A Dynamic Surface Reconstruction Framework for Large Unstructured Point Sets

Rémi Allègre, Raphaëlle Chaine and Samir Akkouche
LIRIS CNRS / Université Lyon 1, Villeurbanne, France

Abstract

We present a method to reconstruct simplified mesh surfaces from large unstructured point sets, extending recent work on dynamic surface reconstruction. The method consists of two core components: an efficient selective reconstruction algorithm, based on geometric convection, that simplifies the input point set while reconstructing a surface, and a local update algorithm that dynamically refines or coarsens the reconstructed surface according to specific local sampling constraints.

We introduce a new data-structure that significantly accelerates the original selective reconstruction algorithm and makes it possible to handle point set models with millions of sample points. Our data-structure mixes a kd-tree with the Delaunay triangulation of the selected points enriched with a sparse subset of landmark sample points. This design efficiently responds to the specific spatial location issues of the geometric convection algorithm. We also develop an out-of-core implementation of the method, that permits to seamlessly reconstruct and interactively update simplified mesh surfaces from point sets that do not fit into main memory.

Keywords: Surface reconstruction, geometric convection, point set simplification, dynamic level of detail update, out-of-core reconstruction.

1 Introduction

The recent advances in 3D scanning technologies have led to an increasing need for techniques capable of processing massive discrete geometric data. In the last years, a great deal of work has been carried out on surface reconstruction from datasets with millions of sample points, including unorganized points sets [BMR*99, DGH01b, OBA*03, OBS05] and sets of range images [LPC*00, RCG*04]. These methods are often used to produce a triangulated mesh surface, which is a standard representation for fast visualization and geometry processing algorithms. However, the data used to generate these meshes are generally overly dense, due to uniform grid sampling patterns, and a mesh simplification step is required for use in common applications.

Point set simplification techniques offer an alternative to the standard pipeline by introducing a simplification step before the reconstruction process. The former aim at reducing the redundancy of the input data in order to accelerate subsequent reconstruction or visualization. Subsampling algorithms decimate the point set [DGH01a, Lin01, WK04] while resampling algorithms compute new point locations [DGH01c, PGK02, MD04]. These techniques rely either on oriented normals and local connectivity information obtained from k-neighborhoods, or on a global Delaunay triangulation or Voronoï diagram, which represents a significant part of a surface reconstruction process that would take all the points into account.

Several algorithms that perform reconstruction and simplification in a single framework have been recently studied. Boissonnat and Cazals [BC01] have proposed a Delaunay-based coarse-to-fine reconstruction algorithm controlled by a signed distance function to an implicit surface. Ohtake et al. [OBS05] have developed an algorithm that resamples a point set using a quadric error metric, coupled with a specific fast local triangulation procedure. In both cases, the resulting sampling remains static, and the reconstructed surface cannot be easily updated, especially if the level of detail needs to be modified afterwards, or if additional data become available later (e.g. when streaming data on a network, or during a digital acquisition project). Allègre et al. [ACA05] have tackled this limitation by devising a *dynamic surface reconstruction* framework in which the reconstruction becomes *selective* and *evolutive*. The originality of their approach is to integrate surface reconstruction, data simplification and dynamic level of detail update into a single framework. Starting from a dense unorganized input point set, the authors reconstruct a simplified triangulated surface by means of a Delaunay-based surface reconstruction algorithm called *geometric convection* [Cha03] coupled with a local point set subsampling procedure. The Delaunay triangulation is constructed only for the retained sample points in order to maintain some history of the

reconstruction process. The reconstructed surface can then be easily updated by inserting or removing sample points without restarting the reconstruction process from scratch. However, the method lacks an efficient data-structure to handle large data.

In this paper, we extend the dynamic surface reconstruction framework proposed in [ACA05] to handle large data efficiently (Fig. 1). We introduce a new data-structure that significantly accelerates the original selective reconstruction algorithm and makes it possible to handle point set models with millions of sample points. Our data-structure mixes a kd-tree with the Delaunay triangulation of the selected points enriched with a subset of landmark sample points obtained from the kd-tree. This design efficiently responds to the specific spatial location issues of the geometric convection algorithm, while being much less expensive than maintaining a global Delaunay triangulation. We also develop an out-of-core implementation of the method to reconstruct simplified mesh surfaces from point sets that do not fit into main memory. Our method involves neither stitching nor consistent orientation issues. We demonstrate the effectiveness of our framework on various detailed scanned statues with several millions of sample points. Our method can reconstruct high-quality simplified triangulated surfaces in a few minutes. Geometric detail can then be recovered or reduced locally whenever needed in a few seconds. The method can be useful for viewpoint-dependent surface reconstruction.

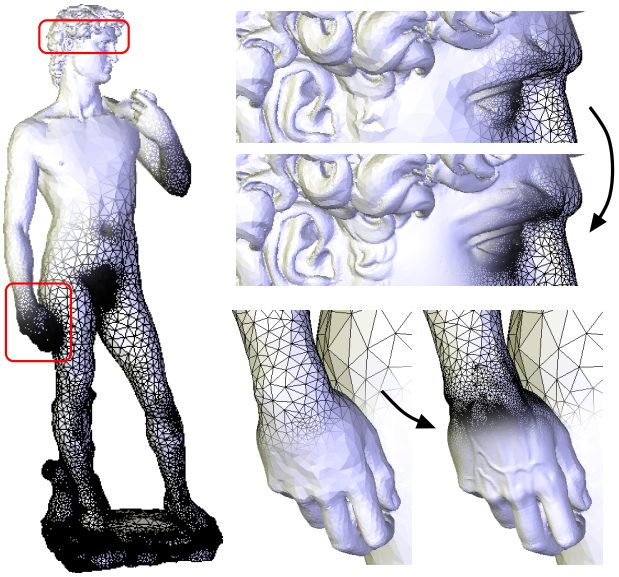

Figure 1: Dynamic surface reconstruction from a large point set model: the DAVID (3.6M points). A simplified mesh has been first reconstructed (left, 137k points, 4 minutes on Pentium IV 3GHz). Then, the result has been locally refined on the right temple and hand (right, 175k points, 28 seconds).

2 Background

In this section, we briefly review the classic geometric convection algorithm described in [Cha03] and its embedding into a dynamic framework with simplification and update abilities, as developed in [ACA05]. We focus on the geometric predicates and queries involved in the surface reconstruction algorithm, as well as on their evaluation.

2.1 Geometric convection

The geometric convection algorithm is a surface reconstruction algorithm that proceeds by filtering the Delaunay triangulation of an input point set sampled from a smooth surface [CG04]. This method has some similarities with the Wrap [Ede02] and Flow Complex [GJ03] techniques. The filtration is guided by a convection scheme related to level set methods [ZOF01] that consists in shrinking an enclosing surface under the influence of the gradient field of a distance function to the closest sample point. This process results in a closed, oriented triangulated surface embedded in the Delaunay triangulation of the point set, and characterized by an *oriented Gabriel property* [Cha03]. This means that for every facet, the diametral half-ball located inside the surface, or *Gabriel half-ball*, contains no sample point.

Let $P \subset \mathbb{R}^3$ denote the input point set and $\hat{S}$ the surface in convection. The convection scheme can be completely achieved through the Delaunay triangulation of P by removing the facets that do not meet the oriented Gabriel property through an iterative sculpting process that starts from the convex hull. The $\hat{S}$ surface is a closed triangulated surface is maintained at every step, all the facets oriented *inwards*, and two self-intersecting facets can collapse locally, which may change its topology. A local study (or a more global solution) is required to dig into *pockets* that may locally block the convection scheme, e.g. based on local granularity. The algorithm is illustrated on a 2D point set in Figure 2.

The geometric evolution of $\hat{S}$ along the convection process is locally guided by a geometric predicate $\mathbf{P}_{\mathsf{og}}$ and a geometric query $\mathbf{Q}_{\mathsf{dt}}$ defined as follows:

($\mathbf{P}_{\mathsf{og}}$) Given an oriented Delaunay facet **pqr**, test whether it satisfies the oriented Gabriel property.

($\mathbf{Q}_{\mathsf{dt}}$) Given an oriented Delaunay facet **pqr**, find the point $\mathbf{s} \in P$ such that **pqrs** forms a Delaunay tetrahedron enclosed in the half-space above the facet.

Assuming that the Delaunay triangulation of the input point set has been constructed, $\mathbf{P}_{\mathsf{og}}$ and $\mathbf{Q}_{\mathsf{dt}}$ are both evaluated in constant time, and the overall complexity of the algorithm is linear in the number of Delaunay cells traversed by the surface.

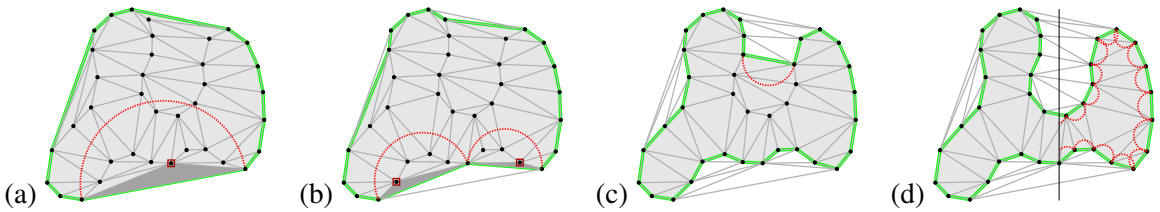

Figure 2: Geometric convection towards a 2D point set. In (a), an enclosing curve is initialized on the convex hull of the point set. The current edge, enclosed by a non-empty Gabriel half-ball, forms a Delaunay triangle (dark gray) with the square point. This triangle becomes external, the curve is updated (b), and it continues to shrink. In (c), an edge is found to block a pocket; it will be forced. The final result is shown in (d) with some empty Gabriel half-balls.

2.2 Selective reconstruction

In presence of an overly dense input point set, the purpose of selective reconstruction is to produce a simplified triangulated surface that remains close from the sampled one. An interesting property of the geometric convection algorithm is that it induces a breadth-first discovery of neighboring sample points on the sampled surface. Exploiting this property, the algorithm described in [ACA05] associates the geometric convection algorithm with a local subsampling procedure that removes sample points that are not geometrically significant, up to an error tolerance, while reconstructing. This process results in a sampling distribution that is locally uniform almost everywhere.

Each time a new sample point $\mathbf{p} \in P$ is incorporated into the surface in convection $\hat{S}$, its nearest neighbors in P that do not already belong to $\hat{S}$ are successively removed while their distance to $\mathbf{p}$ does not exceed a radius $r = \min(r_{geom}, \alpha.r_{topo})$ with r_{geom} and r_{topo} defined as follows:

- r_{geom} is the distance from $\mathbf{p}$ to its first nearest neighbor $\mathbf{p}_i$ in P that does not satisfy:

$$|\mathrm{n}(\mathbf{p}_i) \cdot \mathrm{n}(\mathbf{p})| \geq \rho_{geom} \qquad \rho_{geom} \in [0,1]$$

- r_{topo} is the distance from $\mathbf{p}$ to its first nearest neighbor $\mathbf{p}_j$ in P that does not satisfy:

$$|\mathrm{n}(\mathbf{p}_j) \cdot \frac{\mathbf{p} - \mathbf{p}_j}{\|\mathbf{p} - \mathbf{p}_j\|}| \leq \rho_{topo} \qquad \rho_{topo} \in [0,1]$$

where $\mathrm{n}(\mathbf{x})$ denotes the unit normal vector at a point $\mathbf{x} \in P$. The ρ_{geom} value limits the normal deviation from $\mathrm{n}(\mathbf{p})$ and controls the level of detail of the reconstruction, whereas the ρ_{topo} value restricts the decimated neighborhood to a topological disk. Note that only the normal directions are required, not their orientation. If they are not supplied as part of the input data, these normal directions can be locally estimated by covariance analysis. We introduce the α factor, that was not present in [ACA05], in order to achieve high-quality simplification without a fairing step. Decimating with $\alpha = 1$ results in skinny triangles near sharp features, where the sampling density increases too rapidly. To obtain a smooth density gradient, we multiply the r radius by a factor $\alpha = 0.5$, which reflects the distance to the medial axis.

In addition to the previously defined $\mathbf{P}_{\mathsf{og}}$ predicate and $\mathbf{Q}_{\mathsf{dt}}$ query, the selective reconstruction algorithm requires a query $\mathbf{Q}_{\mathsf{nn}}$ that returns the nearest neighbors of a sample point. If $\hat{R} \subset P$ denotes the set of removed sample points at a given time, then $\mathbf{P}_{\mathsf{og}}$ and $\mathbf{Q}_{\mathsf{dt}}$ are evaluated within $P \setminus \hat{R}$ at that time. Since many sample points may be discarded, constructing the Delaunay triangulation of the whole input point set may be uselessly expensive. The authors instead rely on a triangulated surface data-structure that is initialized by computing a simplified convex hull based on the above subsampling procedure. The $\mathbf{P}_{\mathsf{og}}$ predicate and the $\mathbf{Q}_{\mathsf{dt}}$ query are evaluated on-the-fly during the reconstruction process, but not in a direct manner. Evaluating $\mathbf{P}_{\mathsf{og}}$ now involves an additional query $\mathbf{Q}_{\mathsf{hb}}$ that reports the points in $P \setminus \hat{R}$ located inside the Gabriel half-ball of the facet. The $\mathbf{Q}_{\mathsf{dt}}$ query also needs a predicate $\mathbf{P}_{\mathsf{ct}}$ to test whether a point in $P \setminus \hat{R}$ enters in conflict with a Delaunay tetrahedron.

The question of how to efficiently evaluate $\mathbf{P}_{\mathsf{og}}$, $\mathbf{Q}_{\mathsf{dt}}$, and $\mathbf{Q}_{\mathsf{nn}}$ arises. A kd-tree data-structure is well-suited to report the sample points located inside the Gabriel half-ball of a facet and to search the nearest neighbors of a sample point. For a facet that does not satisfy $\mathbf{P}_{\mathsf{og}}$, the search space for $\mathbf{Q}_{\mathsf{dt}}$ can be reduced to its Gabriel half-ball. However, these half-balls may contain a great part of the input point set, especially at the beginning of the reconstruction process (see Figure 2(a) for example). Moreover, when $\mathbf{P}_{\mathsf{og}}$ is satisfied and that a pocket is detected, the search space for $\mathbf{Q}_{\mathsf{dt}}$ can extend to the whole half-space above the facet.

The main limitation of the algorithm regarding performance is the lack of visibility of "what lies ahead" in the unexplored domain during the convection process. To handle large data efficiently, a better localization of geometric queries is required. In Section 3, we

show that performance can be considerably improved by maintaining a partial Delaunay triangulation of the data in a dynamic fashion.

2.3 Local update

During the reconstruction process, constructing the Delaunay triangulation of the *retained* sample points makes it possible to locally update the reconstructed surface by adding or removing sample points in a dynamic fashion. This functionality takes advantage of the *discovering relation* induced by the convection scheme on the set of Delaunay cells traversed by the surface. This relation is stored in the cells that have been visited. When inserting sample points, these points enter in conflict with a set of Delaunay cells that form a conflict region. This region is retriangulated and the discovering relation between Delaunay cells is restored by restarting the reconstruction process from its boundary parts located out of the current surface. The surface can step back locally when some cells cannot be discovered anymore. When removing sample points, the cells of the conflict region are the cells attached to the points to be removed.

To locally change the level of detail of a reconstructed surface, a region of interest and a local ρ_{geom} value are first defined. All the removed sample points in this region are rehabilitated, and the Delaunay cells whose circumsphere intersects the region of interest form the conflict region. The internal Delaunay vertices are removed and the selective reconstruction process restarts as described above, by taking account of the local simplification parameter.

3 Selective reconstruction from large point sets

This section describes our extensions to the selective reconstruction algorithm presented in [ACA05], that improve its performance and makes it appropriate for large datasets. Our first goal is to accelerate the evaluation of the previously mentioned $\mathbf{P}_{\mathsf{og}}$ predicate and $\mathbf{Q}_{\mathsf{dt}}$ query. This is achieved by first structuring and reducing the search space covered by these operations, based on a partial Delaunay triangulation of the input point set. Spatial search is then performed through a kd-tree data-structure with an optimized algorithm. At the end of the section, we describe an out-of-core selective reconstruction algorithm that mixes the in-core technique with the local update algorithm to handle point sets that do not fit into memory.

3.1 Data-structure and accelerated algorithm

Data-structure In Section 2.2, we highlighted a front visibility issue when running the reconstruction process without a global Delaunay triangulation. Suppose now that we incrementally construct the Delaunay triangulation of the retained sample points while reconstructing, as it is done in anticipation of local update. Every facet of the shrinking surface is the interface between two cells; we call *front cell* the one that is enclosed in the surface. Front cells are connected to opposite vertices on the surface and give information about the extent of the unexplored domain. However, the part of their circumsphere located inside the surface generally encloses a larger spatial domain than the Gabriel half-balls. The Delaunay triangulation of the retained sample points is therefore not sufficient to reduce the search space for the spatial queries involved in the convection algorithm: additional sample points are required to "break" big front cells.

We begin with a set of landmark sample points obtained from a kd-tree structure with a threshold on the maximum number of points per leaf. In every leaf cell, the point that is the closest from the centroid is retained as a landmark (their density will be discussed later). The Delaunay triangulation $\hat{D}$ of these points is then built, and enriched with the corners of an axis-aligned bounding box. The surface is initialized on the convex hull, the bounding box here, and is directly supported by the Delaunay triangulation $\hat{D}$. The reconstruction process can then be run benefiting from *smaller* front cells that will help to accelerate the evaluation of both $\mathbf{P}_{\mathsf{og}}$ and $\mathbf{Q}_{\mathsf{dt}}$. In parallel, spatial search is delegated to a kd-tree data-structure that stores the whole input point set, with a specific algorithm that will be described later.

Accelerated algorithm The accelerated algorithm dynamically updates the Delaunay triangulation $\hat{D}$ all along the reconstruction process by inserting retained sample points and removing unretained landmarks so that the latter do not affect the final result. We exploit here the property that the Delaunay cells that become external to the surface remain until the end of the process, which is not the case for internal cells. External cells are naturally protected from any subsequent vertex insertion or removal.

We continue with the notations of Section 2 to describe the algorithm. An illustration in 2D is provided in Figure 3. To check whether an oriented facet **pqr** of the surface $\hat{S}$ satisfies $\mathbf{P}_{\mathsf{og}}$, we consider its front cell σ in the current Delaunay triangulation; its circumsphere is denoted as $\mathcal{S}$. We call *front vertex* the vertex of σ that is opposite to the facet; its location is denoted as **s**. The Gabriel half-ball of the facet is finally denoted as

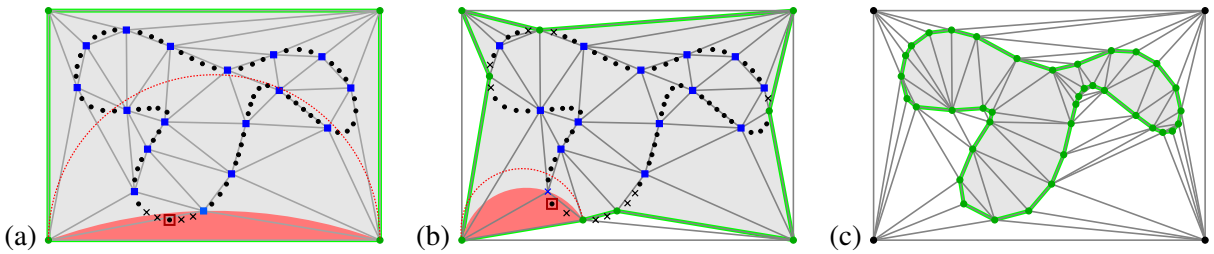

Figure 3: Accelerated selective reconstruction towards a 2D point set. In (a), an enclosing curve is initialized on a bounding box. A Delaunay triangulation has been built from its corners and a set of landmarks (filled square points). The current edge is enclosed by a non-empty Gabriel half-ball: the front vertex lies inside. Then, the point that forms a Delaunay triangle with the edge is searched within the disk that circumscribes the front cell (the square-dot point). The point set is then locally decimated around the retained point (cross points). In (b), the retained point has been inserted in the triangulation and the curve continues to shrink. The facets attached to the corners of the bounding box are forced. The final result is shown in (c).

$\mathcal{B}$, and the half-space above the plane that supports it as $\mathcal{H}$. The first step to evaluate $\mathbf{P_{og}}$ is to check whether $\mathbf{s}$ lies inside or outside $\mathcal{B}$.

1. If $\mathbf{s} \in \mathcal{B}$, then the $\mathbf{P_{og}}$ predicate is not satisfied. The $\mathbf{Q_{dt}}$ query is then performed in $(P \setminus \hat{R}) \cap \mathcal{S} \cap \mathcal{H}$, that corresponds to the set of points that enter in conflict with σ. If this set is empty, then **pqrs** forms a Delaunay tetrahedron in $P \setminus \hat{R}$.
2. If $\mathbf{s} \notin \mathcal{B}$, it it not guaranteed that $\mathbf{P_{og}}$ is satisfied. To evaluate the predicate, we first get all the points in the set $(P \setminus \hat{R}) \cap \mathcal{B}$ through $\mathbf{Q_{hb}}$. If this set is not empty, then $\mathbf{P_{og}}$ is not satisfied and $\mathbf{Q_{dt}}$ is next performed in the set $(P \setminus \hat{R}) \cap \mathcal{B}$.

In the case where $\mathbf{P_{og}}$ is satisfied but that a pocket is detected, then $\mathbf{Q_{dt}}$ is performed in the set of points $(P \setminus \hat{R}) \cap \mathcal{S} \cap \mathcal{H}$ that enter in conflict with σ. If this set is empty, then **pqrs** forms a Delaunay tetrahedron in $P \setminus \hat{R}$.

Each time a new Delaunay tetrahedron is formed from a facet **pqr** and a point **x**, then **x** is inserted into the Delaunay triangulation provided $\mathbf{x} \neq \mathbf{s}$, and the surface is updated. Note that any facet attached to some vertices of the bounding box should be opened, i.e. the query $\mathbf{Q_{dt}}$ performed, even when the predicate $\mathbf{P_{og}}$ is satisfied.

We now discuss the choice of the landmark points. The main benefit of these points is at the beginning of the process, where Gabriel half-balls may contain a lot of sample points. As their size decreases, this benefit also diminishes, because the density of these points becomes insufficient. However, small Gabriel half-balls can be processed more efficiently. If this density is too high, then much time may be spent to remove undesired landmarks. As the final simplification rate highly depends on the shape and on the ρ_{geom} value, the optimal number of landmarks is not easy to determine. In practice, choosing one landmark for a few thousands points (between 1k and 10k) is sufficient to limit the spheres that circumscribe front cells to a few hundreds points in the worst case, and get a significant acceleration of the selective reconstruction process.

3.2 Accelerated spatial search

In the accelerated selective reconstruction algorithm, the $\mathbf{P_{og}}$ predicate is first evaluated by localizing subsets of sample points that enter in conflict with front cells or that fall into Gabriel half-spheres. When the returned set of points is not empty for a given facet, then the point that forms a Delaunay tetrahedron with the facet has to be found ($\mathbf{Q_{dt}}$). Without information about the structure of the input point set, every point in this set is a potential candidate and thus needs to be tested. To reduce the number of tests, we rely on a kd-tree data-structure.

We first focus on the simple case where a facet of the surface is such that its front vertex is located inside the Gabriel half-ball of the facet. We start by searching for the non-empty leaves of the kd-tree that are likely to contain points that fall into the region bounded by the circumsphere of the front cell, restricted to the half-space defined by the facet; we call $\mathcal{C}$ this region. This is achieved through a depth-first traversal of the kd-tree. If a kd-tree cell completely lies inside $\mathcal{C}$, then the leaves of the corresponding sub-tree are returned. The leaves that intersect $\mathcal{C}$ only partially are also returned. Testing whether a kd-tree cell intersects $\mathcal{C}$ involves two predicates: a sphere/box overlap test and an half-space/box overlap test [AM01]. A counter that gives the number of remaining points in a leaf avoids testing empty kd-tree cells.

When non-empty leaves are reported, the next goal is to obtain the point that forms a Delaunay tetrahedron with the facet, with an average complexity better than linear in the number of points contained in the leaves.

Our algorithm proceeds incrementally, starting with the sample point that maps to the front vertex of the facet as candidate. The set of leaves reported for the facet are stored in a queue denoted as L, and the facet is denoted as **pqr**.

1. While L contains more than one element:
   (a) Get one point in each kd-tree leaf of L that falls into $\mathcal{C}$, if existing. Let M denote this set of points.
   (b) Search M for the best point candidate **c**, that is the point such that the circumsphere of **pqrc** contains no other point of M, based on $\mathbf{P}_{\mathsf{ct}}$.
   (c) Remove from L the empty cells and the cells that do not enter in conflict with tetrahedron **pqrc**.
2. Search for the best candidate from the remaining points.

The case where the facet has its opposite vertex outside its Gabriel half-ball is treated in a similar fashion, except that conflicts are first tested within the reported leaves that intersect the Gabriel half-ball in order to determine whether the facet satisfies $\mathbf{P}_{\mathsf{og}}$. As soon as the predicate is found to be unsatisfied or if a pocket is detected, then the search is pursued in order to find the Delaunay candidate.

The method rapidly discards outlier leaves, i.e. that are the least likely to contain the good candidate. However, it is often difficult to decide between the remaining leaves, since the candidates can "jump" from a leaf cell to the other. When the number of remaining leaves stagnates, we stop the process and switch to linear search among the remaining points in order to avoid any computational overhead of testing conflicts between leaf cells and triangulation cells. In practice, the overall gain per facet is typically 10% to 20% of conflict tests between a point and a tetrahedron ($\mathbf{P}_{\mathsf{ct}}$).

3.3 Out-of-core reconstruction algorithm

Starting from a large and dense input point set that cannot be stored in main memory, our goal is to produce a simplified triangulated surface that fits into memory. A common strategy to simplify large unstructured meshes that cannot be entirely loaded into memory consists in partitioning the input data into clusters and then processing each one independently in-core [Lin00, CMRS03, CGG*04]. This strategy does not extend easily to surface reconstruction from large unorganized point sets. Since no connectivity information between the different parts is available, stitching and orientation issues arise [DGH01b]. In our framework, we propose to circumvent this problem by maintaining some kind of global connectivity information based on the Delaunay triangulation of a subset of representative points, and process each cluster independently through the local update algorithm. Our algorithm proceeds in three steps:

1. We filter the input point set P through a regular grid to obtain a subset of representative sample points P_{rep} and a partition of P into clusters $P_1 \cup P_2 \cup \ldots \cup P_n = P$.
2. We build the Delaunay triangulation of P_{rep} and run the classic geometric convection algorithm on this point set.
3. For every subset P_i, we load the points that it contains into memory and then locally refine the reconstruction in the corresponding region of space using the local update and selective reconstruction algorithms.

While partitioning the input point set in the first step, we want to quickly extract a reduced set of representative points giving an approximate idea of the global shape. This sample is then used in the second step to produce a coarse reconstruction. We do not need a precise downsampling because the interest is not for the reconstruction itself, but rather for the *discovering relation* between Delaunay cells that partition the whole data domain. This relation will be the basis for subsequent local reconstruction updates. Even if the initial reconstruction is not topologically correct and misses some small features, this will not affect the quality of the final result; errors will be automatically fixed by local updates. However, from a computational point of view, it is preferable to start these updates with a sufficiently precise reconstruction. Indeed, some major revisions of the surface may be expensive, both in time and

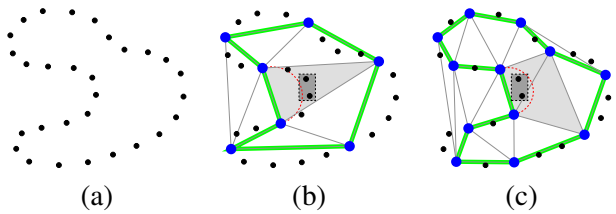

Figure 4: Accessible new data vs. inaccessible new data in 2D. We consider the input point set (a). In (b) and (c), the two new points in the rectangle are loaded. In (b), these points cannot be reached with the reconstruction (bold curve) from P_{rep} (bold points), whereas they can be reached in (c) where the reconstruction is finer.

memory. In order to limit them, the final surface should be *accessible* from the surface reconstructed from P_{rep} in the sense it should be enclosed in the union of Gabriel half-balls of the shrinking surface (Fig. 4). Note that this condition is not mandatory to get a correct reconstruction. In practice, we simply filter the input point set on a grid with a fixed resolution.

The initial filtering and partition step is achieved by reading the input point set three times. During the first pass, we compute the smallest axis-aligned bounding box, that we next subdivide into a regular grid. In the second pass, we compute for each non-empty grid cell the sample point that is the closest from the center. This set of sample points forms the set P_{rep}. During this pass, we also count how many points fall into each grid cell. We next define a recursive binary partition of the grid structure with a user-specified maximum number of sample points per leaf; each leaf cell represents a cluster P_i. The maximum population threshold for each cluster should be set according to the amount of memory available on the target machine. During the third pass, the points are distributed among the different leaf cells. Depending on their number, the content of the clusters may be written in separate files on disk, or they may be filled and processed one at a time, which requires additional reading passes.

In Step 3, we have to determine for each cluster P_i the set of cells of the current Delaunay triangulation that enter in conflict with its points. To avoid multiple point locations in the Delaunay triangulation, conflicts are tested against the smallest axis-aligned bounding box of the points in P_i. We search for the Delaunay cells whose circumsphere intersects this bounding box. This is achieved by first locating the Delaunay cell that contains the center of the box and then extending the conflict region by recursively testing the neighboring cells. The result is a connected set of Delaunay cells that is used to initialize the local reconstruction update process. For spatial search queries, we construct a kd-tree from all the points inside the conflict region. This set includes P_i and may also include some points outside P_i attached to Delaunay cells in conflict with the bounding box of P_i, which guarantees that the different refined parts correctly merge together. The local update process is then achieved as described in Section 2.3.

Two steps of the reconstruction process are illustrated on the LUCY model (14M points) in Figure 5. The boundaries of the different parts may be slightly perceivable in the final result. However, the method produces no discontinuity in the sampling density. These boundaries can be completely eliminated by simply enlarging the clusters so that they contain neighboring sample points up to a distance that depends on the simplification parameter ρ_{geom}.

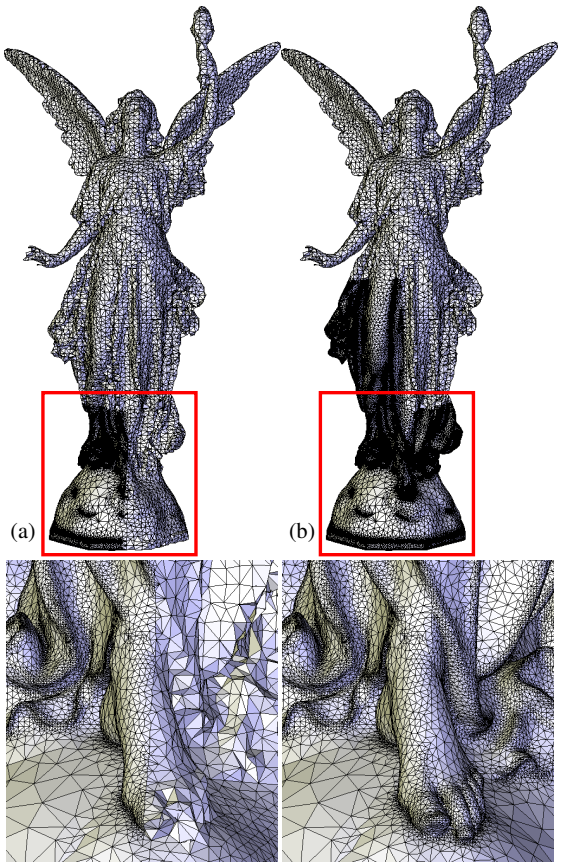

Figure 5: Two reconstruction steps for the LUCY model. In (a), the initial surface has been reconstructed from the representative points (25k) and a first local update step has been performed (bottom-left). In (b), two more clusters have been loaded and the reconstruction has been updated.

4 Experimental results and performance

We have implemented our extended dynamic surface reconstruction framework in C++ on a Linux platform using the Computational Geometry Algorithm Library, CGAL [CGA]. We make use of CGAL for constructing Delaunay triangulations and rely on filtered predicates for robust conflict tests.

We demonstrate the effectiveness of our framework on several large point set models that were obtained from laser range scanning (Figs. 1, 7, 8, 9, 10, 11). The LUCY and ST. MATTHEW models were reconstructed through our out-of-core selective algorithm. For both in-core and out-of-core reconstruction, the user has to provide a value for the error tolerance ρ_{geom}, that determines the level of detail. An initial selective reconstruction is performed, and the result can be next customized through local update features. We have devel-

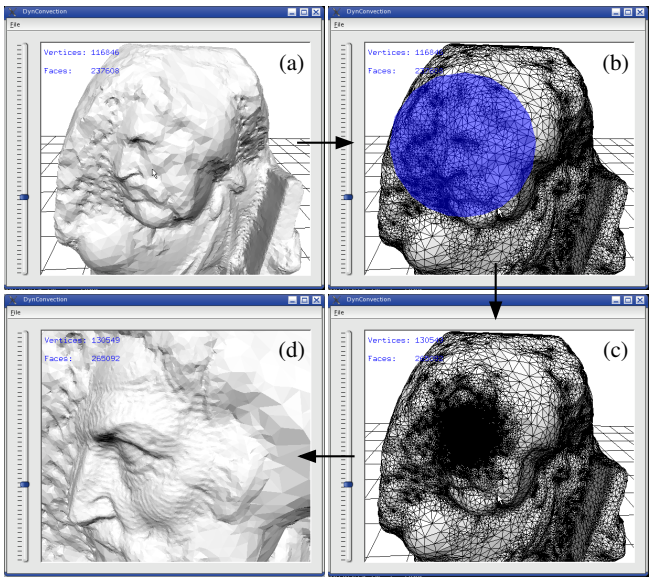

Figure 6: Some screenshots of our dynamic reconstruction interface. In this session, the face of the ST. MATTHEW model (original: 26M points) was refined. In (a), the result of an initial reconstruction has been loaded (time: 4 seconds). In (b), an update region has been selected and the reconstruction is shown refined in (c) and (d) (time: 11 seconds).

oped a graphic user interface (Fig. 6) to load a reconstructed simplified model from disk and interactively change its level of detail locally using the tool described in [ACA05]. Timings and memory usage for initial selective reconstructions as well as for local updates are reported in Table 1. All the results presented here were obtained on a Pentium IV 3.0GHz, 2GB RAM workstation. These timings include the preprocessing time required to build the kd-tree data-structure(s), select the representative and landmark sample points, and construct the initial Delaunay triangulation(s). Table 2 summarizes the overall execution profile for different in-core reconstructions.

Simplification performance The size of our initial simplified models is typically between 1% and 5% of the size of the original point set, which often suffices to preserve the shape of scanned objects at a mid-scale level, and even at fine scale if the point set is very redundant. The method is capable of producing high-quality simplified models directly, without the need of a subsequent mesh fairing step. The majority of the mesh vertices have valences between five and seven, and most facets have good aspect ratios.

Computational performance For in-core selective reconstructions, we set the number of landmark points to 1 for 2k sample points. The preprocessing time was less than 12 seconds in all our tests. According to our experiments, our accelerated selective reconstruction method runs up to *20 times faster* than the original one. The computational overhead due to update operations in the Delaunay triangulation is largely amortized by the reduction of spatial search domains. For out-of-core reconstructions, the LUCY model and ST. MATTHEW model were split so that each cluster contains less than 3.5M points; the resulting number of clusters was respectively 8 and 15. The number of initial representative points was set to 1 for 1k sample points and the initial reconstruction took less than 10 seconds in both cases. For each part, the refinement then took less than 2.5 minutes.

Execution profiles show that evaluating $\mathbf{P}_{\mathsf{og}}$ and $\mathbf{Q}_{\mathsf{dt}}$ is by far the most expensive task in the selective reconstruction algorithm. While we have reduced spatial search domains, the overall cost of spatial search queries still remains proportional to the number of facets through which the surface passes, which represents the bottleneck of the current method. Memory usage is also relatively high due to the storage of both a kd-tree and a Delaunay triangulation.

In the initial selective reconstruction step, our method runs slower than the surface reconstruction techniques with simplification proposed in [OBS05], and is also more memory demanding. However, our method can then perform localized updates at interactive rates, while the reconstructions in [OBS05] cannot evolve so easily. Our method does not involve stitching, and our results are guaranteed to be combinatorial manifolds. The dynamic approach is also powerful because it does not require to start from a well-behaved point sample, which is an advantage for out-of-core surface reconstruction, or even for streaming surface reconstruction.

Model name	BIMBA	ASIAN DRAGON	THAI STATUE
ρ_{geom}	0.8	0.65	0.65
Preprocessing	8.7	6.5	4.2
Evaluation of $\mathbf{P}_{\mathsf{og}}$ and $\mathbf{Q}_{\mathsf{dt}}$	54.9	66.8	74.0
Evaluation of $\mathbf{Q}_{\mathsf{nn}}$	30.7	20.2	13.3
Vertex insertion/removal	5.7	6.5	8.5

Table 2: Execution profile for three selective reconstructions. For each model, the column reports the percentages of overall time spent to accomplish the tasks listed on the left.

5 Conclusions and future work

We have proposed a new data-structure with a selective reconstruction algorithm that permits to efficiently

In-core reconstructions

Model		Selective reconstruction			Local update		Mem. usage
name	#points	ρ_{geom}	#points	time	#points	time	
BIMBA	1,873,832	0.65	31,643	0:47	–	–	380 MB
		0.8	57,630	1:01	–	–	395 MB
ASIAN DRAGON	3,609,600	0.65	185,504	2:48	177,324	0:14	778 MB
DAVID	3,617,008	0.6	137,025	2:06	174,628	0:28	754 MB
THAI STATUE	5,001,964	0.65	571,600	6:21	–	–	1,367 MB

Out-of-core reconstructions

Model		Selective reconstruction			Local update		Mem. usage
name	#points	ρ_{geom}	#points	time	#points	time	
LUCY	14,027,872	0.85	550,877	17:40	–	–	765 MB
ST. MATTHEW	26,034,562	0.5	116,846	31:12	130,549	0:11	836 MB

Table 1: Performance of our dynamic surface reconstruction framework for various input point sets. Computational timings are given in minutes:seconds and include preprocessing (construction of kd-trees and initial Delaunay triangulations). Memory usage corresponds to the maximum amount of memory used during the reconstructions, in megabytes. All tests were performed on a Pentium IV 3.0GHz, 2GB RAM workstation.

reconstruct simplified mesh surfaces from millions of sample points in a dynamic framework. The reconstructed surfaces can be dynamically refined or coarsened benefiting from the same data-structure. We have also proposed an out-of-core selective reconstruction algorithm scalable for input point sets that do not fit into memory.

Our method makes dynamic surface reconstruction practicable for large datasets obtained from laser range scanning, which may represent an alternative to the standard surface reconstruction-mesh simplification pipeline. The user can also completely customize the reconstruction in order to emphasize some particular details. When visualizing a large object, the precision of the reconstruction can be adapted to the viewpoint or to another region of interest at interactive rate. An efficient dynamic surface reconstruction framework may be also useful for processing point set streams on a network, since it does not require random access to the data.

In a near future, we plan to further improve the computational performance of our accelerated framework by reducing the number of spatial queries. Some information about conflicts could be shared between several facets in order to save some spatial queries. Another research direction would be to locally relax the global Delaunay property by choosing approximate candidates and repairing errors on-the-fly when needed. We are also investigating a way to extend the dynamic framework to reconstruct simplified surfaces in a streaming fashion.

6 Acknowledgments

This research is supported by the French Centre National de la Recherche Scientifique (CNRS) and by the Ministère de l'Éducation Nationale, de l'Enseignement Supérieur et de la Recherche. Point set models were provided courtesy of Standord Scanning Repository, Digital Michelangelo Project (Stanford), and AIM@Shape (IMATI and INRIA).

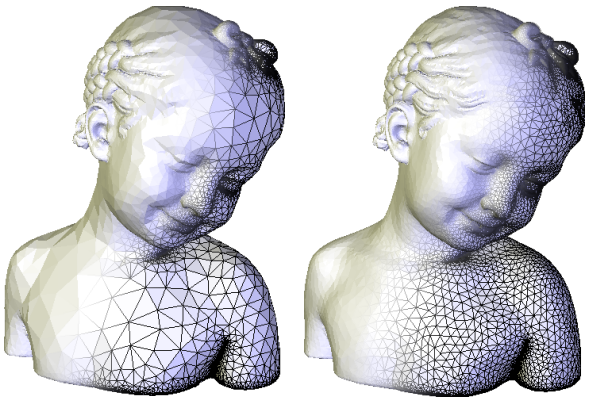

Figure 7: Reconstruction of the BIMBA model (1.9M points) with $\rho_{geom} = 0.65$ (left, 98% of points removed) and $\rho_{geom} = 0.8$ (right, 97% of points removed).

References

[ACA05] ALLÈGRE R., CHAINE R., AKKOUCHE S.: Convection-Driven Dynamic Surface Reconstruction. In *Proc. Shape Modeling*

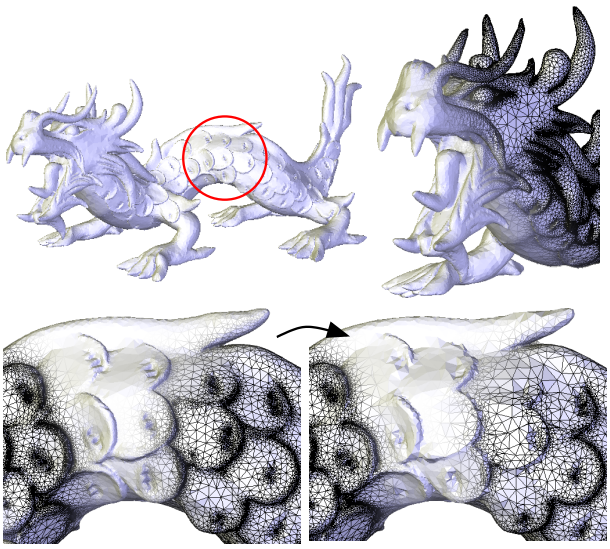

Figure 8: Reconstruction of the ASIAN DRAGON model (3.6M points) with $\rho_{geom} = 0.65$ (95% of points removed). The scales on the back have been coarsened in a second step.

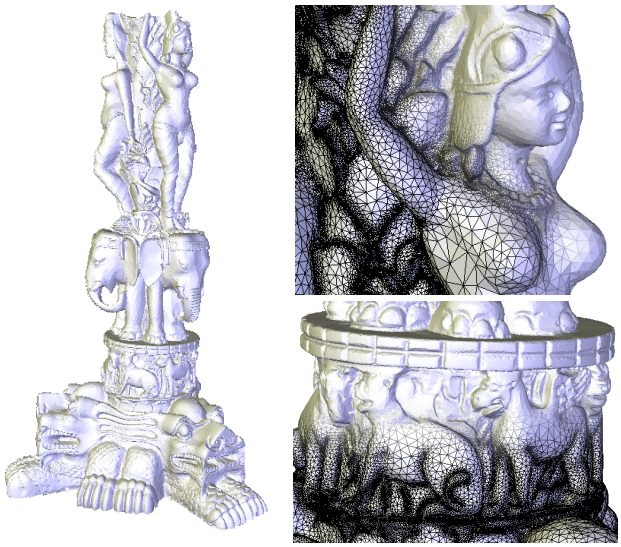

Figure 9: Reconstruction of the THAI STATUE model (5M points) with $\rho_{geom} = 0.65$ (89% of points removed).

Figure 10: Reconstruction of the LUCY model (14M points), with $\rho_{geom} = 0.85$ (96% of points removed).

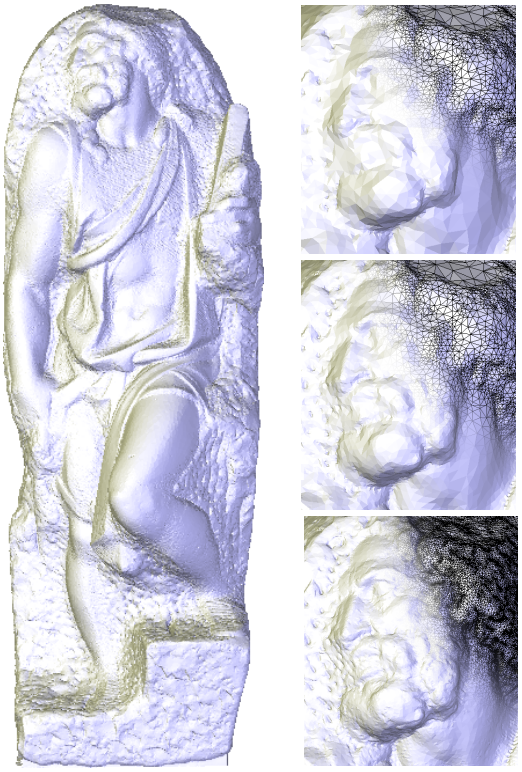

Figure 11: Reconstruction of the ST. MATTHEW model (26M points) at different levels of detail. Left: $\rho_{geom} = 0.8$; right, bottom-up: $\rho_{geom} = 0.8$ (1.2M points), $\rho_{geom} = 0.6$ (180k points), $\rho_{geom} = 0.5$ (117k points).

International (2005), IEEE Computer Society Press, pp. 33–42.

[AM01] AKENINE-MÖLLER T.: Fast 3D Triangle-Box Overlap Testing. *Journal of Graphics Tools 6*, 1 (2001), 29–33.

[BC01] BOISSONNAT J.-D., CAZALS F.: Coarse-to-fine surface simplification with geometric guarantees. In *Proc. Eurographics* (2001), pp. 490–499.

[BMR*99] BERNARDINI F., MITTLEMAN J., RUSHMEIER H., SILVA C., TAUBIN G.: The Ball-Pivoting Algorithm for Surface Reconstruction. *IEEE Transactions on Visualization and Computer Graphics 5*, 4 (1999), 349–359.

[CG04] CAZALS F., GIESEN J.: *Delaunay Triangulation based Surface Reconstruction: Ideas and Algorithms*. Tech. Rep. 5393,

INRIA, November 2004.

[CGA] http://www.cgal.org.

[CGG*04] CIGNONI P., GANOVELLI F., GOBBETTI E., MARTON F., PONCHIO F., SCOPIGNO R.: Adaptive TetraPuzzles – Efficient Out-of-core Construction and Visualization of Gigantic Polygonal Models. *ACM Transactions on Graphics (Proc. SIGGRAPH) 23*, 3 (2004), 796–803.

[Cha03] CHAINE R.: A geometric convection approach of 3-D reconstruction. In *Proc. Eurographics Symposium on Geometry Processing* (2003), pp. 218–229.

[CMRS03] CIGNONI P., MONTANI C., ROCCHINI C., SCOPIGNO R.: IEEE Transactions on Visualization and Computer Graphics. *External Memory Management and Simplification of Huge Meshes 9*, 4 (2003), 525–537.

[DGH01a] DEY T. K., GIESEN J., HUDSON J.: Decimating samples for mesh simplification. In *Proc. Canadian Conference on Computational Geometry* (2001), pp. 85–88.

[DGH01b] DEY T. K., GIESEN J., HUDSON J.: Delaunay-based shape reconstruction from large data. In *Proc. IEEE Symposium in Parallel and Large Data Visualization and Graphics* (2001), pp. 19–27.

[DGH01c] DEY T. K., GIESEN J., HUDSON J.: Sample shuffling for quality hierarchic surface meshing. In *Proc. 10th International Meshing Roundatble Conference* (2001), pp. 143–154.

[Ede02] EDELSBRUNNER H.: Surface reconstruction by wrapping finite point sets in space. In *Ricky Pollack and Eli Goodman Festscrift* (2002), Aronov B., Basu S., Pach J., M. Sharir S.-V., (Eds.), Springer-Verlag, pp. 379–404.

[GJ03] GIESEN J., JOHN M.: The Flow Complex: A Data Structure for Geometric Modeling. In *Proc. ACM-SIAM Symposium on Discrete Algorithms* (2003), pp. 285–294.

[Lin00] LINDSTROM P.: Out-of-core simplification of large polygonal models. In *Proc. SIGGRAPH* (2000), pp. 259–262.

[Lin01] LINSEN L.: *Point cloud representation*. Tech. Rep. 2001-3, Universität Karlsruhe, Germany, 2001.

[LPC*00] LEVOY M., PULLI K., CURLESS B., RUSINKIEWICZ S., KOLLER D., PEREIRA L., GINZTON M., ANDERSON S., DAVIS J., GINSBERG J., SHADE J., FULK D.: The Digital Michelangelo Project: 3D Scanning of Large Statues. In *Proc. SIGGRAPH* (2000), pp. 131–144.

[MD04] MOENNING C., DOGSON N. A.: Intrinsic point cloud simplification. In *Proc. GraphiCon* (2004).

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level Partition of Unity Implicits. *ACM Transactions on Graphics (Proc. SIGGRAPH) 22*, 3 (2003), 463–470.

[OBS05] OHTAKE Y., BELYAEV A. G., SEIDEL H.-P.: An integrating approach to meshing scattered point data. In *Proc. ACM Symposium on Solid and Physical Modeling* (2005), pp. 61–69.

[PGK02] PAULY M., GROSS M., KOBBELT L. P.: Efficient Simplification of Point-Sampled Surfaces. In *Proc. IEEE Visualization Conference* (2002), pp. 163–170.

[RCG*04] ROCCHINI C., CIGNONI P., GANOVELLI F., MONTANI C., PINGI P., SCOPIGNO R.: The Marching Intersections Algorithm for Merging Range Images. *The Visual Computer 20*, 2–3 (2004), 149–164.

[WK04] WU J., KOBBELT L. P.: Optimized Sub-Sampling of Point Sets for Surface Splatting. In *Proc. Eurographics* (2004), pp. 643–652.

[ZOF01] ZHAO H.-K., OSHER S., FEDKIW R.: Fast Surface Reconstruction using the Level Set Method. In *Proc. IEEE Workshop on Variational and Level Set Methods in Computer Vision (VLSM)* (2001), pp. 194–202.