$ (Figure 4(b)).

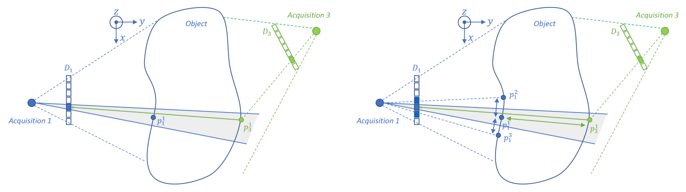


(a) From D_2 to D_1, p_2^1 and p_2^2 are *duplications* of p_1^1, creating two interconnections $(v_2^1, v_2^2) \to v_1^1$.

(b) From D_1 to D_2, p_1^1 is a *duplication* of p_2^2, creating one interconnection $v_1^1 \to v_2^2$.

Fig. 4. Approach proposed for interconnecting the local graphs in the overlapping regions.

As it is, this matching strategy does not manage occlusions. The schematic example of Figure 5(a) illustrates a direct consequence of this: from the first point of view, the point p_3^1, although positioned on the opposite side of the scanned object, is considered as a duplication of p_1^1, which would lead to an incorrect interconnection from G_3 to G_1: $v_3^1 \to v_1^1$. To overcome this unwanted effect, we follow a heuristic approach that consists in calculating the mean distance $\bar{d}$ between p_1^1 and the points associated with its neighboring vertices in G_1: if $\bar{d}$ is lower than the distance $d(p_1^1, p_3^1)$, we consider that p_3^1 is not a duplication of p_1^1, and the interconnection is not created (Figure 5(b)). The idea behind that heuristic is that, when working with depth maps, it is only possible to capture holes having a size at least larger than the distance between points corresponding to neighboring pixels. As a consequence, we consider that filtering out wrong interconnections based on this particularity would lead to a coherent behavior with the acquisition process. Although quite basic, this technique is efficient on the datasets experimented during this project. One could improve such a strategy by locally modeling the underlying surface of the point cloud (with planar or quadratic approximations for example), or by using the method of [16].



(a) Without occlusion awareness, wrong interconnections would be created ($v_3^1 \rightarrow v_1^1$).

(b) With occlusion awareness, wrong links are prevented.

Fig. 5. Heuristic approach for preventing wrong graph interconnections, in case of occlusions.

By generalizing this approach to all the depth maps, these interconnections allow us to structure the entire point cloud, and to navigate from a graph to another as in a global graph-based structure. We note that the interconnections could be used to create a unique global graph, by merging interconnected vertices for example. However, such an approach would defeat the purpose of controlling memory usage.

Determining the interconnections is time-consuming, as all the vertices of all the graphs must be projected onto all the other local graphs, to establish which vertices should be interconnected together (with a time complexity of $O(s * N)$ where s is the total number of vertices and N is the number of local graphs). Therefore this operation is done once and for all in our structuration workflow, and the interconnections are definitively described as additional edges, and stored in our data structure. In the next section, we explain how these edges are included in our data structure with a minimal additional storage cost. Another advantage of including the interconnections in the structure is that for any vertex in any graph we directly know its spatial neighborhood thanks to the knowledge of the intra-graph and inter-graph edges.

C. Management of interconnected vertices

As explained before, an interconnection links two vertices associated with points considered as duplications in the scene. So, if an operation/calculation must be done in this region of the scanned surface, several graphs can be used. The question that flows naturally is *which graph is the best to make this operation/calculation?*

The efficiency or accuracy of many computations made on 3D point clouds strongly depends on the sampling density. Yet, the sampling density of a scanned surface can be extremely different, according to the position and orientation of the scanner. As a basic example, if a surface is scanned from two points of view, one close and one far, the point cloud resulting from the first acquisition will be denser than the cloud resulting from the second acquisition. If we have to select one of the two sets of points to make an operation we will naturally select the first one, because it is more detailed, and thus the result will be more precise. The same argument can be done in function of the orientation of the surface: a surface scanned with a low incidence angle is better sampled than with a high incidence angle.

Inspired by these observations, to select on which graph a given operation has to be done in overlapping regions, we chose the *local sampling density* as a criterion. To well understand, let us consider a small region of a scene that is assigned to several interconnected vertices, hereinafter called the *set of interconnected vertices*. To apply the operation in this region, we will retain among this set the vertex v with the highest local sampling density $w(v)$ in its graph, estimated with the following function

$$\frac{1}{w(v)} = \frac{1}{|\mathcal{N}_1(v)|} \sum_{v' \in \mathcal{N}_1(v)} ||P(v') - P(v)||^2,$$

where $\mathcal{N}_1(v)$ is the $1-$neighborhood of v in the concerned graph, and $P(v)$ and $P(v')$ the 3D coordinates of the point assigned to v and v', respectively. This vertex v will be considered as the *best representative* of this region, among this set of interconnected vertices. By generalization, the vertices of each graph can be classified into two groups:

- *active* : containing the vertices considered as the best representatives among each set of interconnected vertices, and the vertices that are not interconnected to any other graph vertex;
- *inactive* : containing all the other vertices.

Those two groups finally define the vertices on which operations should be computed (*active* vertices) and the vertices that should query the result of operations from their best representative (*inactive* vertices). Doing so permits to process the whole set of points by iteratively processing each local graph and to transmit information between local graphs through interconnections.

The choice of the best representative for each set of interconnected vertices is done once and for all during the structuration workflow, and this information is stored as additional information in the data structure as explained in the next section.

III. Technical Specifications

This section details how the structure presented in the previous section can be implemented to get a low memory footprint.

Details of implementation Regarding the local graphs, our implementation can benefit from the fact that each graph is constructed from a depth map. The set of vertices of each graph can be seen as a simple matrix, where each element corresponds to one pixel and contains the 3D coordinates of the associated point in the cloud, and any additional attribute (color, intensity, normal, etc...).

Considering the edges, their storage is not needed, as the connectivity information is implicit and can be computed on the fly from the structure of the depth map. However, to consider the edges removed by the morphological gradients that in a certain way break this implicit connectivity, we store the thresholded gradients as additional attributes in each graph. When reconstructing the connectivity on the fly, one only has to check this information, to know if vertices are connected in our graph or separated by a border. As there are generally

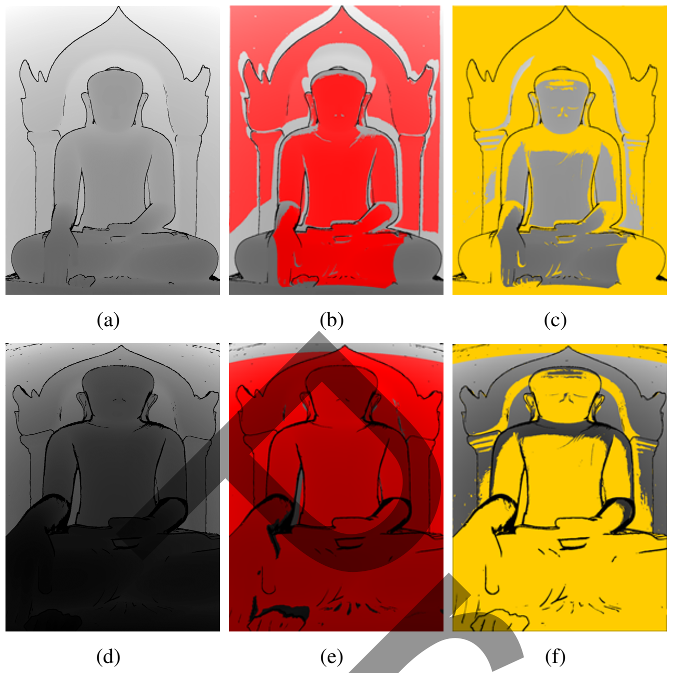


Fig. 6. Management of the interconnected vertices. (a)/(d): two overlapping depth maps associated with two graphs, G_1 and G_2. (b)/(e): in red, the set of pixels both associated with vertices of G_1 and G_2, i.e., the vertices interconnecting the two graphs. (c): in yellow the set of pixels associated with vertices considered as *active* in G_1. These vertices are either best representatives among the vertices interconnecting the two graphs, or assigned to 3D points that are not covered by G_2. (f) In yellow the set of pixels associated with the vertices considered as *active* in G_2.

much fewer edges removed than total edges in each local graph, this is a very sparse additional information. Note that this implementation also involves the storage of zero-pixels (inherent to the matrix structure), but this is quite negligible in comparison to the drastic reduction of memory usage due to the implicit connectivity representation.

Regarding the interconnections between graphs, it is not possible to store them with a similar strategy. The set of interconnected vertices is stored for each vertex, *via* a usual adjacency list representation. Concretely, each vertex stores the vertices of other graphs that are connected with it. To include the notion of *active* vertices, an additional integer value is assigned to each vertex. This value indicates if a vertex is its own best representative (in that case, the value is set to 0), or if another vertex is the best representative (in that case, the value indicates the position of the best representative in its list of interconnected vertices).

Memory consumption Table I indicates in the 4^{th} column the storage cost of the proposed structure as specified immediately above for several large real-life datasets. Each dataset consists of tens of terrestrial LiDAR acquisitions (3^{rd} column), providing up to billions of points (2^{nd} column). These results have been generated with a [Intel Core i7-5960X CPU @ 3.00GHz, NVIDIA Quadro M5000, 32GB RAM and 1TB SSD]. We observe that, even with almost implicit connectivity information for each local graph, the total storage cost of our structure remains important. However, even though these costs overflow the RAM capacity of our computer, our algorithm can process these datasets thanks to very low memory usage (8th column, the first line of each row). Due to the local graph-based approach, even with billions of points and structured data of hundreds of GiB, the memory peak is always lower than 6GiB. In other words, always less than 10% of the whole structure, whatever the data. This is achieved by opting for a strategy that keeps in memory only the information required at each step of the structuration workflow. Representing the vertex attributes in matrices makes swapping data between RAM and disk trivial.

Also, the memory peak is correlated to the dimension of the depth maps on which the graphs are based. So the memory reduction is all the more significant as the number of acquisitions is high. Our structure is thus particularly relevant for processing data coming from massive acquisition campaigns when several hundreds of scans are needed to capture the entire scene.

Management of out-of-core scans A single scan may contain hundreds of millions of points, providing a graph that may exceed the RAM capacity itself, despite the implementation introduced above. To address this issue, our algorithm integrates the ability to split depth maps into *overlapping tiles* (Figure 7) before structuring the point cloud.

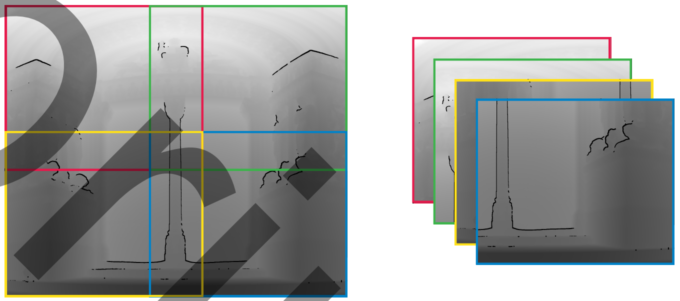


Fig. 7. Division of a depth map into overlapping tiles, to cap the memory usage.

These tiles are then considered as any depth map by the structuration workflow, and one graph per tile is constructed. Note that, by construction, the interconnection between graphs covering tiles is known, the relative pixels in the overlapping areas being the same. Also, as the local sampling density between tiles is also the same, we decide to select the *active* vertices in these areas with respect to the distance from their associated pixels to the border of the tiles.

Table I indicates the memory peak reached when a tiling is done for several datasets. This functionality allows reducing the memory footprint significantly and finally, by tuning the tile size, to strictly curb it. This functionality is particularly attractive if one has to structure point clouds on machines with very low RAM capacities. The counterpart is an increase in the runtime, as the number of local graphs quickly rises. This involves a higher complexity in the construction of interconnections, and the disk I/O are more frequent.

Data	# Points [M]	# Acq.	Cost [GiB]	Tiling	Tiles ($p \times p$)	# Tiles	Mem. [GiB] (Gain)	Time [h:m:s]
TEMPLO MAYOR	349	17	46.2	no	n/a	n/a	3.5 (-92.4%)	00:07:23
				yes	8192	33	1.8 (-96.1%)	00:09:25
				yes	4096	55	1.0 (-97.8%)	00:11:48
				yes	2048	213	0.6 (-98.7%)	00:25:57
PALAIS (EXT.)	977	23	71.8	no	n/a	n/a	5.7 (-92.1%)	00:15:22
				yes	8192	54	2.8 (-96.1%)	00:25:14
				yes	4096	162	1.0 (-98.3%)	01:00:36
				yes	2048	490	0.6 (-99.0%)	03:04:53
MEETING HOUSE	1,493	50	139.0	no	n/a	n/a	5.7 (-95.9%)	00:56:28
				yes	8192	101	2.9 (-97.9%)	01:30:18
				yes	4096	278	1.1 (-99.2%)	03:56:03
				yes	2048	933	0.6 (-99.6%)	12:32:46
PALAIS (INT.)	1,748	37	135.7	no	n/a	n/a	5.7 (-95.2%)	00:43:00
				yes	8192	78	2.8 (-97.9%)	01:04:51
				yes	4096	234	1.1 (-99.1%)	02:26:14
				yes	2048	704	0.6 (-99.5%)	04:30:21
ANANDA OAK KYAUNG	1,703	126	158.7	no	n/a	n/a	1.9 (-98.8%)	02:17:57
WAT PHRA SI SANPHET	5,313	177	641.8	no	n/a	n/a	5.7 (-99.1%)	10:35:28

TABLE I
GLOBAL STORAGE COST (4^{th} COLUMN), MEMORY PEAK (8^{th} COLUMN) AND RUNTIME (9^{th} COLUMN) OF OUR ALGORITHM FOR STRUCTURING REAL-LIFE DATASETS, WITH OR WITHOUT SPLITTING DEPTH MAPS. THE PERCENTAGES IN THE 8^{th} COLUMN INDICATE THE GAINS OF MEMORY USAGE WITH RESPECT TO THE GLOBAL STORAGE COST OF THE STRUCTURE.

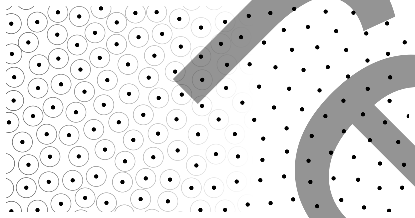


Fig. 8. Maximal Poisson-disk distribution on a planar domain.

IV. RESAMPLING BASED ON LOCAL GRAPHS

We now showcase one popular processing that greatly benefits from the proposed structure: the resampling of point clouds, notably the point clouds provided by digital recordings of large-scale 3D objects/scenes (such as buildings, urban environments, monuments, historical sites...). These point clouds have several characteristics that fully justify a resampling technique based on a piecewise structure based on local graphs like ours: i) they are obtained by merging tens or hundreds of scans, ii) they can contain billions of points, which severely limits their usage, iii) the point density presents strong local variations, in function of position and orientation of scanners on the site, but also because of overlapping areas, among other things.

Our resampling technique is based on Poisson-disk distributions [17], which have the advantage to reduce the number of points while improving the sampling quality by dispatching the points uniformly but irregularly on a surface. Figure 8 illustrates this feature: a minimum distance is ensured between the samples, but with the constraint that the samples do not lie on a spatial regular lattice.

Multiple methods exist to produce Poisson-disk distributions on surface meshes [18], [19], [20], [21], sometimes even to handle point clouds [22]. However, those methods are either requiring a triangulation to work on or considering a euclidean metric instead of a geodesic one (for computational purposes), which can lead to a loss of details for very thin structures that generally have a high genus and a complex geometry.

Recently, a *dart throwing* technique has been developed for resampling surface meshes with Poisson disks [23] according to a geodesic metric. Figure 9 illustrates this technique. Given the minimal distance $2r$ wanted for the final distribution, it consists in i) selecting one vertex among the candidate vertices randomly, ii) drawing a disk of radius r around the selected vertex, and iii) checking if this disk overlaps another disk already constructed. If it does not intersect any existing disk, this vertex is kept for the final set of vertices, and all the vertices covered by the disk are removed from the list of candidates. This process is repeated until there is no possibility to insert a new disk on the surface (in that case, the maximal Poisson-disk distribution is reached).

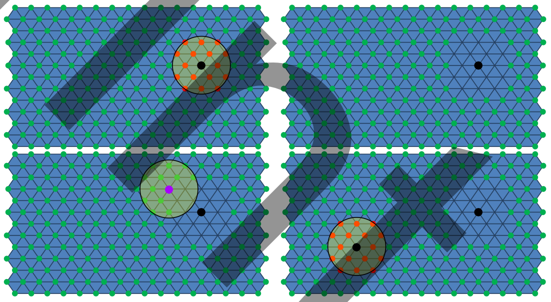


Fig. 9. Dart throwing on a triangle mesh, as proposed in [23] (from left to right, top to bottom). The black dots represent candidate samples that are kept in the final distribution. The purple one is not kept, because its disk overlaps removed candidates.

The resampling algorithm presented in this section can be seen as a generalization of the algorithm introduced in [23] that only takes triangle meshes as input. Our new resampling algorithm can resample any point cloud structured with the representation based on interconnected local graphs described in Section II. The resampling is done graph after graph, which permits to cap the memory usage.

A schematic overview of this algorithm is illustrated in Figure 10. Let us consider a plain surface as a sampling domain,

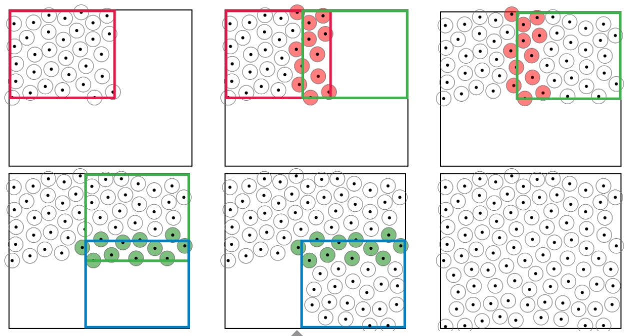


Fig. 10. Schematic overview of our iterative resampling scheme (from left to right, top to bottom). Top left: resampling of the red subdomain from its associated graph. Top middle: initialization of the green subdomain with the disks already positioned in the common zone. Top right: resampling of the green subdomain. Bottom left and middle: same procedure with the green and blue subdomains. Bottom right : final distribution. The transmission of disks is made *via* the lists of interconnected vertices.

described by several overlapping subdomains represented here by rectangles. The process is iterative and piecewise. A first subdomain (red rectangle) is maximally resampled with Poisson disks. A second zone (green rectangle) is then resampled, and so on until the entire sampling domain is resampled. To respect the essential rule (no overlapping disk) all over the surface, the disks already positioned in overlapping areas are "transmitted" to the other subdomains, to be considered during the next iterations.

In our context of aggregated point clouds structured with interconnected local graphs,

- each subdomain in Figure 10 represents a part of the inherent surface covered by one local graph;
- graphs are iteratively processed, starting from a randomly chosen graph;
- the resampling of a given graph is done by dart throwing similarly to [23] but suited for arbitrary connectivity, and by considering only the active vertices as candidate samples;
- the disks are determined by propagation, by using Dijkstra's shortest path algorithm [24];
- transmission of disks between subdomains is done *via* the graph interconnections. After having maximally resampled the active vertices of a graph, each active vertex covered by a disk "informs" all its interconnected vertices that they are already covered, ensuring valid Poisson disk distribution in overlapping areas. Hence, the memory usage is controlled, without sacrificing the quality of the final distribution.
- In the context of visualization, our graph-based structure offers the possibility of applying a *curvature-aware* resampling, to enhance salient features of the scanned surface. With Poisson-disk distributions, it consists in weighting the distances between neighboring vertices with a specific function to put smaller disks in curved areas, involving a higher density in these areas. With our structure, we simply weight the graph edges by the Gaussian curvature computed from the local graphs.

Figure 11 shows uniform distributions on three well-known models provided by our workflow from twelve depth maps (synthetically generated from the surface meshes, as explained in Section II). Visually, the distributions are satisfactory, for the three densities.

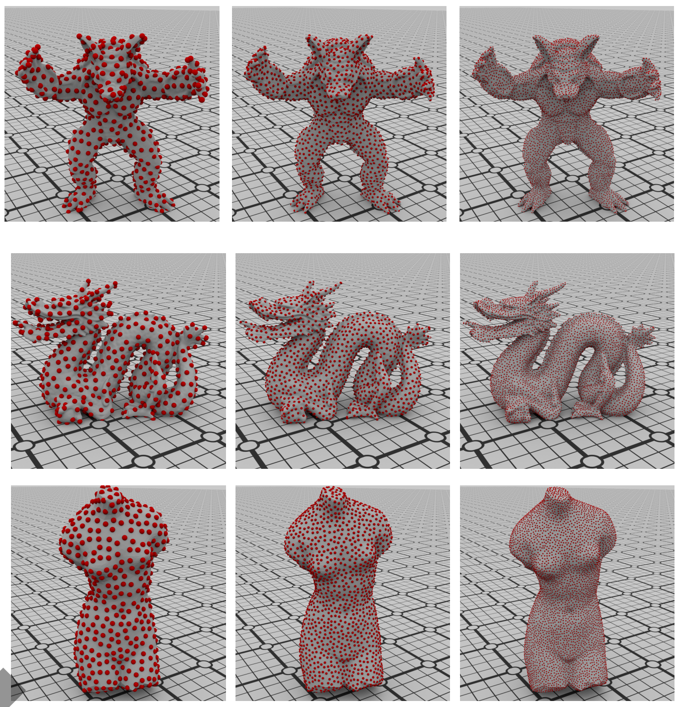


Fig. 11. Uniform Poisson-disk resampling of ARMADILLO, DRAGON and VENUS provided by our workflow for three different densities.

Figure 12 now compares uniform Poisson-disk distributions and the resulting uniform point clouds provided by our algorithm on FERTILITY, with or without transmission of disks between graphs. Without transmission (left side), many disks overlap, providing a distribution that does not respect the minimal distance between all the points. With transmission (right side), the minimal distance is ensured all over the surface, and there is no sampling artifact in overlapping areas, despite an iterative approach processing the graphs successively.

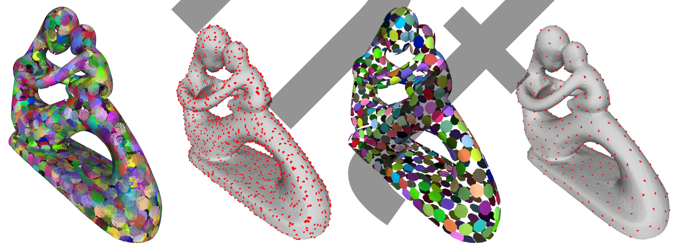


Fig. 12. Comparison of Poisson-disk distributions and resulting point clouds provided by our algorithm with (right) or without (left) taking into account the interconnected vertices already covered by disks. This comparison shows that the disks do not overlap with our approach, and the constraint minimum distance is ensured all over the surface, thanks to the specific management of overlapping areas.

To assess quantitatively the quality of our results, and thus to validate our resampling technique, we selected the tool developed by Wei et al. [25] that performs a differential analysis of the spread of the points on the surfaces. From any set of samples on surfaces, this tool computes a 2D power spectrum, and two statistics can be determined from it: the *Radially Averaged Power Spectrum (RAPS)* and the *anisotropy*. The RAPS shows the distribution of the distances between samples (computed in all the directions around each sample, and averaged radially), while the anisotropy shows the radial uniformity of the sampling pattern over the domain. The typical RAPS for a Poisson-disk distribution consists of a wide zero-region at low frequencies and a flat high-frequency region, both connected with a sharp transition depicting a peak at the cut-off frequency that corresponds to the minimum distance between samples. The typical anisotropy is constant and quite low.

Table II shows the statistics obtained with our resampling method on BUNNY and EGEA: they are typical of Poisson-disk sampling patterns. To confirm these good results, we also show in this table the statistics obtained with the method of [20] that resamples 3D models from triangle meshes, and thus benefits from the knowledge of the surface contrary to our method which inferred it from the connectivity of the generated graphs. These statistics are similar. This analysis highlights the fact that our scheme does not suffer from its iterative and piecewise approach, and that the inferred connectivity of our interconnected graphs is consistent with the explicit connectivity of the input surface mesh.

We now show several results obtained on large real-life datasets. For these datasets, bilateral filtering [26] has been applied to the depth maps right after the acquisition, to remove the acquisition noise.

Figure 13 and Figure 14 show two nice results of resampling on out-of-core data: PALAIS (INT.) composed of 1.5 billion of points (37 scans), and MEETING HOUSE composed of 1.75 billion of points (50 scans). These results notably emphasize the detail enhancement of our curvature-aware sampling strategy based on local graphs (Figure 13, right). Figure 13 shows the same scene resampled uniformly with a prior method presented in [22] (available in Meshlab [27]). This result is similar to ours, which attests that our resampling scheme competes with state-of-the-art methods. Nevertheless, to obtain this result with [22], we had to extract manually the part of the point cloud corresponding to the glazed roof, before resampling it only. Indeed, [22] is based on in-core implementation, and cannot be applied to gigantic datasets such as PALAIS (INT.). On the other hand, our method can resample the entire point cloud on the fly, by using 16GiB of RAM at most, and without tiling. In comparison, [22] requires approximately 7GiB of RAM, just to resample the part corresponding to the glazed roof, composed of 3.3% of the total amount of points.

Figure 15 shows another impressive result with the site of WAT PHRA SI SANPHET (40,000m^2, originally 5.3 billion of points, 156 scans) resampled with “only“ 155 million of points. This result is highly satisfactory since the processing of such a quantity of points is a real challenge per se, and our algorithm is finally able to resample this gigantic point cloud by using never more than 16GiB of RAM. Moreover, our method can efficiently dispatch the points all over the surface, while enhancing extremely subtle details (see for instance the details of the bricks) thanks to our curvature-aware sampling strategy. Our approach is thus particularly relevant in the context of visualization, especially when it comes to recovering complex details on gigantic sites.

Table III sums up the memory consumption of our resampling scheme on six point clouds composed of hundreds of millions of points. To highlight the trade-off between memory footprint and runtime, this table also includes performances when tiling depth maps as explained before. Similarly to our structuration workflow, our resampling scheme is able to deal with gigantic point clouds with a very low memory footprint. Without tiling, the memory peak is always less than 16GiB and can be reduced to fewer than 1GiB with tiles of 2048×2048 pixels. The counterpart is an increase of runtime, due to more frequent disk I/O. However, in function of the tile sizes, the runtime increases very slightly, whereas the memory footprint drops drastically. This is due to the complexity of the sampling algorithm that mostly depends on the number of total vertices and the size of the disks, and not on the number of graphs. As the tiling process adds only a few duplicate vertices between the different tiles of the same initial depth map (depending on the size of the overlapping area), the overall time does not change drastically. This is a very interesting feature of our algorithm, as resampling is often done once and offline, and RAM is often more limited than time for users with standard machines.

Lastly, Figures 17, 16, and 18 show surfaces reconstructed with the method of [28] from point clouds obtained with our resampling algorithm. We chose this reconstruction method because it does not alter the initial positions of the points. Hence, the triangle quality of the reconstructions is highly dependent on the distribution of the points on the surface. We can see that the resulting meshes are of high quality, suggesting that our structure can be highly relevant for the geometry processing community, where the gigantic amount of data alone represents a real scientific challenge per se.

V. CONCLUSION AND FUTURE WORKS

We propose an original workflow for structuring point clouds generated from sets of depth maps. The data representation is based on a set of local graphs, each of them describing a part of the scanned scene. To have a globally consistent representation of the entire scanned surface, the duplicate vertices in the overlapping regions are interconnected and associated with only one graph, to avoid redundant operations.

This piecewise representation allows the construction of the data structure scan after scan, while capping the memory usage. An optional cut of depth maps is also possible in preprocessing to further reduce the memory usage. Hence, billions of points can be structured even on computers with very limited memory capacity.

We believe that many applications can greatly benefit from this structure. We show for instance that Poisson-disk resampling can be efficiently done with only a few gigabytes of

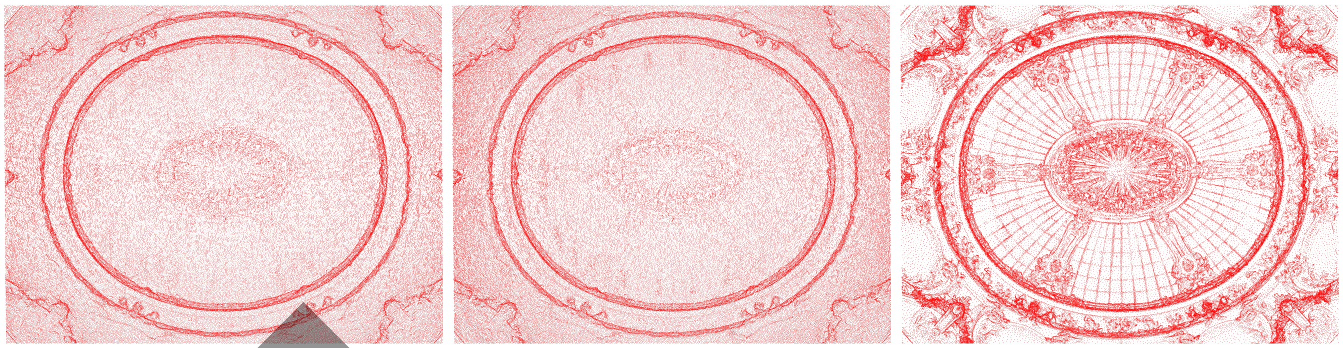


Fig. 13. Glazed roof of PALAIS (INT.) resampled with only 1% of points. From left to right: uniform sampling obtained with [22], with our uniform resampling and with our curvature-aware resampling.

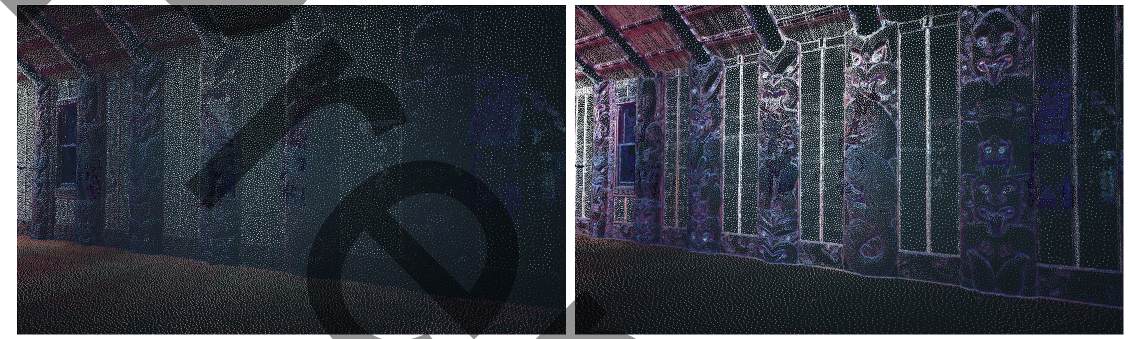


Fig. 14. MEETING HOUSE (1.5 billion of points, 50 acquisitions) resampled with only 1% of points. Left/right: uniform/curvature-aware sampling.

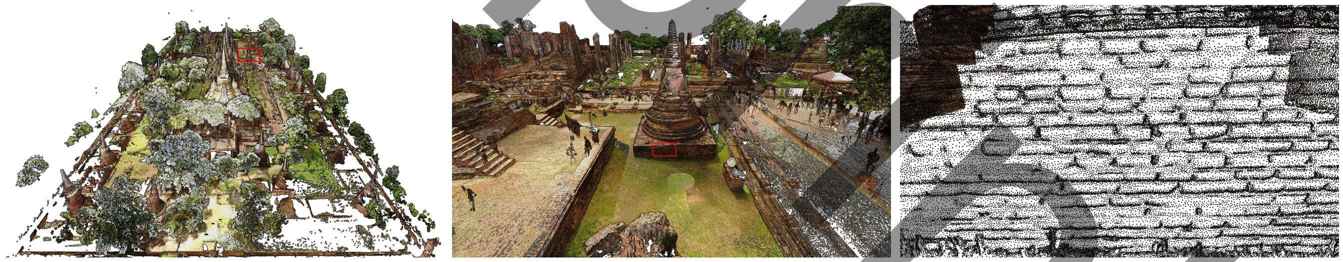


Fig. 15. The entire site of WAT PHRA SI SANPHET (Thailand, 40,000m^2) resampled with our method. From left to right: overview, and two levels of zoom (red boxes indicate where each zoom is done). Initial data: 5.3 billion of points, 156 acquisitions. This simplified cloud contains around 155 million of points (2.8%). Despite the extent of the site, fine details are preserved thanks to our curvature-aware sampling strategy.

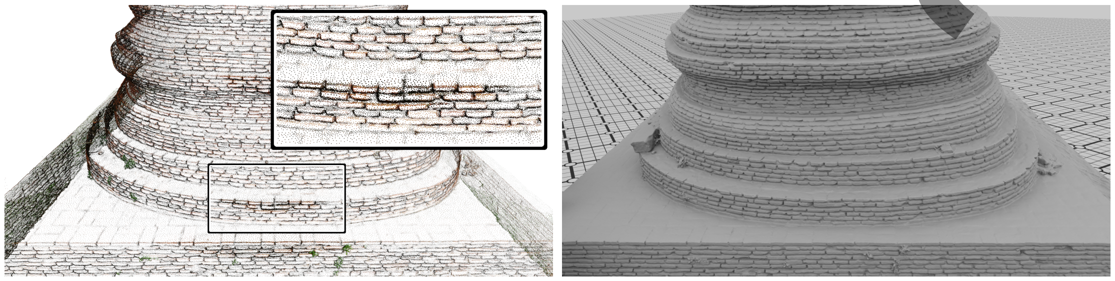


Fig. 16. WAT PHRA SI SANPHET: Close-up view of a point cloud provided by our curvature-aware resampling (left) and the resulting surface reconstructed by [28]. Without resampling stage, reconstruction was not possible, because of the amount of initial data as well as its bad sampling quality.

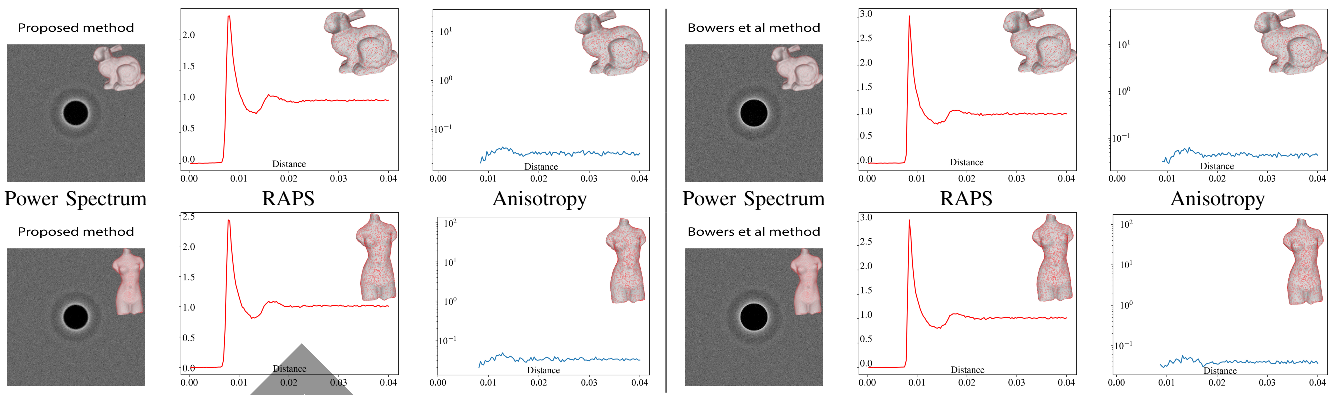


TABLE II
THE DISTRIBUTIONS GENERATED WITH OUR LOCAL GRAPH-BASED RESAMPLING ALGORITHM HAVE TYPICAL STATISTICS OF POISSON-DISK SAMPLING PATTERNS. THESE PROFILES ARE SIMILAR TO THOSE OBTAINED WITH [20], A PRIOR METHOD THAT DIRECTLY RESAMPLES MODELS FROM TRIANGLE MESHES. THE DIFFERENTIAL ANALYSIS IS DONE WITH [25].

Data	# Points (M)	# Acq.	Tiling	Tiles ($p \times p$)	S. density	M.Peak [GiB] (Gain)	Time [h:m:s] (Gain)
TEMPLO MAYOR	349	17	no	n/a	1.2%	7.5	00:08:48
			yes	8192		3.9 (-47.8%)	00:09:01 (+2.5%)
			yes	4096		1.9 (-75.0%)	00:09:26 (+7.2%)
			yes	2048		0.8 (-90.0%)	00:10:08 (+15.2%)
PALAIS (EXT.)	977	23	no	n/a	0.4%	11.6	00:29:10
			yes	8192		6.2 (-46.7%)	00:29:58 (+2.7%)
			yes	4096		2.2 (-81.0%)	00:30:20 (+4.0%)
			yes	2048		0.8 (-93.2%)	00:30:24 (+4.2%)
MEETING HOUSE	1,493	50	no	n/a	1.1%	11.4	00:43:28
			yes	8192		5.8 (-49.3%)	00:44:29 (+2.3%)
			yes	4096		2.2 (-81.0%)	00:46:10 (+6.2%)
			yes	2048		0.8 (-92.8%)	00:47:55 (+10.2%)
PALAIS (INT.)	1,748	37	no	n/a	0.2%	15.9	00:56:15
			yes	8192		7.8 (-50.8%)	01:00:10 (+7.0%)
			yes	4096		2.2 (-86.1%)	01:00:39 (+7.8%)
			yes	2048		0.8 (-94.8%)	01:02:25 (+11.0%)
ANANDA OAK KYAUNG	1,703	126	no	n/a	0.7%	5.3	01:01:23
WAT PHRA SI SANPHET	5,313	177	no	n/a	0.7%	15.8	03:50:10

TABLE III
MEMORY CONSUMPTION AND EXECUTION TIMES OF OUR RESAMPLING ALGORITHM. THOSE RESULTS HAVE BEEN GENERATED FOR A PARTICULAR TARGET SAMPLING DENSITY (PRESENTED AS A PERCENTAGE OF REMAINING POINTS IN THE COLUMN *S. density*). PERCENTAGES OF TIME AND MEMORY INDICATE THE COST IN COMPARISON TO THE PROCESSING WITHOUT TILING STRATEGY.

RAM, on gigantic point clouds merging hundreds of scans and composed of several billions of points. This application is particularly attractive for dealing with point clouds representing buildings, urban scenes, historical sites, or any very large-scale scene.

Another potential application is surface reconstruction, which is still a serious challenge in the domain of geometry processing. We are currently investigating the construction of Voronoi diagrams on surfaces described by point clouds structured with our representation. We already obtained promising preliminary results, which allows us to get surface meshes from these diagrams. Globally, our structure can serve as a support for the definition of many functions over the point clouds.

In the future, the current data structure could be improved to reduce its storage cost. During this work, given the size of data we manipulated, our main objective was to control the memory consumption, but no effort has been done to get a compact structure of the local graphs, and their interconnections.

Technically, our structuration workflow could be also improved. For instance, the management of occlusions (Section II-B) follows quite a simple approach that could be improved by modeling the surface locally, to abstract the interconnection step from the sampling density of the point cloud. Also, our experiments showed that the structuration of the point clouds may be long, especially when large depth maps are cut into too small tiles. Interconnecting the graphs is currently the most time-consuming step of our structuration algorithm. However, nothing restrains the possibility of parallelizing the search of matching points on several graphs at the same time, which could reduce the computing time of this step by a factor close to the number of threads used. Overall, the I/O strategy used could be improved to minimize the I/O exchanges, which can be impactful when the number of local graphs is important.

Lastly, our datasets during this study were LiDAR scans providing depth maps, but we believe that the proposed approach could be extended to unstructured acquisitions. Though, one could imagine constructing several local graphs, by defining overlapping spatially segmented areas of a point

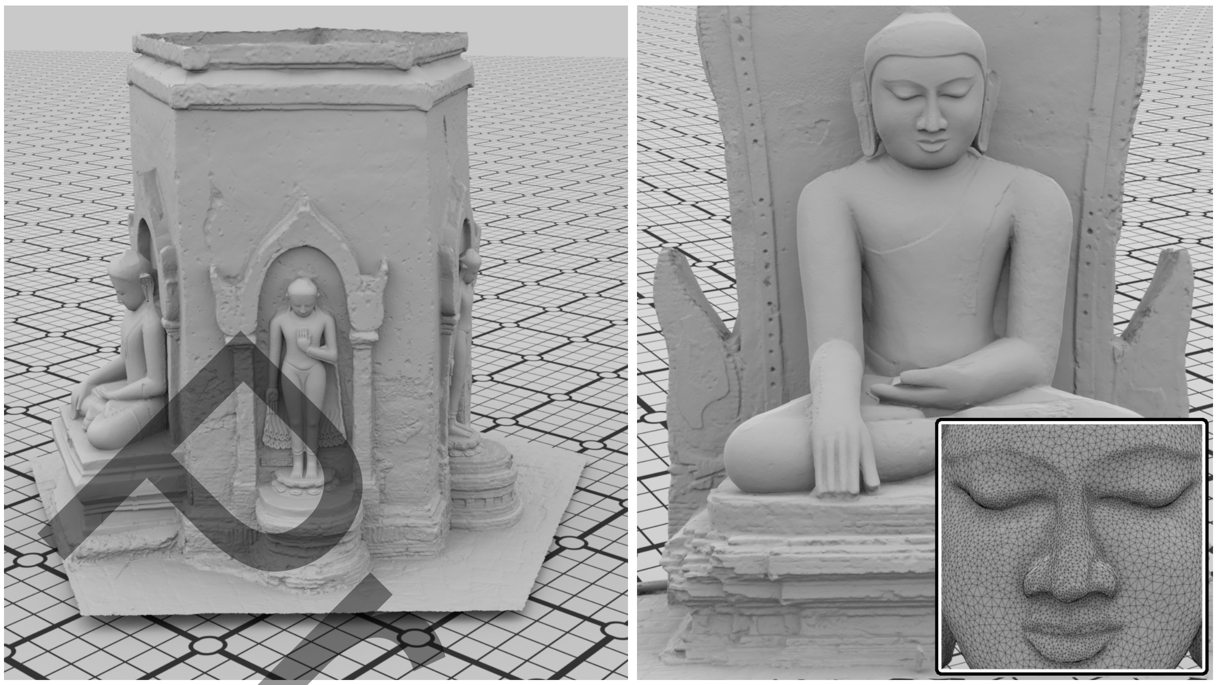


Fig. 17. Two other surfaces (parts of the sites EIM YA KYAUNG and KHAYMINGHA, from left to right, respectively) reconstructed with [28] from simplified point clouds generated with our curvature-aware resampling.

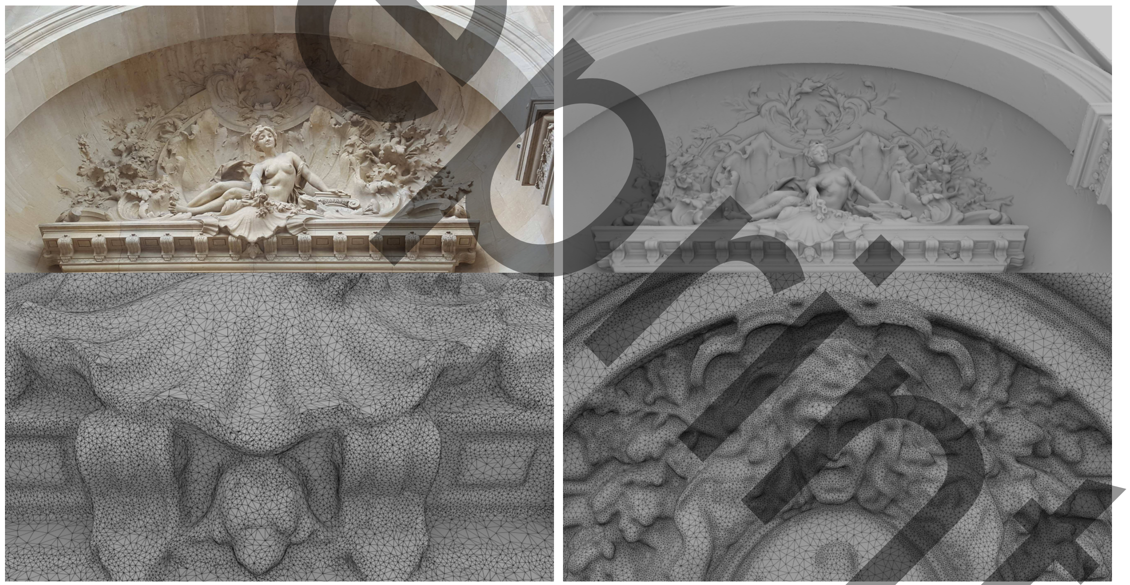


Fig. 18. Top-left: picture of a part of the facade of PALAIS (EXT.). Top-right: surface reconstructed with [28] from a point cloud generated with our curvature-aware resampling. Bottom: close-up views of the reconstructed surface.

cloud, in which all the points could be connected using an approach similar to [12].

VI. ACKNOWLEDGEMENTS

TEMPLO MAYOR, MEETING HOUSE, ANANDA OAK KYAUNG, WAT PHRA SI SANPHET, EIM YA KYAUNG and KHAYMINGHA are courtesy of the *CyArk/Google Open Heritage Program* [29]. PALAIS (EXT.) and PALAIS (INT.) are courtesy of *Art Graphique et Patrimoine*. FERTILITY and VENUS are courtesy of *AIM@SHAPE-VISIONAIR Shape Repository* [30]. ARMADILLO and DRAGON are courtesy of *The Stanford 3D scanning repository* [31].

REFERENCES

[1] A. Bletterer, F. Payan, M. Antonini, and A. Meftah, "Towards the reconstruction of wide historical sites: A local graph-based representation to resample gigantic acquisitions," in *16th EUROGRAPHICS Workshop on Graphics and Cultural Heritage (EG GCH)*, Vienna, Austria, Nov. 2018, pp. 1 – 9. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01875580

[2] ——, "Out-of-core Resampling of Gigantic Point Clouds," in *Symposium on Geometry Processing 2018- Posters*, T. Ju and A. Vaxman, Eds. The Eurographics Association, 2018.

[3] S. Rusinkiewicz and M. Levoy, “Qsplat: A multiresolution point rendering system for large meshes,” in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 343–352.

[4] C. Dachsbacher, C. Vogelgsang, and M. Stamminger, “Sequential point trees,” in *ACM Transactions on Graphics*, vol. 22, no. 3. ACM, 2003, pp. 657–662.

[5] J. Ryde and H. Hu, “3D mapping with multi-resolution occupied voxel lists,” *Autonomous Robots*, vol. 28, no. 2, p. 169, 2010.

[6] C. Loop, C. Zhang, and Z. Zhang, “Real-time high-resolution sparse voxelization with application to image-based modeling,” in *Proceedings of the 5th High-Performance Graphics Conference*. ACM, 2013, pp. 73–79.

[7] J. Peng and C. J. Kuo, “Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition,” *ACM Trans. Graph.*, vol. 24, no. 3, pp. 609–616, 2005.

[8] R. Schnabel and R. Klein, “Octree-based point-cloud compression.” in *Spbg*, 2006, pp. 111–120.

[9] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, “Octomap: An efficient probabilistic 3D mapping framework based on octrees,” *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[10] J. Elseberg, D. Borrmann, and A. Nüchter, “One billion points in the cloud–an octree for efficient processing of 3D laser scans,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 76, pp. 76–88, 2013.

[11] D. Thanou, P. A. Chou, and P. Frossard, “Graph-based compression of dynamic 3D point cloud sequences,” *IEEE Trans. Image Processing*, September 2015.

[12] S. Chen, D. Tian, C. Feng, A. Vetro, and J. Kovačević, “Fast resampling of three-dimensional point clouds via graphs,” *IEEE Transactions on Signal Processing*, vol. 66, no. 3, pp. 666–681, 2018.

[13] B. Bellekens, V. Spruyt, R. Berkvens, and M. Weyn, “A survey of rigid 3D pointcloud registration algorithms,” in *AMBIENT 2014: the Fourth International Conference on Ambient Computing, Applications, Services and Technologies, August 24-28, 2014, Rome, Italy*, 2014, pp. 8–13.

[14] N. Pietroni, M. Tarini, O. Sorkine, and D. Zorin, “Global parametrization of range image sets,” *ACM Trans. Graph.*, vol. 30, no. 6, p. 1–10, Dec. 2011. [Online]. Available: https://doi.org/10.1145/2070781.2024183

[15] J.-F. Rivest, P. Soille, and S. Beucher, “Morphological gradients.” *J. Electronic Imaging*, vol. 2, no. 4, pp. 326–336, 1993.

[16] P. Biasutti, A. Bugeau, J.-F. Aujol, and M. Brédif, “Visibility estimation in point clouds with variable density,” in *International Conference on Computer Vision Theory and Applications (VISAPP)*, ser. Proceedings of the 14th International Conference on Computer Vision Theory and Applications, Prague, Czech Republic, Feb. 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01812061

[17] R. L. Cook, “Stochastic sampling in computer graphics,” *ACM Trans. Graph.*, vol. 5, no. 1, pp. 51–72, Jan. 1986. [Online]. Available: http://doi.acm.org/10.1145/7529.8927

[18] Y. Fu and B. Zhou, “Direct sampling on surfaces for high quality remeshing,” in *Proceedings of the 2008 ACM Symposium on Solid and Physical Modeling*, ser. SPM ’08. New York, NY, USA: Association for Computing Machinery, 2008, p. 115–124. [Online]. Available: https://doi.org/10.1145/1364901.1364919

[19] D. Cline, S. Jeschke, K. White, A. Razdan, and P. Wonka, “Dart throwing on surfaces,” in *Computer Graphics Forum*, vol. 28, no. 4. Wiley Online Library, 2009, pp. 1217–1226.

[20] J. Bowers, R. Wang, L.-Y. Wei, and D. Maletz, “Parallel poisson disk sampling with spectrum analysis on surfaces,” *ACM Transactions on Graphics (TOG)*, vol. 29, no. 6, p. 166, 2010.

[21] Y. Xu, R. Hu, C. Gotsman, and L. Liu, “Blue noise sampling of surfaces,” *Computers & Graphics*, vol. 36, no. 4, pp. 232–240, 2012.

[22] M. Corsini, P. Cignoni, and R. Scopigno, “Efficient and flexible sampling with blue noise properties of triangular meshes,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 6, pp. 914–924, 2012.

[23] J.-L. Peyrot, F. Payan, and M. Antonini, “Direct blue noise resampling of meshes of arbitrary topology,” *The Visual Computer*, vol. 31, no. 10, pp. 1365–1381, October 2015.

[24] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959. [Online]. Available: http://dx.doi.org/10.1007/BF01386390

[25] L.-Y. Wei and R. Wang, “Differential domain analysis for non-uniform sampling,” *ACM Trans. Graph.*, vol. 30, pp. 50:1–50:10, 2011.

[26] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand, “A gentle introduction to bilateral filtering and its applications,” in *ACM SIGGRAPH 2007 courses*. ACM, 2007, p. 1.

[27] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, “MeshLab: an Open-Source Mesh Processing Tool,” in *Eurographics Italian Chapter Conference*, V. Scarano, R. D. Chiara, and U. Erra, Eds. The Eurographics Association, 2008.

[28] D. Boltcheva and B. Levy, “Surface reconstruction by computing restricted voronoi cells in parallel,” *Computer-Aided Design*, vol. 90, pp. 123–134, 2017.

[29] Google/CyArk. Open heritage. [Online]. Available: https://artsandculture.google.com/project/cyark

[30] AIM@SHAPE-VISIONAIR. Aim@shape-visionair shape repository. [Online]. Available: http://visionair.ge.imati.cnr.it

[31] M. Levoy, J. Gerth, B. Curless, and K. Pull. The stanford 3D scanning repository. [Online]. Available: http://graphics.stanford.edu/data/3Dscanrep

Arnaud Bletterer Arnaud Bletterer received the B.S. and M.S. degrees in Computer Science from the University of Strasbourg, France, in 2012 and 2014, respectively, and the Ph.D degree in Signal and Image Processing from Université Côte d’Azur, France, in 2018. The topic of his Ph.D was about the surface reconstruction from gigantic point clouds. He is now a research engineer in QuantifiCare, France. His research interests include geometry processing, statistical modelling and machine learning.

Frédéric Payan received the Ph.D degree in signal and image processing in December 2004 from the University of Nice-Sophia Antipolis (France). After a one-year postdoctoral research fellow (2005-2006) at the Laboratory LMC-IMAG (CNRS) at Grenoble (France), he became associate professor in 2006 at the University of Nice-Sophia Antipolis (become Université Côte d’Azur), and joined the laboratory I3S (CNRS). Today, his research interests are focused on the geometry processing notably in the context of acquisition of 3D objects (surface resampling, reconstruction, data visualization...) but also on the compression of volume meshes in geosciences. He is co-author of 60 publications, 1 book chapter, and 2 patents. He supervised 3 Ph.D students, and is currently supervising 2 Ph.D students and 1 research engineer.

Marc Antonini is research director at CNRS since 2004. Author of more than 240 papers, 7 book chapters and 12 patents, his research activities cover image, video and geometry coding. He is currently working on bio-inspired applications for image and video compression, and methods for archival storage on synthetic DNA. His works on wavelet transform have been included in JPEG2000. From 1995 to 2001, he was involved with CNES Toulouse in the Earth Observation program Pleiades for the definition of the on-board image coder. He is a member IEEE, Associate Editor for Eurasip JIVP, and member of IEEE MMSP Technical Commitee. He is a co-founder of Cintoo, a start-up created in 2013. He supervised 27 Ph.D students and is currently supervising 4 Ph.D students.