Eurographics Symposium on Geometry Processing (2007)
Alexander Belyaev, Michael Garland (Editors)

A Streaming Algorithm for Surface Reconstruction

Rémi Allègre, Raphaëlle Chaine, Samir Akkouche

Université de Lyon, Université Claude Bernard Lyon 1, LIRIS UMR CNRS 5205, France

Abstract

We present a streaming algorithm for reconstructing closed surfaces from large non-uniform point sets based on a geometric convection technique. Assuming that the sample points are organized into slices stacked along one coordinate axis, a triangle mesh can be efficiently reconstructed in a streamable layout with a controlled memory footprint. Our algorithm associates a streaming 3D Delaunay triangulation data-structure with a multilayer version of the geometric convection algorithm. Our method can process millions of sample points at the rate of 50k points per minute with 350 MB of main memory.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

Keywords: Streaming surface reconstruction, streaming Delaunay triangulation, geometric convection.

1. Introduction

We consider the problem of reconstructing mesh surfaces from large sets of sample points using a Delaunay triangulation. This geometric data-structure has proved to be particularly well-suited for devising general and efficient 3D surface reconstruction algorithms [Dey04, CG06]. Most of them consider constructing the Delaunay triangulation of the whole input point set as a preprocessing step. Although many efficient three-dimensional Delaunay triangulation algorithms have been proposed [LJ05], dealing with data sets larger than a few millions points remains a challenging issue, mainly because their design is sensitive to the available memory. It is therefore tempting to discard the global Delaunay triangulation and to resort to more localized data-structures [BMR*99, GKS00, DGH01, OBS05]. However, this can be done to the detriment of robustness, especially when dealing with non-uniform point samples, and may involve stitching and coherent orientation issues that need to be addressed at global scale. In this paper, we investigate a scalable Delaunay-based streaming surface reconstruction algorithm that overrides these limitations.

A streaming algorithm processes a data stream on-the-fly using a memory buffer whose size is a fraction of the length of the stream. Streaming algorithms for large-scale geometry processing have attracted a lot of attention in the past few years [WK03, ILGS03, ILS05, Paj05, IL05]. They have proved to be more efficient than external memory algorithms for handling massive data by minimizing online disk access operations. The common basic idea of streaming geometric algorithms is to exploit the spatial coherence of the data stream, which is the ability for close geometric entities in space to have close representations in the stream. The success of this approach therefore depends on an initial organization of the input data, that may be more or less advanced and easy to obtain [IL05].

A streaming approach for constructing Delaunay triangulations of massive point sets in 2D and 3D has been studied in a recent paper by Isenburg et al. [ILSS06]. Their algorithm relies on a concept of spatial finalization. Space is partitioned into regions and each point is associated with the region in which it falls. Simplices whose circumsphere lies entirely in regions where all points have been read can be written to disk immediately and removed from memory. Provided the stream has sufficient spatial coherence, and there are not too large circumspheres in the resulting triangulation, the proposed algorithm is capable of processing gigantic point sets with a very low memory footprint, which makes it suitable for terrain and volumetric data. However, if the points are distributed on a curve or on a surface, large circumspheres may delay the certification of simplices as Delaunay and cause performance to collapse.

Certifying tetrahedra as Delaunay appears as the major bottleneck for constructing the Delaunay triangulation of a stream of points sampled from a surface on-the-fly. Since the amount of memory required depends on the surface geometry, performance cannot be easily increased through simple

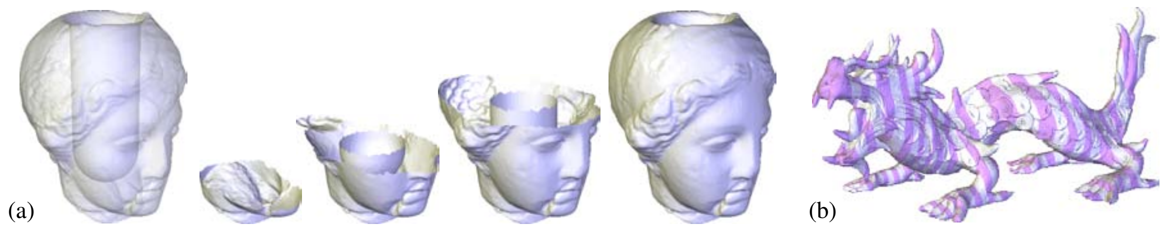

Figure 1: *In (a): Streaming surface reconstruction of a vase model (*IGEA *model drilled by a cylinder). The sampled surface is shown on the left. The streaming reconstruction process has 6 steps illustrated on the right, corresponding to the successive processing of 6 data slices. The concave opening illustrates the ability of the algorithm to adapt to complex topology in each slice, e.g. with internal surface parts. In (b): Result of a streaming reconstruction of the* ASIAN DRAGON *model (3.6M points, 100k points per slice, 100 minutes, 350 MB RAM), with the successive slices depicted with alternating colors.*

data reorganization. In the context of Delaunay-based surface reconstruction however, one important point is that connectivity relations between points that lie far away are not necessarily relevant and may be broken, as done in the Crust algorithm [Ame99].

Our contribution is a streaming surface reconstruction algorithm based on a streaming Delaunay triangulation data-structure and an adaptation of the geometric convection technique [Cha03]. The output is a streamable triangulated surface interpolating the input points that is ideally the same as that would be obtained from the original geometric convection algorithm. One key idea is to introduce extra points in the triangulation to certify tetrahedra as globally Delaunay with a control over the number of data points in main memory. The input point set is partitioned into slices stacked along one coordinate axis that are successively loaded into memory. The streaming surface reconstruction algorithm alternates the incremental Delaunay triangulation of the loaded data, a Delaunay refinement process that splits Delaunay tetrahedra having too large circumspheres, a local surface reconstruction step using a multilayer geometric convection algorithm, and a memory deallocation step for the geometric elements that will be no longer accessed.

Our method makes it possible to reconstruct surfaces from large non-uniform point point sets sampled from closed surfaces with a modest memory footprint. The only requirement is that the data stream is organized into slices with a fixed maximum number of sample points in each slice, and whose height is larger than the data resolution. The algorithm involves only one reading pass on the data stream, that is not required to be complete to start the reconstruction process. The triangle mesh is reconstructed directly in a streamable layout, and can be piped to a streaming mesh processor. The reconstruction process involves no stitching, nor coherent orientation issues by maintaining a single oriented surface all along it.

We demonstrate the effectiveness of our method on various detailed scanned statues, including preprocessed as well as raw data. Our algorithm inherits the robustness properties of the original geometric convection algorithm [Cha03], that is, it is capable of processing raw scanner data with reasonable measurement noise or small misalignments between range scans. Note that the described algorithm performs no subsampling, nor resampling. We concentrate on the ideas that underpin a streaming approach to the problem of surface reconstruction. Redundancy, noise, and undersampling issues will be addressed in future work. A current limitation of the streaming version is that undersampled or unsampled areas may yield incoherent facet orientations.

2. Geometric convection

In this section, we briefly review the geometric convection algorithm as described by Chaine in [Cha03]. This algorithm serves as the basis for our streaming surface reconstruction algorithm.

The geometric convection algorithm is a surface reconstruction algorithm that proceeds by filtering the Delaunay triangulation of an input point set sampled from a closed smooth surface [CG06]. 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 the 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-sphere oriented inside the surface, or *Gabriel half-sphere*, contains no sample point. The result has always manifold topology, but it is not necessarily manifold from a geometric point of view if the input point set is sampled from a surface with boundary or exhibits undersampling. Some parts can share common geometry, while remaining topologically independent.

Let $P \subset \mathbb{R}^3$ denote the input point set and $\hat{\Sigma}$ the surface in convection. The convection scheme can be completely achieved in 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 shrinking surface $\hat{\Sigma}$ is a closed triangulated surface that is maintained at every step, all the facets oriented *inward*. Two facets with opposite orientations can meet –

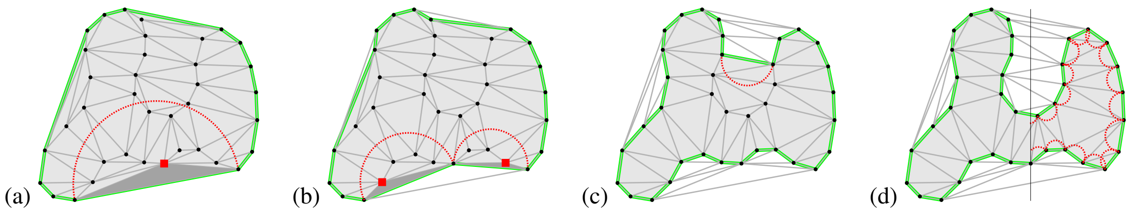

Figure 2: *Geometric convection towards a 2D point set. In (a), an enclosing curve is initialized to the convex hull of the point set. The current edge, enclosed by a non-empty Gabriel half-sphere, forms a Delaunay triangle (dark grey) with the square point. This triangle becomes external, the curve 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-spheres.*

they are said to be *coupled*. Coupled facets can collapse locally, which may change the topology of $\hat{\Sigma}$. 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 order in which the facets of the evolving surface are processed does not influence the result. This is a reason why this algorithm is a good candidate to be translated into a streaming version.

3. Streaming surface reconstruction

The input to our algorithm is a 3D point stream P sampled from a smooth closed surface Σ. We suppose that P is sufficiently dense in the sense it forms an ε-sample of Σ for some sufficiently small constant ε [Ame99]. Our streaming surface reconstruction algorithm proceeds like shrink wrapping P using a single global surface subject to a convection process. Our goal is to make the reconstruction to evolve through one data slice at a time, to certify all surface final facets in that slice, to write them on disk, and then to delete the traversed part of the triangulation before proceeding with the next slice.

Point stream P is required to be organized into a stack of slices $S_0, S_1, \ldots, S_n$ sorted along one coordinate axis (usually the direction of the greatest dimension). Each slice may contain an arbitrary number of points, that do not need to be ordered. It is also not required to know the number of slices in advance. If P does not fit this organization, but has sufficient spatial coherence along one coordinate axis, one way to obtain it at low cost is to perform some kind of on-the-fly low precision sorting in the spirit of the work by Isenburg et al. [ILSS06]. Without loss of generality, slice ordering will be assumed along the z axis throughout the paper, called the *sweep direction*. Any slice S_i is enclosed in a rectangular box with extents along the x and y directions fixed to maximum values common to all slices. These extents have to be larger than those of the smallest slice so that the boxes do not interact with the data. For instance, the maximum extents can be related to the size of the volume in which the data have been sampled, or they can be related to the precision of the arithmetic used to represent the point coordinates. In the following, S_i will indifferently refer to a slice and its enclosing box. The bounding box of the slice set will be denoted as B.

During the streaming reconstruction process, the Delaunay triangulation of the sample points is built incrementally, starting from the lower corners of S_0. Each slice S_i is read in turn to be reconstructed, with i increasing. Together with S_i, slices S_{i+1}, S_{i+2} and S_{i+3} are also loaded, the reason for this will be explained later. Starting from the reconstruction result on slice S_{i-1} for $i > 0$, the goal is to extend the reconstruction of the surface coherently through S_i. A preliminary triangulation refinement step is carried out to ensure that this reconstructed part of the surface will be preserved when loading subsequent slices and updating the triangulation. At the end of the refinement process, the reconstruction can be safely pursued on S_i, and the result is piped to the output mesh file. The explored tetrahedra, except those supporting the interface to the non-visited tetrahedra, are then deleted from the triangulation before the process is iterated on the next slice.

In the following paragraphs, we describe the streaming Delaunay triangulation data-structure and the reconstruction algorithm in detail. Since the proposed algorithm is also valid in two dimensions, a supporting illustration for a sampled planar curve is provided in Fig. 8.

3.1. Streaming Delaunay triangulation

A tetrahedron is said to be in *conflict* with a point $\mathbf{p}$ if $\mathbf{p}$ is contained in its circumsphere. At a given time, a tetrahedon is said to be *final* if it cannot be in conflict with any further inserted point. A subset of tetrahedra of the triangulation is said to be *finalized* if all are final. Suppose that we have successively loaded and triangulated slices $S_0, S_1, \ldots, S_k$, with $k < n$. Reconstructing a coherent surface using geometric convection requires that every tetrahedron to be traversed by the shrinking surface is certified as final. Let t denote a tetrahedron with a vertex $\mathbf{p}$ in $\cup_{i=0}^{k} S_i$. If the circumsphere of t does not overlap S_{k+1}, then t is final. Otherwise, there is no known upper bound on the number of slices that still have to be loaded before t is certified. To make the reconstruction process possible while controlling the number of loaded slices, our strategy consists in computing and inserting extra points in the triangulation through a Delaunay refinement process, so that the set of tetrahedra intersecting a target slice can be finalized, and the extra points are far from the sampled surface in this slice. Before loading a new slice, extra points falling above the loaded slices are removed to prevent them from interfering with the reconstruction result. Finalized tetrahedra are preserved under these extra point removal steps.

After a detailed description of the Delaunay refinement process, we will demonstrate that the point insertion and removal mechanism permits to finalize the set of tetrahedra at a target slice, and therefore that it is possible to perform the surface reconstruction process in a streaming fashion using this data-structure. The conditions under which an approximation of the sampled surface remains available as a subset of the streaming triangulation are discussed in an appendix.

Delaunay refinement process

The iterative slice loading and Delaunay refinement process aims at maintaining a Delaunay triangulation such that at the beginning of step $i \geq 0$, all tetrahedra having one vertex in $\cup_{k=0}^{i-1} S_k$ are certified as final. Such a triangulation is said to be finalized *below* S_i.

At the beginning (Step 0), the data in slices S_0, S_1 and S_2 are loaded in memory, and triangulated. The reason why we need to have three slices loaded at a time will become clear in this section. Slice S_3 is also loaded to reduce the number of extra points required, since some points in S_3 will naturally act as extra points. The resulting triangulation is obviously finalized below S_0, since $\cup_{k=0}^{i-1} S_k$ is empty. Now at Step i, assume that slices $S_0, \ldots, S_{i+3}$ have been loaded and inserted in the current triangulation, that is supposed to be finalized below S_i. We explain how the refinement process is locally performed in order to certify every tetrahedron having its lowest vertex in S_i as final. Such a tetrahedron is said to *belong* to slice S_i. The process iteratively inserts tetrahedra circumcenters in order to discard the tetrahedra that do not fulfill the following circumsphere-slice overlapping condition at both slices S_i and S_{i+1}.

Definition A tetrahedron t is said to be encroached by a slice S_k if the circumsphere of t overlaps S_k.

Condition (Circumsphere-slice overlapping at S_k) Let t denote a tetrahedron such that S_k is the lowest slice that encroaches t. Then t meets the circumsphere-slice overlapping condition at S_k if t is not encroached by S_{k+2}.

The refinement process restricted to slices S_i and S_{i+1} thus guarantees that every tetrahedron circumsphere that overlaps S_i or S_{i+1} is not astride more than two slices. This condition is required to certify S_i tetrahedra as final. Indeed, some circumcenters of S_{i+1} tetrahedra to be refined can conflict with S_i tetrahedra that satisfy the circumsphere-slice overlapping condition at S_i, see Fig. 3(a) for an illustration. The refinement on S_i necessitates loading S_i and S_{i+1}, and the refinement on S_{i+1} also needs S_{i+2}, hence the necessity to load at least three slices at a time. Fig. 3(b) shows the different kinds of authorized circumspheres that may result from the refinement process.

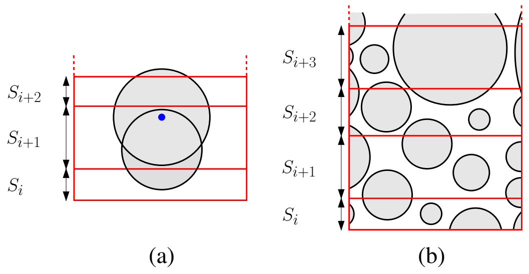

Figure 3: *Configuration illustrating the necessity to load at least three slices at a time (a), and the different types of authorized tetrahedra circumspheres (b) when processing slices S_i and S_{i+1} for finalization of S_i tetrahedra.*

The algorithm refines the tetrahedra that do not satisfy the circumsphere-slice overlapping condition, called *bad* tetrahedra, by adding their circumcenters as vertices. Furthermore, the algorithm guarantees that no extra point is inserted outside the bounding box B. The detailed refinement rule can be stated as follows.

Rule If there is a bad tetrahedron t:

```
compute the circumcenter $\mathbf{c}_t$ of $t$;
if $\mathbf{c}_t$ is outside $B$ then
    let $f$ denote a facet of $t$ visible by $\mathbf{c}_t$;
    compute the circumcenter $\mathbf{c}_f$ of $f$;
    if $\mathbf{c}_f$ is outside $B$ then
        let $e$ denote the edge of $f$ visible by $\mathbf{c}_f$;
        compute the midpoint $\mathbf{c}_e$ of $e$;
        insert $\mathbf{c}_e$;
    else
        insert $\mathbf{c}_f$;
    end if
else
    insert $\mathbf{c}_t$;
end if
```

This algorithm is derived from the classic Delaunay refinement algorithm proposed by Shewchuk [She97]. The elements of the bounding box (vertices, segments, and faces) act as input constraints together with the sample points. We have simplified it to match our specific target conditions and take the particular geometric nature of our constraints into account. In practice, all tetrahedra in S_i and S_{i+1} are pushed into a priority queue. The tetrahedron with the biggest circumsphere is first processed. This optimizes their spatial distribution and contributes to less extra point insertions.

Extra points are constrained to stay on, or inside the bounding box, to ensure that the convex hull of the points is never altered, which is required by the surface reconstruction algorithm. This aspect will be highlighted in the next section. The presented refinement algorithm also ensures that every extra point is inserted as far as possible from the loaded data points. Some extra points may however fall in the vicinity of data points to be loaded later on. This is the reason why before loading S_{i+4}, all extra points outside $\cup_{k=0}^{i+3} S_k$ are removed from the triangulation in a dynamic fashion [DT03]. The removed vertices are necessarily incident to non-final tetrahedra. This vertex removal step is therefore guaranteed not to interfere with the finalized part of the triangulation at S_i. New tetrahedra with too large circumspheres are discarded at the time of the next refinement process.

Finalization guarantee and recurrence

Below we demonstrate that the triangulation resulting from the Delaunay refinement step restricted to S_i and S_{i+1} matches the finalization requirements for S_i tetrahedra, and extends to the next slices.

Lemma After the refinement step restricted to S_i and S_{i+1}, every tetrahedron that belongs to S_i is certified as final. This implies that after having loaded S_{i+4} at Step $i+1$, the triangulation is finalized below S_{i+1}.

Proof Before S_{i+4} is loaded, extra points located above S_{i+3} are removed from the triangulation. The previous refinement steps ensure that the removed points cannot be vertices of S_i nor S_{i+1} tetrahedra. The latter are therefore preserved under this extra point removal step. Similarly, S_i and S_{i+1} tetrahedra are preserved when the data points in S_{i+4} are loaded, this is guaranteed by the fact that they are not encroached by S_{i+4}. Problems could arise from the refinement step following the loading of S_{i+4}. However, the lowest slice that encroaches tetrahedra to be refined is S_{i+2}, which means that the lowest slice for extra point insertions is S_{i+2}. These insertions can break some S_{i+1} tetrahedra encroached by S_{i+2}, but new S_{i+1} tetrahedra resulting from these insertions meet the circumsphere-slice overlapping condition so that their circumcenter will not be inserted in turn. As a consequence, S_i tetrahedra are preserved by the refinement step that follows the loading of S_{i+4}. The same argument runs after the loading of S_{i+k}, $k \geq 4$, at Step $i+k-3$. $\square$

This result ensures that it will be possible to run the surface reconstruction process throughout the successive slices whose tetrahedra have been certified as final. Furthermore, the geometric elements of these tetrahedra can be entirely removed from memory afterwards with the certainty that they will not be accessed later on.

3.2. Reconstruction step

Now that we have an appropriate streaming Delaunay datastructure, we take a look into the surface reconstruction process. In a stream processing context, the basic geometric convection algorithm is not directly practicable. Folds in the sampled surface astride several slices indeed deserve special care (Fig. 4). If the standard convection process was run on such a data set, then it could be required to load the same slice several times. The sliding window consisting of the slices in memory could translate both forward and backward to follow the shrinking surface throughout the cavities of the sampled surface, which is inappropriate in a streaming framework. When reconstructing the surface in a particular slice, we address this issue by locally running the convection process on the successive encountered parts of the surface like an onion peeling process through a succession of convection levels (Fig. 5). Each convection level is associated with an index to remember the rank at which it has been run. The first external level convection is indexed 0 and the convection of level $k+1$ begins where the convection of level k has stabilized. Depending on a surface layer is reached from the outside or from the inside by the shrinking surface, the convection process is labelled either *external*, with an even level index, or *internal*, with an odd level index. The reconstruction process therefore alternates external and internal convections until an empty result is obtained. This also makes it possible to discover voids, which can be useful e.g. with tomography data.

In the following paragraphs, we explain how the surface is first initialized and how its evolution is controlled throughout a slice.

Surface initialization Once all S_0 tetrahedra are certified as final, an external shrinking surface $\hat{\Sigma}$ is initialized on the

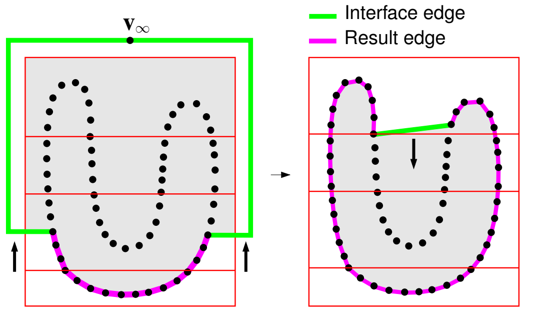

Figure 4: *The standard convection process applied to a 2D point set sampled from a closed curve folded astride several slices. Some parts of the curve can only be discovered if some slices are visited twice.*

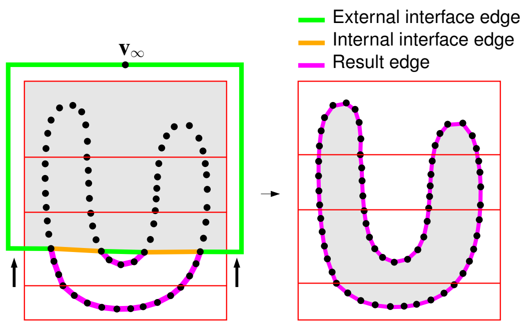

Figure 5: *The multilayer convection process applied to the same 2D point set as previously. Every part of the curve can now be discovered with only one pass on the slice stack.*

lower convex hull of the points located in S_0 (no matter if they are input data points or extra points). $\hat{\Sigma}$ is artificially closed by also including some infinite facets at the interface between S_0 and S_1 tetrahedra. Infinite facets are facets of the Delaunay triangulation connecting every edge of the convex hull to an infinite vertex denoted as $\mathbf{v}_\infty$ that has no geometry associated with it.

Infinite facets are elements of infinite tetrahedra that connect every facet of the convex hull to $\mathbf{v}_\infty$. $\hat{\Sigma}$ traverses finite as well as infinite tetrahedra during the surface evolution process. In order to extend the reconstruction in a coherent fashion from one slice to the other, the visited set of infinite tetrahedra also has to be finalized. To be final, an infinite tetrahedron connected to a convex hull facet f has to be such that any further inserted point is invisible from f. This explains the interest of bounding the region of space in which all points are inserted using a convex volume. Preserving the convex hull all along the streaming reconstruction process is a vital precaution for the success of the method.

Surface evolution (at level k) We describe the evolution of $\hat{\Sigma}$ at convection level k throughout a slice S_i, assuming that layers $S_0, \ldots, S_i$ have been previously loaded and triangulated, and that the triangulation is finalized below S_{i+1}. A facet $f \in \hat{\Sigma}$ is open if one of these conditions is satisfied:

- f Gabriel half-sphere contains an input data point or f hides a pocket.
- one vertex of f is an extra point or $\mathbf{v}_\infty$.
- f is a facet certified to be in $\hat{\Sigma}$ at the end of the convection of level $k-1$ for $k > 0$.

The process locally stops when a facet of $\hat{\Sigma}$ is at the interface between a tetrahedron of S_i and a tetrahedron of S_{i+1}.

The reconstruction result includes *result-certified* facets and *waiting* facets. Result-certified facets are those that are guaranteed to appear in the reconstruction result. They include the facets whose full Gabriel sphere contains no sample points, and whose coupled facet does not hide any pocket. These facets cannot collapse. Conversely, waiting facets are such that their associated Gabriel half-sphere is empty, but not their full Gabriel sphere, or their coupled facet hides a pocket. These facets are not yet guaranteed to be part of the final result: they may be reached from behind by a subsequent convection process at a level of the the same parity and collapse.

When the convection process stops at a given level, the evolving surface may still enclose some tetrahedra of the current slice S_i. At the next level, the convection process is relaunched upward from the facets that have just been result-certified. If a waiting facet is reached from behind by a subsequent convection process at a level of opposite parity, then it is result-certified.

After all the convection levels have been achieved on S_i, the tetrahedra that have been traversed can be deleted. Only those supporting the surface are kept, until the next slice has been processed. The refinement step is run on the triangulation in order to finalize the tetrahedra in S_{i+1}, all the convection levels are performed, one after the other, starting from where they were previously stuck.

4. Implementation details

We have implemented our algorithm using the Computational Geometry Algorithms Library, CGAL [CGA]. We use CGAL's Delaunay triangulation data-structure with filtered arithmetic for robust conflict tests. We focus below on the implementation of point insertion in the triangulation, and on the ordering of the output facets to match a streamable layout.

Point insertions For each point to insert, the enclosing tetrahedron is first located. These locations have to be constrained in order to avoid traversing the deleted parts of the triangulation. This is achieved with a visibility walk that favors visible facets oriented upward according to the sweep direction. The starting point of that walk is the newest created tetrahedron. Then the triangulation is updated locally using the standard CGAL procedure that consists in removing the tetrahedra that are no longer Delaunay, and then starring the cavity.

Output facets ordering The facets of the final result are put into a priority queue so that when they are written to disk, they are sorted along the z axis. The output mesh may be easily described in a streamable layout. According to the terminology of Isenburg and Lindstrom [IL05], we choose post-order formatting, which is the most direct way to generate a streamable mesh in this context. A vertex appears just after the appearance of its last incident facets. This layout is easily obtained by maintaining an incident facets counter for every vertex decremented each time a facet is written to disk.

5. Results and discussion

We have tested our algorithm on several non-uniform point set models obtained from laser-range scanning. The data sets include raw scanner data (Fig. 6) as well as preprocessed data (Fig. 1). Prior to the streaming reconstruction process, all input point streams were organized into stack of slices sorted along the coordinate axis of maximum dimension, and each slice contains a fixed number of points (maybe except the last one that may contain points in excess). Timings and memory use are reported in Table 1. All the results presented here were obtained on a Pentium IV 3.0 GHz, 2 GB RAM workstation.

Performance and scalability Fig. 7 details the performance of our algorithm for the IGEA model with varying slice sizes. On this model, every every slice has approximately the same height, which reflects the average case. The diagram shows that the memory consumption decreases linearly with the number of points per slice, although the number of extra points grows cubically as the height of the slices decreases. The streaming reconstruction time increases linearly with the number of slices. Some variations can be however expected when the slices heights are not constant, as

Model		Streaming surface reconstruction						Rec. with [Cha03]	
name	#points	#points/slice	#slices	#extra points	#facets	time	mem. use	time	mem. use
RAM	622,716	50,000	12	2,264	1,233,254	9:53	171 MB	15:32	281 MB
		100,000	6	241	1,233,254	6:34	312 MB		
ASIAN DRAGON	3,609,600	50,000	72	538,176	7,216,158	262:16	203 MB	38:23	1,510 MB
		100,000	36	75,214	7,216,158	100:36	350 MB		
THAI STATUE	5,001,964	100,000	50	98,795	10,002,341	142:47	365 MB	–	> 2,000 MB

Table 1: *Performance of our streaming reconstruction framework for various input point sets. Computational timings are given in minutes:second for both initial reconstruction and correction steps. Memory use corresponds to the maximum amount of memory used, in megabytes. All tests were performed on a Pentium IV 3.0 GHz, 2 GB RAM workstation.*

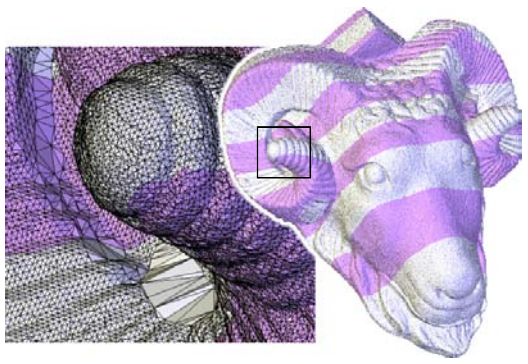

Figure 6: *Streaming reconstruction of the* RAM *model (622,716 points) from a raw data set with noise and undersampling. The close up view shows triangles on the boundary of an unsampled region.*

with the RAM model (Fig. 6). The maximum memory use will always depend on the smallest height of a slice. Compared to the performance of the original in-core geometric convection algorithm [Cha03], memory use diminishes drastically as the ratio point set size/slice size increases (Table 1).

We currently have no bound on the maximum number of extra points required for a model. This depends on the shape of the surface to be reconstructed, on the sampling density, and on the height of the slices. A theoretical condition interrelating these three aspects would be the key for proving the scalability of the method.

Results quality Provided no extra point is inserted too close to the data, our streaming surface reconstruction algorithm produces results that are identical to the results of the original geometric convection algorithm in a global Delaunay triangulation of the data [Cha03], maybe except at some sliver tetrahedra (see appendix discussion). Our method naturally inherits the robustness properties of the original algorithm with noisy data (Fig. 6). However, we have supposed that the result of the streaming surface reconstruction algorithm is a volume. If the interior and exterior parts of the surface at each slice are not distinguishable, then the facets of the result may not be all consistently oriented.

Comparison with other work To our knowledge, no existing surface reconstruction algorithm has been specif-

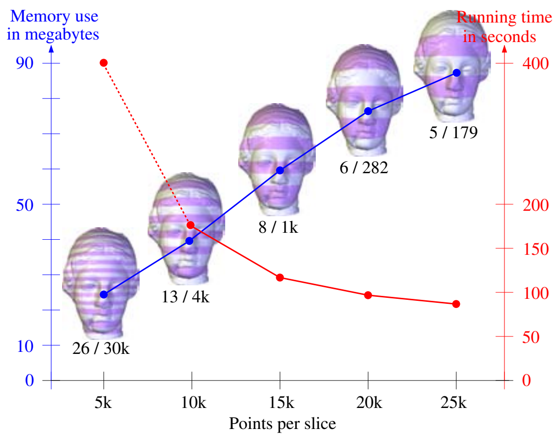

Figure 7: *Performance of our method on the* IGEA *model (134,345 points) with different slice size. For each model, the number of slice is provided (left) together with the number of extra points required (right).*

ically designed to process streams of points. Nonetheless, many state-of-the-art two-dimensional triangulation methods could be adapted without much effort [BMR*99, OBS05, SFS05]. Some other streaming-capable surface reconstruction algorithms are based on range scans merging [CL96, RCG*04]. Our method is different in the sense it is designed for streaming surface reconstruction, and can process more general data than previous streaming-capable methods thanks to a Delaunay triangulation data-structure. No normal vectors, nor constraint on sampling uniformity are required.

Another advantage of our algorithm over some existing surface reconstruction methods is that not all data need to be known in advance. Algorithms relying on hierarchical data-structures, including the extremely efficient methods by Kazhdan et al. [KBH] and Boubekeur et al. [BHGS06], cannot easily accomodate progressive data availability.

Currently, the output of our algorithm is a triangle mesh interpolating the input data points. Our method is therefore not competitive with recent surface reconstruction methods that consider simplifying the input point set [ACA06], or resampling it using local surface approximation techniques [OBS05, SFS05]. It would be worth extending our algorithm in this way to accelerate the reconstruction process and adapt the sampling density to the local geometry.

6. Conclusion and future work

In this paper, we have presented a simple, scalable streaming approach to the problem of surface reconstruction from point sets sampled from a closed surface. We have designed a streaming Delaunay triangulation data-structure that fits the purpose of reconstructing surfaces from a stream of points without the need to compute the triangulation of the whole data set and to maintain it in memory. The only requirement is that the sample points are organized into a stack of slices ordered along one coordinate axis. Based on this data-structure, we have developed a streaming extension of the geometric convection algorithm that can directly reconstruct streamable mesh surfaces. The results are similar to the results of the original geometric convection algorithm in a global Delaunay triangulation of the data, provided the slices are large with respect to the sampling density. Our framework may also offer interesting perspectives for other Delaunay-based surface reconstruction algorithms.

Our algorithm is a first step towards a more elaborated streaming surface reconstruction framework. Several aspects deserve further investigations in order to offer guarantees on the results as well as on its scalability. In the future, we also plan to develop several extensions. We first would like to improve the performance of point insertions in the triangulation, e.g. using localized Delaunay hierarchies [Dev02]. Our current method interpolates the input points, which can be an advantage for some applications. But performance could be significantly improved in case the data need simplification. In the same way as the geometric convection algorithm was extended to a selective version [ACA06, ACA07], our streaming algorithm could be extended in order to simplify the input data on-the-fly while reconstructing, without inserting every data point into the triangulation, and also offer update abilities. In the presence of noise or incomplete data, it could be also an advantage to mix our method with some local surface approximation technique [SFS05, OBS05].

Acknowledgments

Point set models are provided courtesy of Cyberware (IGEA), F. Bernardini (RAM), and Stanford Scanning Repository (XYZ RGB ASIAN DRAGON and THAI STATUE).

References

[ACA06] ALLÈGRE R., CHAINE R., AKKOUCHE S.: A Dynamic Surface Reconstruction Framework for Large Unstructured Point Sets. In *Proc. IEEE/Eurographics Symposium on Point-Based Graphics* (2006), pp. 17–26. 7, 8

[ACA07] ALLÈGRE R., CHAINE R., AKKOUCHE S.: A Flexible Framework for Surface Reconstruction from Large Point Sets. *Computers & Graphics 31*, 2 (2007). 8

[Ame99] AMENTA N.: The Crust Algorithm for 3D Surface Reconstruction. In *Proc. Symposium on Computational Geometry* (1999), pp. 423–424. 2, 3, 10

[BHGS06] BOUBEKEUR T., HEIDRICH W., GRANIER X., SCHLICK C.: Volume-Surface Trees. In *Proc. Eurographics* (2006), pp. 399–406. 7

[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. 1, 7

[CG06] CAZALS F., GIESEN J.: Delaunay Triangulation based Surface Reconstruction: Ideas and Algorithms. In *Effective Computational Geometry of Curves and Surfaces* (2006), Boissonnat J.-D., Teillaud M., (Eds.). 1, 2

[CGA] http://www.cgal.org. 6

[Cha03] CHAINE R.: A Geometric Convection Approach of 3-D Reconstruction. In *Proc. Symposium on Geometry Processing* (2003), pp. 218–229. 2, 7

[CL96] CURLESS B., LEVOY M.: A Volumetric Method for Building Complex Models from Range Images. *Computer Graphics (Proc. SIGGRAPH) 30* (1996), 303–312. 7

[Dev02] DEVILLERS O.: The Delaunay hierarchy. *Internat. J. Found. Comput. Sci. 13*, 2 (2002), 163–180. 8

[Dey04] DEY T. K.: Curve and surface reconstruction. *Chapter in Handbook of Discrete and Computational Geometry, Goodman and O' Rourke eds., CRC press, 2nd edition* (2004). 1

[DGH01] 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. 1

[DT03] DEVILLERS O., TEILLAUD M.: Perturbations and Vertex Removal in a 3D Delaunay Triangulation. In *Proc. ACM-SIAM Symposium on Discrete Algorithms* (2003), pp. 313–319. 5

[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. 2

[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. 2

[GKS00] GOPI M., KRISHNAN S., SILVA C. T.: Surface Reconstruction based on Lower Dimensional Localized Delaunay Triangulation. In *Proc. Eurographics* (2000), pp. 363–371. 1

[IL05] ISENBURG M., LINDSTROM P.: Streaming Meshes. In *Proc. IEEE Visualization Conference* (2005), pp. 231–238. 1, 6

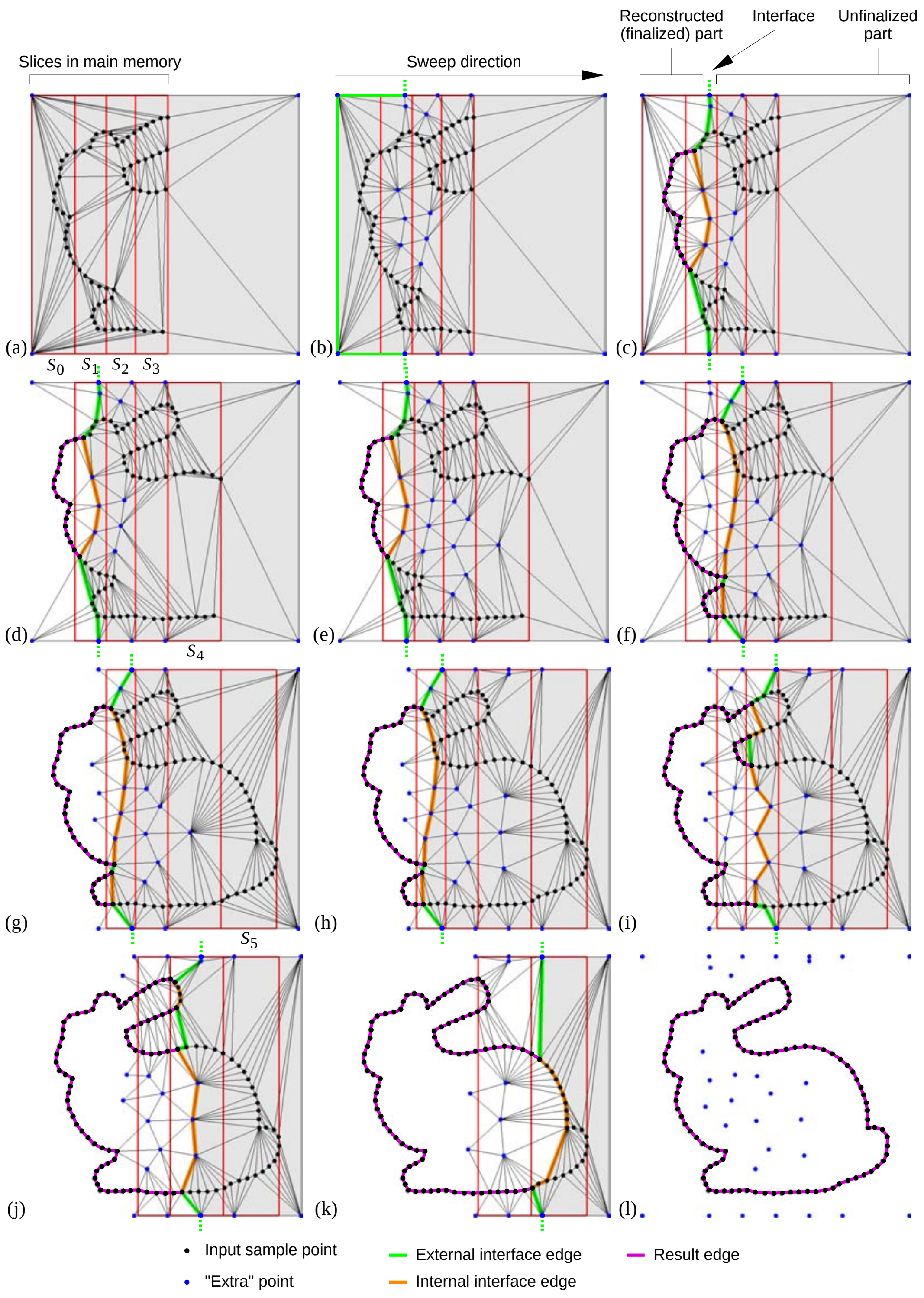

Figure 8: *Streaming reconstruction of a curve in 2D. In (a), the first 4 slices are loaded, and the Delaunay triangulation of the points built. In (b), the triangulation is refined and the bounding curve is initialized. The result of the first reconstruction step on S_0 is shown in (c). Figures (d)–(f) show the loading and triangulation of S_4, the consecutive refinement step, and the result of the reconstruction through S_1. Figures (g)–(i) illustrate the remainder of the process on S_2–S_5. The final reconstruction result is shown in (l) together with the required set of extra points.*

[ILGS03] ISENBURG M., LINDSTROM P., GUMHOLD S., SNOEYINK J.: Large Mesh Simplification using Processing Sequences. In *Proc. IEEE Visualization* (2003), pp. 465–472. 1

[ILS05] ISENBURG M., LINDSTROM P., SNOEYINK J.: Streaming Compression of Triangle Meshes. In *Proc. Symposium on Geometry Processing* (2005), pp. 111–118. 1

[ILSS06] ISENBURG M., LIU Y., SHEWCHUK J., SNOEYINK J.: Streaming Computation of Delaunay Triangulations. *ACM Transactions on Graphics (Proc. SIGGRAPH) 25*, 3 (2006), 1049–1056. 1, 3

[KBH] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proc. Symposium on Geometry Processing*, pp. 61–70. 7

[Kol05] KOLLURI R.: Provably good moving least squares. In *Proc. ACM-SIAM Symposium on Discrete Algorithms* (2005), pp. 1008–1017.

[LJ05] LIU Y., J.SNOEYINK: A comparison of five implementations of 3D Delaunay tessellation. In *Combinatorial and Computational Geometry* (2005), Goodman J. E., Pach J., Welzl E., (Eds.), vol. 52 of MSRI Publications, Cambridge, pp. 435–453. 1

[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. 1, 7, 8

[Paj05] PAJAROLA R.: Stream-Processing Points. In *Proc. IEEE Visualization* (2005), pp. 239–246. 1

[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. 7

[SFS05] SCHEIDEGGER C. E., FLEISHMAN S., SILVA C. T.: Triangulating point set surfaces with bounded error. In *Proc. Symposium on Geometry Processing* (2005), pp. 63–72. 7, 8

[She97] SHEWCHUK J. R.: *Delaunay Refinement Mesh Generation*. PhD thesis, Technical Report CMU-CS-97-137, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1997. 4

[WK03] WU J., KOBBELT L.: A Stream Algorithm for the Decimation of Massive Meshes. In *Proc. Graphics Interface* (2003), pp. 185–192. 1

[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. 2

Appendix: Considerations on the slices heights

The slice height is a property of the input data slices, that all incorporate a fixed maximum number of points. Slices that are too narrow with regard to the sampling density can influence the reconstruction result.

Our goal is to reconstruct an approximation of the sampled surface Σ that is close to the result of the original geometric convection process from P. Let E denote the set of extra points inserted in the Delaunay triangulation, removed extra points excluded. The central concern is to ensure that an approximation of Σ is available as a subset of the Delaunay triangulation of $P \cup E$, which requires that the extra points lie sufficiently far from the points in P. For this purpose, the slices heights have to be large enough with respect to the resolution of the input point set (Fig. 9).

The insertion of an extra point aims at breaking a large tetrahedron circumsphere astride three slices. If the input data set is sufficiently dense, such a large sphere is approximately centered on the medial axis of the volume bounded by Σ, or it may coincide with the circumcenter of a sliver tetrahedron, i.e. a tetrahedron whose four vertices lie close to a plane [Ame99]. If the circumcenter of such a sliver lies near Σ, it can be stated that its circumsphere cannot be large, and thus that it does not require to be split. Each time that a new extra point is inserted, the medial axis considered is the medial axis of the volume minus the previously inserted extra point. Granted that the refinement algorithm converges in a finite number of steps, the extra points are then all inserted *as far as possible* from Σ.

Provided the height of every slice is large enough regarding the maximum local granularity, the centers of the largest remaining tetrahedra circumspheres remain faraway from Σ. Otherwise, some extra points can be inserted near Σ so that an approximation can become unavailable in the Delaunay triangulation. In the example in Fig. 9(c), the height of one slice is almost too small. In order to derive a conservative bound from the sampling granularity, further considerations on the Delaunay refinement process would be required.

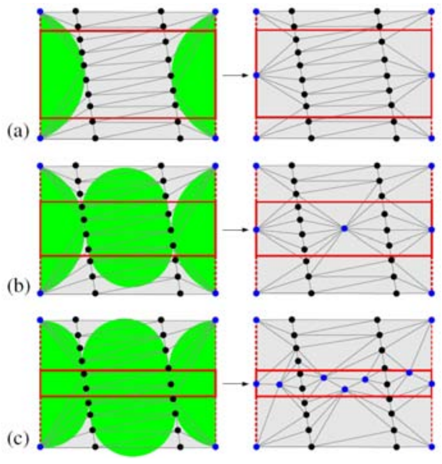

Figure 9: *Refinement results for a given slice with decreasing height from (a) to (c). The circumspheres of the tetrahedra to be refined are shown in green.*