Surface reconstruction from a sparse point cloud by enforcing visibility consistency and topology constraints

Maxime Lhuillier

Université Clermont Auvergne, CNRS, SIGMA Clermont, Institut Pascal, F-63000 Clermont-Ferrand, France

The reference of this paper is: Maxime Lhuillier, Surface reconstruction from a sparse point cloud by enforcing visibility consistency and topology constraints, Computer Vision and Image Understanding, 175:52-71, 2018.

This is the accepted manuscript version that is available at the webpage of the author. The published version (DOI: 10.1016/j.cviu.2018.09.007) is available at Elsevier via `https://doi.org/10.1016/j.cviu.2018.09.007`

Highlights

- Surface reconstruction under-uses topology constraints in Computer Vision
- Start from a previous method enforcing manifoldness on a sparse point cloud
- Simultaneously enforce visibility consistency and low genus for the first time
- Improve the removal of surface singularities and the escape from local extrema
- Experiment on long video sequences taken by a helmet-held omnidirectional camera

Abstract

There are reasons to reconstruct a surface from a sparse cloud of 3D points estimated from an image sequence: to avoid computationally expensive dense stereo, e.g. for applications that do not need high level of details and have limited resources, or to initialize dense stereo in other cases. It is also interesting to enforce topology constraints (like manifoldness) for both surface regularization and applications. In this article, we improve by several ways a previous method that enforces the manifold constraint given a sparse point cloud. We enforce lowered genus, i.e. simplified topology, as a further regularization constraint for maximizing the visibility consistency encoded in a 3D Delaunay triangulation of the points. We also provide more efficient escapes from local extrema, an acceleration of the manifold test and more efficient removals of surface singularities. We experiment on a sparse point cloud reconstructed from videos, that are taken by a helmet-held omnidirectional multi-camera moving in an university campus.

Keywords: Surface Reconstruction, Manifold, Genus, Visibility, Sparse Features, Environment modeling.

1. Introduction

The automatic surface reconstruction of an environment from an image sequence is still an active research topic. Here we present a method that generates a surface using topology constraints and given a sparse cloud of 3D points reconstructed from the images. First Sec. 1.1 describes these constraints and their importance, since they are under-explored in Computer Vision. Then Secs. 1.2 and 1.3 summarize previous works that reconstruct a surface from such a cloud or topology constraints, respectively. Last Sec. 1.4 presents our contributions.

1.1. Why enforcing topology constraints ?

There are two reasons to enforce constraints on the computed surface. First this regularizes the problem and helps to deal with noisy and lacking input points. Second topological constraints are needed in downstream processing and applications most of the time. There are several topology constraints.

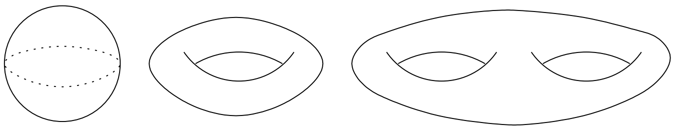

Figure 1: Examples of manifold surfaces in $\mathbb{R}^3$ having different genuses g. Left: sphere (with dotted equator) meets $g = 0$. Middle: simple torus meets $g = 1$. Right: double torus meets $g = 2$.

1.1.1. Manifoldness

A surface is manifold if every of its point has a small neighborhood in the surface that is mapped to $\mathbb{R}^2$ by a homeomorphism (a bijective and continuous function whose inverse is continuous). If a surface point does not meet this condition, the point is singular.

Now we briefly remind arguments of [22]. Method implicitly assumes that surface is manifold if it estimates normal or curvature. This holds not only for surface reconstruction methods that regularize using smoothing ([12, 40]), but also for applications like rendering ([5]). The manifold constraint can be seen as the weakest smoothing that can be applied to a surface, even when the density of input points is too low to estimate reliable normal or curvature. According to experiments of [22], the manifold constraint is necessary in our context: 25% of the vertices at the surface are singular if we do not enforce it.

1.1.2. Low genus

Assume that we have a manifold surface in $\mathbb{R}^3$. Its genus g is the number of surface handles, e.g. $g = 0$ for a sphere and $g = 1$ for a torus as shown in Fig. 1.

We say that there is topological noise ([42]) if the estimated surface has a g excess compared to the true surface, i.e. if the estimate has spurious handle(s). Both lack of points and point noise in the input are reasons of topological noise. Enforcing a low genus during surface computation can provide a strong regularization ([49]), but most methods are unable to do this.

The topological noise is usually removed by a post-processing. This is useful for downstream processing such as smoothing, refinement and texture mapping ([42, 5]), which require low topological noise. An example is the dense stereo refinement: the surface evolves in 3D such that it minimizes a photo-consistency function without changing the genus ([12, 40, 33]). The lower the topological noise, the better the expected convergence. (Think about a torus that tries to approximate a sphere by minimizing a photo-consistency score.)

Last we provide theoretical remarks, which may motivate to consider genus in Computer Vision methods that reconstruct surface. First g and Gaussian curvature K meet $\iint_S K = 4\pi(c - g)$ if S is a manifold surface with c connected components ([7]). This is consistent with intuition: we can reduce the (negative Gaussian) curvature of the surface by removing handles if $c \ll g$, and this looks like curvature-based regularization. Second there are assumptions ([1]) on the input point cloud and the true scene surface such that a 3D Delaunay triangulation based surface reconstruction method can provide the good g.

1.1.3. Connectedness

Similarly as in the genus case, both noise and lack of points can generate spurious connected components of the surface. This is also topological noise. Thus connectedness can be used as constraint ([30]).

1.2. Previous sparse methods

The sparse methods estimate a surface from a 3D point cloud, which is sparse since it is computed from interest points and/or contour points having uneven and low distributions in the images. Nevertheless the points have decent confidence thanks to standard pipeline including point selections, robust and optimal reconstruction by using RANSAC and non-linear least squares.

We shortly remind advantages of the sparse methods. First they are interesting for both time and space complexities, especially for obtaining compact models of large scale environments. This is useful for computations with limited hardware resource, or for applications that need high scalability and do not need the level of details provided by dense stereo. Second they can initialize dense stereo methods to improve accuracy and obtain more detailed reconstructions, if the experimental conditions are favorable to get enough texture in the images.

Most sparse methods use a 3D Delaunay triangulation of the input points. This discretizes the space by a set T of tetrahedra. Furthermore, every point has visibility information: a set of line segments called rays linking the point and every camera location that contributed to reconstruct the point. A bipartition of T is obtained: the free-space is the set of the tetrahedra intersected by at least one ray, the free-space complement is the matter. The set of triangles separating free-space and matter is a candidate surface ([26, 23]), but it is neither manifold nor robust to bad points. This surface can be converted into a manifold surface by vertex duplications ([10]) or perturbations ([34]), but the result is not robust to bad points since the geometry is (almost) the same. Thus previous sparse methods ([8, 31, 16, 40, 14, 36, 22, 25, 32]) estimate another bipartition of T, that is the first one up to a regularization. Sec. 1.3.1 will summarize these works if they enforce the manifold constraint.

Other sparse methods ([27, 39, 37, 13, 18, 2]) use 2D (not 3D) Delaunay triangulations constrained in images by interest points and image contours. However, [27, 39] are limited to simple topology (plane/sphere), large scale scenes cannot be computed efficiently by [37] due to the use of regular decomposition of the 3D space, and [13, 18, 2] do not generate a globally consistent manifold surface but several (view-centered) local models with redundancy. There are also methods that need strong assumptions to segment the scene using planar regions ([41, 9]) or swept surfaces for architectural scenes ([43]).

1.3. Previous works enforcing topology constraints

Sec. 1.3 excludes works that are far from our context, e.g. the dense stereo refinements that deform a surface without topology change. (Number of connected components and genus are fixed.)

1.3.1. Manifoldness

Here we focus on the surface reconstruction methods with same input as ours: a sparse point cloud estimated from images and their visibility information. Works ([8, 22, 25, 32]) first compute a 3D Delaunay triangulation T of the points and the free-space/matter bipartition of T (Sec. 1.2), then use the manifold property as a constraint to search a surface interpolating most points. In works ([22, 25, 32]), a second bipartition of T in outside tetrahedra and their inside complement is computed by growing an outside set O in the free-space: O should be as large as possible such that the set of the triangles separating outside and inside (i.e. the boundary of O) forms a manifold surface. Visual artifacts of [22] are corrected by [25] and are limited by [32], but the artifact problem is not completely solved ([20]). The visibility is partly ignored by [8] since the inside grows in the matter.

We note that methods ([22, 25, 32]) are shrink-wrapping methods in the discrete space T guided by the visibility and the manifold constraint. Previous shrink-wrapping methods ([17, 11]) ignore the visibility, [11] do not use the manifoldness as a constraint on the surface since they use the method of [10], the surface has zero-genus by [17].

1.3.2. Low genus

The topological noise is usually removed, or the genus decreases, by a post-processing of the surface reconstruction: [42, 46] remove spurious handles (assuming that they are smaller than true handles) and [11] provide multiresolution surfaces including topology simplifications. Now we focus on methods enforcing a low genus during the surface reconstruction. This has been reported in very few literature.

Assuming that the surface topology is known (both number of connected components and genus), [49] show that it is possible to reconstruct an accurate surface from a few planar cross-sections. Note that a cross-section is nothing but a sparse cloud of points forming closed curves in $\mathbb{R}^3$.

In a different context, [15] estimate a surface by using a regular grid of voxels (which is inefficient for large scale scenes like ours) to discretize the unsigned distance function to the input points. Compared to the other volumetric methods that use a signed distance function, this method reduces topological noise artifacts caused by misalignments of 3d scan points.

Our previous work ([22]) removes the largest ("visually critical") handles generated by a step of the method. Such a handle removal contributes to the visibility consistency optimization, since the outside O increases in the free-space. (It is not like a post-processing ignoring the visibility.) At first glance this quantitatively enforces low genus during surface reconstruction, but we show that this is not the case.

1.3.3. Connectedness

The connectedness is enforced by [30] to preserve fine scale details (such as a stick or a rope) without decreasing the genus. First an approximate visual hull including the surface is computed thanks to a green room. Then a convex optimization problem is solved: find an occupancy labeling (sampled in a regular grid) that minimizes a photo-consistency score subject to linear constraints on labeling derivative. The constraints are defined from the visual hull such that the final surface and the visual hull boundary have similar topology.

In contrast to the previous example which enforces inside connectedness, we use outside connectedness like [22, 25]: the outside O grows continuously in the free-space during most computation. This constraint is legitimate assuming continuous camera motion and opaque matter in the scene.

1.4. Our contributions

The surface reconstruction method of [22] is the basis of our work. We improve it in different ways: acceleration thanks to a new manifold test (Sec. 3), more efficient repair for singularity removal (Sec. 4), enforce low genus (Sec. 5), more efficient escapes from local extrema (Sec. 6). Only a part of Sec. 5 was published by [25]. Now we describe the four contributions.

The new manifold test is important since it is used in almost all operations of our methods. It is faster than the previous test by [22] for reasons that were ignored before: the 3D Delaunay triangulation is orientable and the number of surface triangles including a vertex is small in practice. We also provide details on the orientability itself since this property (in the 3D case) is rarely described in the surface reconstruction bibliography.

The new repair operation succeeds more often than the old repair by [22] since the former is guided by an analysis of the surface singularities. (The latter is not.) The former is also quite faster; the latter is the most time consuming operation used by [22].

Euler's formula (unused by [22, 25]) reveals that the genus of the O boundary increases a lot due to operations of [22]. We present two methods that greatly reduce the genus. The main idea is to apply such an operation only if it generates a visually non-negligible update of the O boundary at the camera locations used to reconstruct the input points. Up to our knowledge, these methods are the only ones that simultaneously enforce low genus and visibility consistency in surface reconstruction.

Local extrema are partly due to limitations of an operation called "Shelling", which adds one tetrahedron at once to the growing set O. Although it provides most tetrahedra in O, it can "get stuck" in unexpected cases ([20, 47]). We propose a method based on the new repair to escape from local extrema.

In the experiments (Sec. 7), we check the four successive improvements provided by Secs. 3, 4, 5 and 6. We also compare our final method to a graph-cut method by [40]. Here we do not focus on incremental surface reconstruction like [25, 32], although our contributions can be applied to it. (Incremental methods have their own topics.)

2. Prerequisites

2.1. Main notations and definitions

Our input is a set of 3D points and their visibility, i.e. every point $\mathbf{p}_i \in \mathbb{R}^3$ is reconstructed from viewpoints $\mathbf{c}_j \in \mathbb{R}^3$ such that $j \in V_i$. A line segment $\mathbf{p}_i\mathbf{c}_j$ is called a *ray* if $j \in V_i$. Let T be the 3D Delaunay triangulation computed from the $\mathbf{p}_i$; T is a tetrahedron set that discretizes the convex hull of the $\mathbf{p}_i$. The rays define a bipartition of T: the set $F \subseteq T$ of the *free-space* tetrahedra that are intersected by at least one ray, and the others (the *matter* tetrahedra).

A *simplex* σ is the convex hull of $k+1$ points $\mathbf{v}_0 \cdots \mathbf{v}_k$ in general position in $\mathbb{R}^3$, i.e. $\mathbf{v}_1 - \mathbf{v}_0 \cdots \mathbf{v}_k - \mathbf{v}_0$ are linearly independent. If k is 0, 1, 2 or 3, σ is a vertex, edge, triangle, or tetrahedron, respectively. A simplex σ' is a *face* of σ if σ' is the convex hull of some of the $\mathbf{v}_i$ above. (Thus $\sigma' \subseteq \sigma$.) Two tetrahedra are *adjacent* if they have a common triangle face. Vertices have bold fonts, e.g. $\mathbf{v}$ is a vertex and $\mathbf{ab}$ is an edge. We use the following notations: Δ is always a tetrahedron, the sets of the tetrahedra in T that include $\mathbf{v}$ or $\mathbf{ab}$ are

$$T_{\mathbf{v}} = \{\Delta \in T, \mathbf{v} \in \Delta\} \text{ and } T_{\mathbf{ab}} = \{\Delta \in T, \mathbf{ab} \subseteq \Delta\}. \quad (1)$$

If W is a set of simplices, $c(W)$ is the *closure* of W, i.e. the set of the faces (including the vertices) of the simplices in W. We have $W \subseteq c(W)$. If $K \subseteq c(T)$, we sometimes write K instead of the union $|K|$ of its simplices, e.g. we say that K is connected.

If $X \subseteq T$, ∂X is the *boundary* of X, i.e. the set of every triangle that is a face of exactly one tetrahedron in X. We have $\partial X \subseteq c(X)$ and every triangle in $c(X) \setminus \partial X$ is a face of exactly two tetrahedra in X (one in each side of the triangle).

Let $\mathbf{v} \in c(\partial X)$, i.e. $\mathbf{v}$ is a vertex of a triangle in ∂X. We say that $\mathbf{v}$ is *regular in* ∂X if the triangles in ∂X including $\mathbf{v}$ form a ring around $\mathbf{v}$: the set $\{\mathbf{ab}, \mathbf{abv} \in \partial X\}$ is a cycle ([3, 19]). Otherwise, $\mathbf{v}$ is *singular in* ∂X. Furthermore, ∂X is *manifold* iff (if and only if) every vertex in $c(\partial X)$ is regular in ∂X. In general, ∂F is not manifold. Let O be a set of *outside* tetrahedra: we have $O \subseteq F \subseteq T$ and ∂O is manifold. Let g be the genus of ∂O.

2.2. Operations on the set of outside tetrahedra

Here we summarize operations introduced by [22, 25]. They are basic components of methods considered in this paper. Technical details are omitted (as most as possible) to make easier the paper understanding.

2.2.1. Principles

The target of our methods is $O \subseteq F$ that maximizes a visibility score function $f(O)$ subject to the constraint that ∂O is manifold. For efficiency, $f(O)$ is defined as the sum for each tetrahedron $\Delta \in O$ of a positive real $f(\Delta)$. We choose $f(\Delta)$ as the number of ray intersection(s) for Δ, but alternative definitions of f could be investigated inspired by [31, 32] and others. (This is not the topic of the paper.) Intuitively, a large O in F in the inclusion sense is a good solution if ∂O is manifold.

Every operation tries to add to O tetrahedra that are in $F \setminus O$. It iteratively generates a series of tetrahedron sets $O_0, O_1 \cdots O_n$ where O_0 is the initial value of O and $n > 0$. The operation is successful iff ∂O_n is manifold and $f(O_n) \geq f(O_0)$. It returns the new value of O: O_n if success or O_0 if failure.

2.2.2. Operations Sh., T.E., F.R., C.H.R.s, and S.G.

Almost all operations are local. Fig. 2 summarizes them in the 2D case: tetrahedra are replaced by triangles. (The latter can be seen as cross-sections of the former, e.g. the cross-section inside a torus is an annulus or a disconnected set of triangles.)

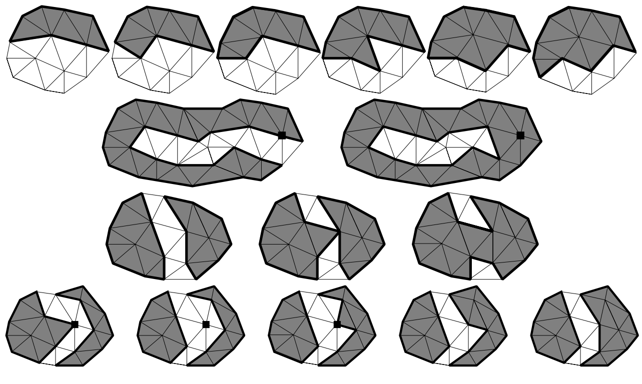

Figure 2: Local operations in the 2D case. (Tetrahedra are replaced by triangles.) From top to bottom: Shelling (Sh.), Topology Extension (T.E.), Force-Repair (F.R.) and Shrink-Grow (S.G.). Every line shows successive values of O in gray. Sh. adds one tetrahedron at once to O and g does not change. T.E. adds all tetrahedra around a vertex to O so that g can increase. F.R. adds tetrahedra, a singularity is first created and then removed. S.G. first removes then adds tetrahedra to O.

Shelling (Sh.) adds one tetrahedron at once to O such that ∂O is always manifold and the number of connected component(s) of O does not change. The first line of Fig. 2 shows an example. In more details, Sh. builds a tetrahedron series $O_0, O_1, \cdots O_n$ such that $O_{i+1} = O_i \cup \{\Delta_i\}$ and the tetrahedron $\Delta_i \in F \setminus O_i$ has at least one triangle face in ∂O_i (if $O_i \neq \emptyset$) and every ∂O_i is manifold. We first consider Δ_i candidates that have the largest $f(\Delta_i)$, such that O globally grows from the most confident free-space to the less confident one. Once Δ_0 is selected, the other Δ_i are selected in a growing neighborhood of Δ_0 thanks to a priority queue. In practice, Sh. is fast and adds almost all tetrahedra in O. A local Shelling is possible by specifying the first added tetrahedron Δ_0 as input. More details can be found in algorithm 1 of [22].

However, Sh. is not sufficient since it cannot change g ([20]), the genus of ∂O. If the genus of the true surface is not 0, e.g. if the camera path (the $\mathbf{c}_j$ series) includes a loop around a high building, g should increase.

Topology Extension (T.E.) builds $O_1 = O_0 \cup T_{\mathbf{v}}$ if a vertex $\mathbf{v} \in c(\partial O_0)$ and $T_{\mathbf{v}} \subseteq F$. It is successful if ∂O_1 is manifold. Now g can increase as shown in the second line of Fig. 2, since T.E. adds to O several tetrahedra at once.

However, T.E. can also generate spurious handles as shown in the left of Fig. 15. Thus another operation is needed; it takes as input a tetrahedron set $H \subseteq F \setminus O$ that we would like to add to O. For example, H is a cross section of a handle that we would like to remove.

Force-and-Repair (F.R.) builds a series $O_0, O_1, \cdots O_n$ defined as follows. First we "force" $O_1 = O_0 \cup H$ and ignore singular vertices in ∂O_1. Then we try to "repair". Let s_i be the number of singular vertices of ∂O_i. We select a tetrahedron $\Delta_i \in F \setminus O_i$ having triangle face(s) in ∂O_i and set $O_{i+1} = O_i \cup \{\Delta_i\}$ while $s_{i+1} \leq s_i$. F.R. is successful if $s_n = 0$. More details on "repair" can be found in Algorithm 3 of [22] (using notation $G = H$).

The third line of Fig. 2 shows a F.R. example.

Steps *Critical Handle Removals* (C.H.R.s) of [22] and [25] select several sets H and try to add each of them to O by using F.R. The C.H.R.s remove spurious handles (Sec. 1.3.2) that are visually important. More details can be found in Algorithm 2 of [22] and Sec. III.C of [25].

Now we note that all operations above (Sh., T.E., F.R.) and their compositions (C.H.R.s) are O growing in the free-space F. By combining them, we obtain a descent method for minimizing function $-f(O)$ such that ∂O is manifold, which can get stuck to a local extremum. The following operation removes some tetrahedra from O (Thus $f(O)$ decreases temporarily.) to kick the algorithm out of local extrema.

Shrink-and-Grow (S.G.) builds a series that is not 100% growing as shown in the last line of Fig. 2. First O shrinks: $O_1 = O_0 \setminus T_{\mathbf{v}}$ if a vertex $\mathbf{v} \in c(\partial O_0)$ and ∂O_1 is manifold. Then O grows: we try local Shelling $O_1, O_2, \cdots O_n$ started by a tetrahedron $\Delta \in F \setminus O_0$ that has the vertex $\mathbf{v}$. S.G. is successful if $f(O_n) \geq f(O_0)$. In practice, S.G. is tried at several selected vertices $\mathbf{v} \in c(\partial O)$. More details can be found in algorithm of Sec. III.B of [25]. (See also Appendix D.) In the paper remainder, we only need this composition of several S.G. operations and directly use notation S.G. for the composition.

3. Directed edge-based manifold test

Manifold test at a vertex is important since it is used in all operations in Sec. 2.2.2 except Shelling. (Shelling has its own manifold test.) Sec. 3.1 and 3.2 are reminders about simplex orientation and orientability of T. Then Sec. 3.3 introduces a method that checks if a vertex is regular in ∂O. It is based on a theorem, whose proof is in Sec. 3.4.

3.1. Simplex orientation

According to [48], an *orientation* of a simplex $\mathbf{v}_0\mathbf{v}_1 \cdots \mathbf{v}_k$ is an equivalence class of its vertex orderings such that $(\mathbf{v}_0, \mathbf{v}_1, \cdots \mathbf{v}_k)$ and $(\mathbf{v}_{\pi 0}, \mathbf{v}_{\pi 1}, \cdots \mathbf{v}_{\pi k})$ are equivalent iff the permutation π is even (using shortened notation $\pi i = \pi(i)$). We remind that every permutation is a product of transposition(s), and a permutation is even if it is the product of an even number of transpositions. Furthermore, if there are distinct integers a and b such that $\pi a = b$ and $\pi b = a$ and $\pi c = c$ where $c \in \mathbb{N} \setminus \{a, b\}$, π is a transposition and we use notation $\pi = (a\ b)$.

Let A_{k+1} be the set of the even permutations of $\{0, \cdots k\}$. Thus

$$\overline{(\mathbf{v}_0, \mathbf{v}_1, \cdots, \mathbf{v}_k)} = \{(\mathbf{v}_{\pi 0}, \mathbf{v}_{\pi 1}, \cdots \mathbf{v}_{\pi k}), \pi \in A_{k+1}\} \tag{2}$$

is the equivalence class of $(\mathbf{v}_0, \mathbf{v}_1, \cdots \mathbf{v}_k)$ and we have

$$\pi \in A_{k+1} \iff \overline{(\mathbf{v}_0, \mathbf{v}_1, \cdots \mathbf{v}_k)} = \overline{(\mathbf{v}_{\pi 0}, \mathbf{v}_{\pi 1}, \cdots \mathbf{v}_{\pi k})}. \tag{3}$$

Here are examples using $\pi_1 = (0\ 1) \circ (2\ 3)$ and $\pi_2 = (0\ 2) \circ (0\ 1)$:

$$(0\ 1) \notin A_4 \Rightarrow \overline{(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)} \neq \overline{(\mathbf{v}_1, \mathbf{v}_0, \mathbf{v}_2, \mathbf{v}_3)} \tag{4}$$

$$\pi_1 \in A_4 \Rightarrow \overline{(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)} = \overline{(\mathbf{v}_1, \mathbf{v}_0, \mathbf{v}_3, \mathbf{v}_2)} \tag{5}$$

$$\pi_2 \in A_4 \Rightarrow \overline{(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)} = \overline{(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_0, \mathbf{v}_3)} \tag{6}$$

$$\overline{(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)} = \overline{(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_0)} = \overline{(\mathbf{v}_2, \mathbf{v}_0, \mathbf{v}_1)}. \tag{7}$$

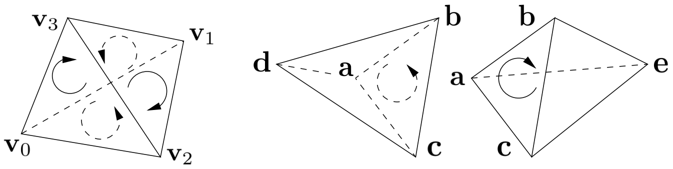

Figure 3: Left: triangle orientations induced by the tetrahedron orientation $\overline{(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)}$. Front triangles have continuous arrows, back triangles have dotted arrows. Right: two adjacent tetrahedra with consistent orientations $\overline{(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})}$ and $\overline{(\mathbf{b}, \mathbf{a}, \mathbf{c}, \mathbf{e})}$. (The triangle **abc** is duplicated for readability.) The arrows in **abc** rotate in opposite directions.

We note that there are exactly two possible orientations if $k \geq 1$: $\overline{(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \cdots \mathbf{v}_k)}$ and $\overline{(\mathbf{v}_1, \mathbf{v}_0, \mathbf{v}_2, \cdots \mathbf{v}_k)}$. Here we swap $\mathbf{v}_0$ and $\mathbf{v}_1$ as in Eq. 4, but other vertices can be swapped. We also note that this orientation definition does not require vertex coordinates, although it is related to orientation in vector spaces: function $(\mathbf{v}_0, \cdots, \mathbf{v}_k) \mapsto \det\begin{pmatrix} \mathbf{v}_0 & \cdots & \mathbf{v}_k \\ 1 & \cdots & 1 \end{pmatrix}$ is constant in an equivalence class and changes its sign in the other class.

3.2. Orientability of T

We first provide the definition of induced orientation since it helps to understand orientability. An orientation $\overline{(\mathbf{v}_0, \mathbf{v}_1, \cdots \mathbf{v}_k)}$ induces orientations of faces of simplex $\mathbf{v}_0\mathbf{v}_1 \cdots \mathbf{v}_k$ as follows: the *induced orientation* of $\mathbf{v}_{\pi 1} \cdots \mathbf{v}_{\pi k}$ is $\overline{(\mathbf{v}_{\pi 1}, \cdots \mathbf{v}_{\pi k})}$ if $\pi \in A_{k+1}$. For example, a triangle orientation $\overline{(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2)}$ induces edge orientations $\overline{(\mathbf{v}_1, \mathbf{v}_2)}$, $\overline{(\mathbf{v}_2, \mathbf{v}_0)}$ and $\overline{(\mathbf{v}_0, \mathbf{v}_1)}$ thanks to Eq. 7. Similarly, a tetrahedron orientation $\overline{(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)}$ induces triangle orientations $\overline{(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)}$, $\overline{(\mathbf{v}_2, \mathbf{v}_0, \mathbf{v}_3)}$, $\overline{(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_3)}$ and $\overline{(\mathbf{v}_2, \mathbf{v}_1, \mathbf{v}_0)}$. The left of Fig. 3 shows these induced orientations by using a standard convention: the orientation $\overline{(\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c)}$ is represented by an arrow that rotates in the same direction as $\mathbf{v}_a \to \mathbf{v}_b \to \mathbf{v}_c \to \mathbf{v}_a$.

Now assume that we have orientations $\overline{(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)}$ and $\overline{(\mathbf{v}'_0, \mathbf{v}'_1, \mathbf{v}'_2, \mathbf{v}'_3)}$ of adjacent tetrahedra Δ and Δ', respectively. We say that Δ and Δ' are *consistently oriented* if there are $\pi \in A_4$ and $\pi' \in A_4$ such that $(\mathbf{v}_{\pi 0}, \mathbf{v}_{\pi 1}, \mathbf{v}_{\pi 2}) = (\mathbf{v}'_{\pi' 1}, \mathbf{v}'_{\pi' 0}, \mathbf{v}'_{\pi' 2})$ ([38]). Fig. 3 shows tetrahedra **abcd** and **abce** with consistent orientations $\overline{(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})}$ and $\overline{(\mathbf{b}, \mathbf{a}, \mathbf{c}, \mathbf{e})}$. We check[1] that they induce different orientations $\overline{(\mathbf{b}, \mathbf{a}, \mathbf{c})}$ and $\overline{(\mathbf{a}, \mathbf{b}, \mathbf{c})}$ for their shared triangle face **abc**. Intuitively, **d** and **e** "see" **abc** in opposite orientations.

Last T is *orientable* ([38, 4], see also **Proofs.pdf** in the supplementary material), i.e.

Theorem 1. *There are orientation choices for all tetrahedra in T such that every pair of adjacent tetrahedra in T is consistently oriented.*

3.3. Manifold test

We first introduce notations and definitions to check that a vertex $\mathbf{v} \in c(\partial O)$ is regular in ∂O. Theorem 1 provides an orientation $o(\Delta)$ for every tetrahedron $\Delta \in T$. This is done once

[1] using equalities $\overline{(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d})} = \overline{(\mathbf{d}, \mathbf{b}, \mathbf{a}, \mathbf{c})}$ and $\overline{(\mathbf{b}, \mathbf{a}, \mathbf{c}, \mathbf{e})} = \overline{(\mathbf{e}, \mathbf{a}, \mathbf{b}, \mathbf{c})}$.

before all manifold tests. We consider a set of directed edges:

$$D = \bigcup_{\Delta \in O} \{(\mathbf{v}_2, \mathbf{v}_3), (\mathbf{v}, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) \in o(\Delta), \mathbf{v}\mathbf{v}_2\mathbf{v}_3 \in \partial O\}. \tag{8}$$

If $D = \{(\mathbf{q}_1, \mathbf{q}_2), \cdots (\mathbf{q}_{m-1}, \mathbf{q}_m), (\mathbf{q}_m, \mathbf{q}_1)\}$ and $m \geq 3$ and $\mathbf{q}_1 \cdots \mathbf{q}_m$ are m distinct vertices, D is *a directed cycle*.

3.3.1. Theorem

Our test is based on the following theorem.

Theorem 2. *The vertex $\mathbf{v} \in c(\partial O)$ is regular in ∂O iff D is a directed cycle.*

We check Theorem 2 in simple examples to provide intuition. In the first example, a tetrahedron $\mathbf{vabc} \in O$ and $o(\mathbf{vabc}) = \overline{(\mathbf{v}, \mathbf{a}, \mathbf{b}, \mathbf{c})}$ and triangles $\mathbf{vbc}$, $\mathbf{vca}$, $\mathbf{vab}$ are in ∂O. Since

$$\{(\mathbf{v}, \mathbf{a}, \mathbf{b}, \mathbf{c}), (\mathbf{v}, \mathbf{b}, \mathbf{c}, \mathbf{a}), (\mathbf{v}, \mathbf{c}, \mathbf{a}, \mathbf{b})\} \subset o(\mathbf{vabc}), \tag{9}$$

$\mathbf{vabc}$ has the following contribution to D: it adds directed edges $(\mathbf{b}, \mathbf{c})$, $(\mathbf{c}, \mathbf{a})$ and $(\mathbf{a}, \mathbf{b})$ to D. If there is no other tetrahedron in O with the vertex $\mathbf{v}$ and a triangle face(s) in ∂O, we see that D is a directed cycle and $\mathbf{v}$ is regular. Otherwise, D is not a directed cycle (It has supplementary directed edges.) and $\mathbf{v}$ is not regular. (There are other triangles t such that $\mathbf{v} \in t \in \partial O$.)

In the second example, tetrahedra $\mathbf{vuab}$, $\mathbf{vubc}$, $\mathbf{vuca}$ are in O and triangles $\mathbf{vab}$, $\mathbf{vbc}$, $\mathbf{vca}$ are in ∂O. We choose orientations $o(\mathbf{vuab}) = \overline{(\mathbf{v}, \mathbf{u}, \mathbf{a}, \mathbf{b})}$, $o(\mathbf{vubc}) = \overline{(\mathbf{v}, \mathbf{u}, \mathbf{b}, \mathbf{c})}$, $o(\mathbf{vuca}) = \overline{(\mathbf{v}, \mathbf{u}, \mathbf{c}, \mathbf{a})}$ and check orientation consistency, e.g. we have $o(\mathbf{vuab}) = \overline{(\mathbf{u}, \mathbf{v}, \mathbf{b}, \mathbf{a})}$ and $o(\mathbf{vubc}) = \overline{(\mathbf{v}, \mathbf{u}, \mathbf{b}, \mathbf{c})}$ for $\mathbf{vuab}$ and $\mathbf{vubc}$. Thus the contribution of the three tetrahedra to D is the following: they add directed edges $(\mathbf{a}, \mathbf{b})$, $(\mathbf{b}, \mathbf{c})$ and $(\mathbf{c}, \mathbf{a})$ to D. Now we do the same observations as in the first example.

3.3.2. Implementation

Here we summarize the implementation of the manifold test. For every vertex $\mathbf{w} \in c(T)$, $T_\mathbf{w}$ is stored in a table. We store D in a table E of vertices where the i-th directed edge of D has its start-vertex in E[2i] and its end-vertex in E[2i+1]. For every tetrahedron $\Delta = \mathbf{v}_0^\Delta \mathbf{v}_1^\Delta \mathbf{v}_2^\Delta \mathbf{v}_3^\Delta \in T$, we store a vertex ordering $(\mathbf{v}_0^\Delta, \mathbf{v}_1^\Delta, \mathbf{v}_2^\Delta, \mathbf{v}_3^\Delta)$ such that $o(\Delta) = \overline{(\mathbf{v}_0^\Delta, \mathbf{v}_1^\Delta, \mathbf{v}_2^\Delta, \mathbf{v}_3^\Delta)}$. Thus,

$$D = \bigcup_{\Delta \in O} \{(\mathbf{v}_{\pi 2}^\Delta, \mathbf{v}_{\pi 3}^\Delta), \pi \in A_4, \mathbf{v}_{\pi 0}^\Delta = \mathbf{v}, \mathbf{v}_{\pi 0}^\Delta \mathbf{v}_{\pi 2}^\Delta \mathbf{v}_{\pi 3}^\Delta \in \partial O\}. \tag{10}$$

Thanks to Eq. 10, for every $\Delta \in T_\mathbf{v} \cap O$ and $\pi \in A_4$, we collect in E the edges using a C-like line as

if ($\mathbf{v}_{\pi 0}^\Delta$==$\mathbf{v}$ && $\mathbf{v}_{\pi 0}^\Delta \mathbf{v}_{\pi 2}^\Delta \mathbf{v}_{\pi 3}^\Delta \in \partial O$) { E[n++]=$\mathbf{v}_{\pi 2}^\Delta$; E[n++]=$\mathbf{v}_{\pi 3}^\Delta$; }

after initializing n=0. Last we check that D is a directed cycle using a naive sorting algorithm: swap directed edges in the table E to form path E[0]E[1], E[2]E[3] with E[1]==E[2], E[4]E[5] with E[3]==E[4] and so on.

Appendix A gives more details about the algorithm. This appendix also details a previous test algorithm of [22] that is compared in our experiments. The efficiency of the new test is based on facts that are not considered before: small D size and consistent orientations in T. Thus the naive sorting in E is fast in spite of its complexity (quadratic in the size of D).

3.4. Proof of Theorem 2

This paragraph only provides the principle of the proof; the details are in Appendix B. The proof may look difficult at first glance, but it should be stressed that the corresponding algorithm is not: Algorithm 3 only reads directed edges and naively sorts them in a table to detect a directed cycle.

The undirected version of D is

$$U = \{\mathbf{ab}, (\mathbf{a}, \mathbf{b}) \in D\}. \tag{11}$$

We note that U is a cycle (with undirected edges) if D is a directed cycle (with directed edges), but the converse can be false.

First it is not difficult to see that

Lemma 1. *U is the set of the $\mathbf{v}$-opposite edges in the triangles of ∂O: $U = \{\mathbf{ab}, \mathbf{abv} \in \partial O\}$.*

Since $\mathbf{v}$ is regular iff $\{\mathbf{ab}, \mathbf{abv} \in \partial O\}$ is a cycle,

Lemma 2. *The vertex $\mathbf{v}$ is regular in ∂O iff U is a cycle.*

The easy part of Theorem 2's proof is the sufficient condition "D is a directed cycle $\Rightarrow$ $\mathbf{v}$ is regular" thanks to Lemma 2, which simplifies this to "D is a directed cycle $\Rightarrow$ U is a cycle", which in turn is straightforward. The necessary condition "$\mathbf{v}$ is regular $\Rightarrow$ D is a directed cycle" is more difficult although Lemma 2 simplifies this to "U is a cycle $\Rightarrow$ D is a directed cycle". The main part of the proof is to show that the consistent orientations in T imply good edge directions in D; this is done by studying consistent orientations of the tetrahedra sharing a common edge face $\mathbf{vu} \in c(\partial O)$.

4. Repair by analyzing the surface singularities

Sec. 4.3 presents a new repair operation that is guided by an analysis of the surface singularities to be removed. The analysis is done by using two Theorems in Sec. 4.2, which count the connected components of adjacency graphs of tetrahedra. We first extend T (Sec. 4.1) to avoid special cases in the Theorems.

4.1. Abstract extension T^∞ of T

To avoid special cases in statements and proofs, i.e. if $c(\partial T)$ includes singular vertices of ∂O, we extend T into T^∞ such that $\partial T^\infty = \emptyset$ like [4, 19].

First we replace every vertex of $c(T)$ by an integer that identifies the vertex. Then every simplex of $c(T)$ with 2,3,4 vertices is replaced by a set of 2,3,4 integers that identifies its vertices, respectively. (e.g. an edge becomes the set of the two integers of its vertices.) The relation "is a face of" (inclusion) does not change. Bold fonts and words vertex/edge/triangle/tetrahedron are also used for this new "abstract" simplex definition.

Then we extend T. Let $\mathbf{v}_\infty$ be an integer that is different to all vertices of $c(T)$. For every triangle $\mathbf{abc} \in \partial T$, we create a new tetrahedron $\mathbf{abcv}_\infty$ by adding $\mathbf{v}_\infty$ to integer set $\mathbf{abc}$. Let

$$T^\infty = T \cup \{\mathbf{abcv}_\infty, \mathbf{abc} \in \partial T\}. \tag{12}$$

Now $T \subset T^\infty$ and every triangle in $c(T^\infty)$ is a face of exactly two tetrahedra in T^∞. Fig. 4 shows $\mathbf{v}_\infty$ and T^∞. The tetrahedron sets in T^∞ that include a vertex $\mathbf{v}$ or an edge $\mathbf{ab}$ in $c(T^\infty)$ are

$$T_\mathbf{v}^\infty = \{\Delta \in T^\infty, \mathbf{v} \in \Delta\} \text{ and } T_\mathbf{ab}^\infty = \{\Delta \in T^\infty, \mathbf{ab} \subseteq \Delta\}. \tag{13}$$

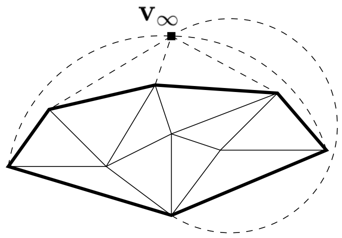

Figure 4: $\mathbf{v}_\infty$ and T^∞ in the 2D case. (Tetrahedra are drawn as triangles, triangles are drawn as edges.) The tetrahedra in T have continuous edges, those in $T^\infty \setminus T$ have dotted edges. The bold edges form ∂T.

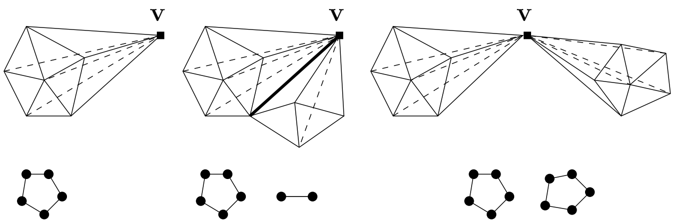

Figure 5: $T^\infty_{\mathbf{v}} \setminus O$ (top) and the corresponding $g^O_{\mathbf{v}}$ restricted to $T^\infty_{\mathbf{v}} \setminus O$ (bottom) in three cases. Case 1 (left): $\mathbf{v}$ is regular in ∂O and $g^O_{\mathbf{v}}$ has one connected component in $T^\infty_{\mathbf{v}} \setminus O$. Cases 2 and 3 (middle and right): $\mathbf{v}$ is singular in ∂O and $g^O_{\mathbf{v}}$ has two connected component in $T^\infty_{\mathbf{v}} \setminus O$. The bold edge (middle of top) is singular in ∂O.

4.2. Vertex singularities and edge singularities

Let vertex $\mathbf{v} \in c(\partial O)$ and $g_{\mathbf{v}}$ be the adjacency graph of the tetrahedra in $T^\infty_{\mathbf{v}}$. (There is a bijection between the vertices of $g_{\mathbf{v}}$ and the tetrahedra in $T^\infty_{\mathbf{v}}$, every edge of $g_{\mathbf{v}}$ links two vertices if their corresponding tetrahedra are adjacent.) Let $g^O_{\mathbf{v}}$ be the graph obtained from $g_{\mathbf{v}}$ by removing every edge between a tetrahedron in $T^\infty_{\mathbf{v}} \cap O$ and a tetrahedron in $T^\infty_{\mathbf{v}} \setminus O$. Thus

Lemma 3. *Every connected component of $g^O_{\mathbf{v}}$ is included in $T^\infty_{\mathbf{v}} \cap O$ or included in $T^\infty_{\mathbf{v}} \setminus O$. There is at least one component in $T^\infty_{\mathbf{v}} \cap O$ and at least one component in $T^\infty_{\mathbf{v}} \setminus O$.*

Proof. Since all edges between $T^\infty_{\mathbf{v}} \cap O$ and $T^\infty_{\mathbf{v}} \setminus O$ are removed from $g^O_{\mathbf{v}}$, the first assertion is true. There is a triangle $t \in \partial O$ including $\mathbf{v}$; t is a face of a tetrahedron in $T^\infty_{\mathbf{v}} \cap O$ and is a face of another one in $T^\infty_{\mathbf{v}} \setminus O$. Thus the second assertion is true. □

Theorem 3. *The vertex $\mathbf{v}$ is singular in ∂O iff $g^O_{\mathbf{v}}$ has at least three connected components. (See Fig. 5, see also Fig. 4 of [19].)*

Proof. Thanks to Lemma 3, $g^O_{\mathbf{v}}$ has at least two connected components. Thanks to Theorem 3 of [19], $\mathbf{v} \in c(\partial O)$ is singular iff $g^O_{\mathbf{v}}$ has at least three connected components. □

Similarly, let edge $\mathbf{ab} \in c(\partial O)$ and $g_{\mathbf{ab}}$ be the adjacency graph of the tetrahedra in $T^\infty_{\mathbf{ab}}$. Let $g^O_{\mathbf{ab}}$ be the graph obtained from $g_{\mathbf{ab}}$ by removing every edge between a tetrahedron in $T^\infty_{\mathbf{ab}} \cap O$ and a tetrahedron in $T^\infty_{\mathbf{ab}} \setminus O$. Thus

Lemma 4. *The graph $g_{\mathbf{ab}}$ is a cycle. The graph $g^O_{\mathbf{ab}}$ has exactly $2n$ connected components: n in $T^\infty_{\mathbf{ab}} \cap O$ and n in $T^\infty_{\mathbf{ab}} \setminus O$ (Fig. 6).*

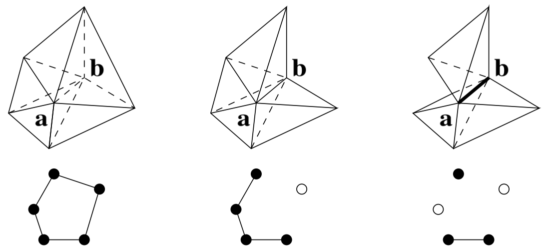

Figure 6: $T^\infty_{\mathbf{ab}} \cap O$ (top) and the corresponding $g^O_{\mathbf{ab}}$ (bottom) in three cases. Case 1 (left): $\mathbf{ab} \notin c(\partial O)$ and $g^O_{\mathbf{ab}} = g_{\mathbf{ab}}$ has one connected component. Case 2 (middle): $\mathbf{ab} \in c(\partial O)$ is not singular in ∂O and $g^O_{\mathbf{ab}}$ has two connected components. Case 3 (right): $\mathbf{ab} \in c(\partial O)$ is singular in ∂O and $g^O_{\mathbf{ab}}$ has four connected components.

Proof. Since T^∞ is an (extended) 3D Delaunay triangulation,

$$T^\infty_{\mathbf{ab}} = \{\mathbf{abc}_0\mathbf{c}_1, \mathbf{abc}_1\mathbf{c}_2, \mathbf{abc}_2\mathbf{c}_3, \cdots \mathbf{abc}_k\mathbf{c}_0\} \tag{14}$$

where the edges $\mathbf{c}_0\mathbf{c}_1, \cdots \mathbf{c}_{k-1}\mathbf{c}_k$, $\mathbf{c}_k\mathbf{c}_0$ form a cycle. Thus $g_{\mathbf{ab}}$ is a cycle. Furthermore, all edges between $T^\infty_{\mathbf{ab}} \cap O$ and $T^\infty_{\mathbf{ab}} \setminus O$ are removed in $g^O_{\mathbf{ab}}$. Thus every connected component of $g^O_{\mathbf{ab}}$ is a path in $T^\infty_{\mathbf{ab}} \cap O$ or in $T^\infty_{\mathbf{ab}} \setminus O$. Since every path in $T^\infty_{\mathbf{ab}} \cap O$ is before a path in $T^\infty_{\mathbf{ab}} \setminus O$ in the cycle (or conversely), the numbers of components in $T^\infty_{\mathbf{ab}} \cap O$ and $T^\infty_{\mathbf{ab}} \setminus O$ are the same. □

We say that the edge $\mathbf{ab}$ is *singular in ∂O* if $g^O_{\mathbf{ab}}$ has at least three connected components. Thus

Theorem 4. *Assume that the edge $\mathbf{ab}$ is singular in ∂O. Then $g^O_{\mathbf{ab}}$ has at least four connected components (Fig. 6). Furthermore, both vertices $\mathbf{a}$ and $\mathbf{b}$ are singular in ∂O.*

Proof. Thanks to Lemma 4, the first assertion is true. For the second assertion, we assume that $\mathbf{a}$ is regular in ∂O and show that $g^O_{\mathbf{ab}}$ has exactly two connected components. Thanks to Lemma 4, ∂O has exactly $2n$ triangles including $\mathbf{ab}$. (Such a triangle is between $T^\infty_{\mathbf{ab}} \cap O$ and $T^\infty_{\mathbf{ab}} \setminus O$.) Since $\mathbf{a}$ is regular, there are exactly two triangles in ∂O having the edge $\mathbf{ab}$ ([19]). Thus $n = 1$ and $g^O_{\mathbf{ab}}$ has exactly two connected components. □

4.3. Method

Singular vertices appears while adding to O a tetrahedron set $H \subseteq F \setminus O$. Our repair tries to remove them by adding to O other tetrahedra in $F \setminus O$ without creating new singular vertices.

4.3.1. Principle

If $\mathbf{ab}$ is a singular edge of ∂O, $g^O_{\mathbf{ab}}$ has at least two connected components in $T^\infty_{\mathbf{ab}} \setminus O$ (right of Fig. 6). We can add to O such a connected component if it is included in F and no additional singular vertex appears. Then the number of connected components of $g^O_{\mathbf{ab}}$ decreases (middle of Fig. 6), and we progress toward a non-singular $\mathbf{ab}$ thanks to Theorem 4. If $\mathbf{ab}$ becomes non-singular, vertices $\mathbf{a}$ or/and $\mathbf{b}$ can become non-singular too.

If $\mathbf{a}$ is a singular vertex of ∂O, $g^O_{\mathbf{a}}$ has at least one connected component in $T^\infty_{\mathbf{a}} \setminus O$ (e.g. two on the right of Fig. 5). Similarly, we can add to O such a connected component if it is included in F and no additional singular vertex appears. Then the number of connected components of $g^O_{\mathbf{a}}$ decreases (left of Fig. 5) and we progress toward a non-singular $\mathbf{a}$ thanks to Theorem 3. If $T^\infty_{\mathbf{a}} \setminus O$ becomes empty, $\mathbf{a}$ become non-singular since $\mathbf{a} \notin c(\partial O)$.

At first glance, the adds near singular edges are useless since

- a singular edge is a special case of a pair of singular vertices according to Theorem 4 and
- ∂O is manifold iff its vertices are non-singular (Sec. 2.1).

This is wrong: there are cases where a singular edge $\mathbf{ab}$ is only removable by an add near the edge since the adds near vertices $\mathbf{a}$ and $\mathbf{b}$ are impossible. For example, assume that $T^\infty_{\mathbf{ab}}$ has four tetrahedra: $\mathbf{abc}_0\mathbf{c}_1 \in T \setminus F$, $\mathbf{abc}_1\mathbf{c}_2 \in O$, $\mathbf{abc}_2\mathbf{c}_3 \in F \setminus O$ and $\mathbf{abc}_3\mathbf{c}_0 \in O$. Here $\{\mathbf{abc}_2\mathbf{c}_3\}$ is a connected component of $g^O_{\mathbf{ab}}$ that can be added to O. If $g^O_{\mathbf{a}}$ has only one connected component in $T^\infty_{\mathbf{a}} \setminus O$, this component cannot be added to O since it includes $\mathbf{abc}_0\mathbf{c}_1 \notin F$ (and similarly for $g^O_{\mathbf{b}}$).

4.3.2. Algorithm

First we need an auxiliary function ReduceSingularity, which is detailed in Algorithm 1. Let V be the set of the singular vertices of ∂O. ReduceSingularity adds to O a tetrahedron set $C \subseteq F \setminus O$ if this does not add a new vertex to V. In this case, V can decrease and C is added to A. (A is used at the repair end.)

Algorithm 1. *ReduceSingularity(C,V,O,A)*

Let $\{\mathbf{v}_0, \mathbf{v}_1, \cdots \mathbf{v}_c\}$ be the vertex set of C;
Let b^0_i be true iff $\mathbf{v}_i \notin V$;
$O \leftarrow O \cup C$;
Let b^1_i be true iff $\mathbf{v}_i$ is regular in ∂O; // use manifold test
for each $i \in \{0, 1, \cdots c\}$ // detect failures
  if ($b^0_i = true$ && $b^1_i = false$) { $O \leftarrow O \setminus C$; **return** 0; }
for each $i \in \{0, 1, \cdots c\}$
  if ($b^0_i = false$ && $b^1_i = true$) $V \leftarrow V \setminus \{\mathbf{v}_i\}$;
$A \leftarrow A \cup C$;
return 1; // success

end

Then we detail our Repair2 method in Algorithm 2 (in C style). It takes as input O and the set H that was just added to O. (∂O was manifold before that.) It use a set E where the singular edges are searched. It also use a set A that collects the new tetrahedra added to O; A is used to restore O in case of failure or to improve O in case of success. Repair2 examines singular edges (special cases) before singular vertices. The set V can only decrease and Repair2 is successful iff $V = \emptyset$. Fig. 7 shows an example of Repair2. The main loop is not infinite, since its number of iterations is less than the number of tetrahedra in $F \setminus O$ having a vertex in $c(H)$. In our implementation, the sets H, V, E, C and A are stored in tables.

Algorithm 2. *Repair2*

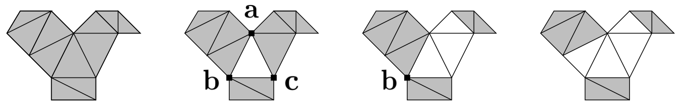

Figure 7: Repair2 in the 2D case. O is white and $F \setminus O$ is the set of triangles in gray. From left to right: ∂O is manifold, add $H = \{\mathbf{abc}\}$ to O and obtain the singular vertex set $V = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$, add to O a connected components of $g^O_{\mathbf{a}}$ included in $F \setminus O$ and obtain $V = \{\mathbf{b}\}$, add to O a connected components of $g^O_{\mathbf{b}}$ included in $F \setminus O$ and obtain $V = \emptyset$.

$V = \emptyset$;
for each vertex $\mathbf{v} \in c(H)$
  if ($\mathbf{v}$ is singular in ∂O) $V \leftarrow V \cup \{\mathbf{v}\}$; // use manifold test
$E = \emptyset$; // E includes all singular edges
for each vertices $\mathbf{a}$ and $\mathbf{b}$ in V
  if ($\mathbf{ab} \in c(T)$) $E \leftarrow E \cup \{\mathbf{ab}\}$;
$A = H$; // A is the set of the tetrahedra added to O
do {
  IsImproved= 0;
  for each edge $\mathbf{ab} \in E$
    if ($\mathbf{a} \in V$ && $\mathbf{b} \in V$) {
      Let $C_1, \cdots, C_k$ be the connected components of $g^O_{\mathbf{ab}}$;
      if (there is a C_i such that $C_i \subseteq F \setminus O$ &&
        ReduceSingularity(C_i, V, O, A)) IsImproved= 1;
    } **else** $E \leftarrow E \setminus \{\mathbf{ab}\}$;
  for each vertex $\mathbf{a} \in V$ {
    Let $C_1, \cdots, C_k$ be the connected components of $g^O_{\mathbf{a}}$;
    if (there is a C_i such that $C_i \subseteq F \setminus O$ &&
      ReduceSingularity(C_i, V, O, A)) IsImproved= 1;
  }
} **while** (IsImproved)
if ($V \neq \emptyset$) { $O \leftarrow O \setminus A$; **return** 0; } // restore O if failure
// complete O using Shelling if success
for each tetrahedron $\Delta \in A$
  for each 4-neighbor tetrahedron Δ' of Δ
    if ($\Delta' \in F \setminus O$), try a Shelling started from Δ';
return 1;

end

Last we explain why Repair (Algorithm 3 of [22]) is less efficient than Repair2 above. Repair ignores the structure of the singularities (in terms of connected components of adjacent tetrahedra) and does not distinguish singular edges from general singular vertices; it blindly adds tetrahedra one-by-one to O such that no new singular vertices appears. Furthermore, Repair is time consuming since it applies manifold tests before and after adding every tetrahedron. Thanks to V, Repair2 only needs to apply manifold tests after adding packs of tetrahedra.

5. Lowered genus during surface reconstruction

Sec. 5.1 shortly summarizes the method of [22] and explains how the genus g of ∂O evolves during the computation. Sec. 5.2

and 5.3 present two methods estimating ∂O with a smaller g. We remember Euler's formula ([5]) if ∂O is connected:

$$2(1-g) = v - e + t \tag{15}$$

where v, e and t are the numbers of vertices, edges and triangles in $c(\partial O)$, respectively. If ∂O is not connected, its genus is the sum of genuses of its connected components.

5.1. Previous method and g evolution

The previous method of [22] has several steps defined from operations summarized in Sec. 2.2.2.

5.1.1. Step 1: Shelling plus Topology Extension

The set O is initialized by Sh., which tries to add first to O the tetrahedra that have the largest number of ray intersections. (The first tetrahedron in O maximizes f.) The resulting ∂O meets $g = 0$ although the true surface has a non-zero genus, e.g. if the camera path has loops around buildings. Then T.E. is tried on every vertex $\mathbf{v} \in c(\partial O)$. Now $g > 0$, but spurious handles occur.

5.1.2. Step 2: Critical Handle Removal

C.H.R. in Sec. 4.3 of [22] is applied to remove critical handles, i.e. spurious handles that the human eye cannot miss at the camera locations $\mathbf{c}_j$ where the images are taken. Here we provide useful details on C.H.R.

First we define *critical edges* $\mathbf{ab}$ for a given angle threshold $\alpha > 0$: $\mathbf{ab}$ is a face of a tetrahedron in $F \setminus O$ (which can be added to O), it is only included in tetrahedra in F, it is large enough such that angle $\widehat{\mathbf{ac}_j\mathbf{b}} > \alpha$. Thus the set of the critical edges is concisely defined by

$$L_\alpha = \{\mathbf{ab} \in c(F \setminus O) \setminus c(\partial T), T_{\mathbf{ab}} \subseteq F, \exists j, \widehat{\mathbf{ac}_j\mathbf{b}} > \alpha\}. \tag{16}$$

Then C.H.R. subdivides every critical edge by adding a Steiner vertex at the middle to create new tetrahedra, such that ∂O is still manifold. (A tetrahedron can be split in two new tetrahedra that are in O or F iff the original one is.) A Steiner vertex is an extra point that is not in the input cloud. Last F.R. tries to add to O a tetrahedron set H included in $T_\mathbf{v} \cap (F \setminus O)$, for every vertex $\mathbf{v}$ of the split edges.

Although critical handles are removed like this, our experiment shows that g increases a lot during this step.

5.1.3. Step 3: post-processing including Peak Removal

The post-processing improves ∂O thanks to prior knowledge of the scene. It includes a step Peak Removal (P.R), a Laplacian smoothing and a process to remove triangles in the sky. The two latters are not summarized since they do not change g. P.R. removes bad points in the surface. It is tried on every vertex $\mathbf{v} \in c(\partial O)$ such that the ring of the triangles in ∂O including $\mathbf{v}$ has a solid angle w in the $T^\infty_\mathbf{v} \setminus O$ side such that $w < w_0$. Then it add to O the set $T_\mathbf{v}$ if ∂O remains a manifold. (The process is similar in the $T^\infty_\mathbf{v} \cap O$ side.) It looks like T.E. without condition $T_\mathbf{v} \subseteq F$. Our experiment shows that g decreases thanks to P.R.

5.2. Method 1: cancel C.H.R. tentatives

The first idea that comes to mind is to cancel C.H.R. if g increases. Let δg be the g variation due to a successful F.R. operation during C.H.R. We also would like an efficient computation of δg without a large traversal of ∂O. Since T is implemented by the adjacency graph of the tetrahedra, large traversals in T should be avoided.

Let c be the number of the connected components of ∂O. By summing Eq. 15 for all connected components, we obtain

$$\delta v - \delta e + \delta t = 2(\delta c - \delta g) \tag{17}$$

where δv, δe, δt and δc are the variations of v, e, t and c due to a successful F.R. operation. Sec. 5.2.1 presents an efficient computation of $\delta v - \delta e + \delta t$. Then Sec. 5.2.2 presents a method to avoid the computation of δc which is not efficient.

5.2.1. Efficient computation of $\delta v - \delta e + \delta t$

Let $\tilde{H} \subseteq F \setminus O$ be all tetrahedra that are added to the initial O by a successful F.R. operation during C.H.R. (Note that $H \subseteq \tilde{H}$ by using the notation H in Sec. 2.2.2.) Let

$$J = (\partial O \setminus \partial(O \cup \tilde{H})) \cup (\partial(O \cup \tilde{H}) \setminus \partial O). \tag{18}$$

We note that the triangles in $\partial O \cap \partial(O \cup \tilde{H})$) do not contribute to δt. (They add 0 to δt.) Similarly, the vertices and edges of these triangles do not contribute to δv and δe. Thus we only need a traversal of J to find all the vertices/edges/triangles that contribute to $\delta v - \delta e + \delta t$.

Intuitively, we only need a traversal of $\tilde{H}$ to compute $\delta v - \delta e + \delta t$ since the boundary change $|J|$ is included in the volume change $|\tilde{H}|$. Indeed, Appendix C shows that

Lemma 5. *We have $\partial\tilde{H} = J$.*

Thanks to a traversal of $\tilde{H}$, we access to the triangles in $\partial\tilde{H}$ and count those of them that are in ∂O. We also count those of them that are in $\partial(O \cup \tilde{H})$ and obtain δt. Similarly, we access to the vertices and edges in $c(\partial\tilde{H})$ and count those of them that are in $c(\partial O)$ and those that are in $c(\partial(O \cup \tilde{H}))$, then we obtain δv and δe. This computation is efficient since $\tilde{H}$ is quite smaller than O and $O \cup \tilde{H}$ and their boundaries.

5.2.2. Avoid inefficient computation of δc

At first glance, we accept a successful F.R. operation iff $\delta g \leq 0$. Here δg is estimated using Eq. 17 and Sec. 5.2.1 and the computation of δc. Unfortunately, the δc computation is not efficient. Indeed, the top row of Fig. 8 shows that we cannot distinguish cases ($\delta g = -1, \delta c = 0$) and ($\delta g = 0, \delta c = 1$) by a traversal of the small set $\tilde{H}$; a traversal of large set $\partial(O \cup \tilde{H})$ should be done instead.

Then we use condition $\delta c - \delta g \geq 0$ instead of $\delta g \leq 0$ for three reasons. First both conditions are simultaneously true or simultaneously false in all cases of Fig. 8. Second, the computation of $\delta c - \delta g$ is efficient thanks to Eq. 17 and Sec. 5.2.1. Last $\delta g < 0$ "almost" implies $\delta c - \delta g \geq 0$ according to the following theorem (proof in Sec. 5.2.3).

Theorem 5. *The set $\tilde{H}$ is connected. If O is connected and ∂O is manifold and $\tilde{H} \cap O = \emptyset$, then $\delta c \geq -1$ (in most cases $\delta c \geq 0$).*

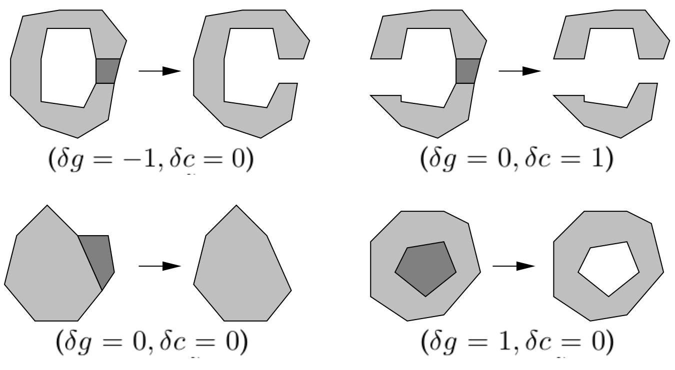

Figure 8: Increments $(\delta g, \delta c)$ in four cases where we apply $O \leftarrow O \cup \tilde{H}$. O is white, $\tilde{H}$ is dark gray, $T \setminus O$ is gray (light and dark). A gray region without (resp. with) a white hole represents a surface in $\mathbb{R}^3$ that has the sphere (resp. torus) topology. The case on the bottom right corner is rejected, the others are accepted.

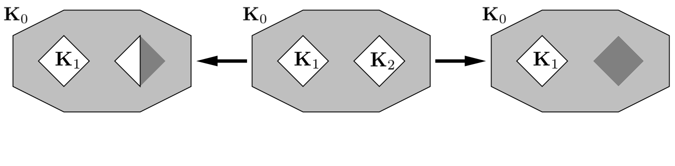

Figure 9: Examples for Theorem 5's proof if $k = 2$: cases $\delta c = 0$ (left) and $\delta c = -1$ (right). O is light gray, $\tilde{H}$ is dark gray. In the middle, $\mathbb{R}^3 \setminus (|O| \setminus |\partial O|)$ is white and has three connected components K_0 (unbounded), K_1 and K_2 (bounded). ∂O has three connected components C_0, C_1 and C_2. In the left and right, $\tilde{H} \subseteq K_2$ and $\partial(O \cup \tilde{H})$ has at least 2 connected components C_1 and C_0.

Indeed, the assumptions of Theorem 5 are "almost" meet during C.H.R. of [22] since the number of connected components of O (Note: this is not c.) is very low in practice.

5.2.3. Sketch proof of Theorem 5

The proof details are in file **Proofs.pdf** of the supplementary material. First $\tilde{H}$ is connected since all tetrahedra in H share a same vertex and $\tilde{H}$ is obtained by a continuous growing started from H ([22]). Second we see that O is a "solid" ([35]): it is like one piece of gruyere with one external boundary and k internal boundaries, each of them is the manifold boundary of a connected component K_i of $\mathbb{R}^3 \setminus (|O| \setminus |\partial O|)$. (See middle of Fig. 9.) Third we see that $\tilde{H}$ is included in a single K_i. (Since $\tilde{H}$ is connected and $\tilde{H} \cap O = \emptyset$.) Fourth, K_j is unchanged by the add of $\tilde{H}$ to O if $j \neq i$. Last there are at least k connected components of ∂O that are unchanged by the add of $\tilde{H}$ to O, and we conclude.

We note that $\delta c \geq 0$ in most cases according to Fig. 9, i.e. iff $\tilde{H}$ is strictly included in K_i. (We have $\delta c = -1$ iff $\tilde{H} = K_i$.)

5.3. Method 2: replace T.E. and C.H.R. operations

This method has the same start (Sh. defined by Algorithm 1 of [22]) and same end (Sec. 5.1.3) as the previous method in Sec. 5.1. T.E. is replaced by the step in Sec. 5.3.2, and C.H.R. is replaced by the step in Sec. 5.3.3. The genus g moderately increases thanks to these two new steps using ideas in Sec. 5.3.1.

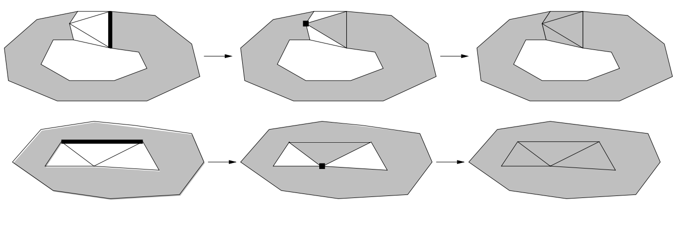

Figure 10: Increase (top) or decrease (bottom) of g done by C.E.R. O is gray and $F \setminus O$ is the set of triangles filled in white. Left: a critical edge e (bold edge) is detected in $c(\partial O)$. Middle: a singular vertex (black dot) appears when applying $O \leftarrow O \cup (T_e \cap (F \setminus O))$. Right: the singular vertex is removed by repairing (growing) O.

5.3.1. Principle

According to Sec. 5.1, g increases due to T.E. and C.H.R. We remind that T.E. is tried on every vertex in $c(\partial O)$, and F.R. (during C.H.R.) is tried on every vertex of all (split) critical edges. A reason of spurious handles and g excess is the lack of selection of locations were these operations are done.

We moderate the g increase as follows. First, T.E. is replaced by an operation that is applied only if it provides a visually non-negligible change of ∂O. In our case, this operation removes a critical edge from $c(\partial O)$. Second, the C.H.R. of [22] is replaced by a new C.H.R. that is more selective on set H of tetrahedra in $F \setminus O$ that it tries to add to O. Intuitively, H is the set of the tetrahedra in a handle that can be cut by a plane. Furthermore, $c(H)$ also includes a critical edge.

5.3.2. Critical Edge Removal (C.E.R.)

This step removes critical edges from $c(\partial O)$. For every edge $\mathbf{ab} \in L_\alpha \cap c(\partial O)$, F.R. (Sec. 2.2.2) is applied using $H = T_{\mathbf{ab}} \cap (F \setminus O)$. In more details, we first apply $O \leftarrow O \cup H$, then we try to remove the singular vertices in ∂O using Algorithm 3 of [22] with input $G = H$. If the final ∂O is not manifold, the **ab** removal fails and we restore O to its initial value.

This step acts as T.E. at locations (critical edges) that are selected by α, i.e. g can increase only if this provides a visually important change of ∂O. Fig. 10 shows that g can increase or even decrease by using C.E.R. Note that O is improved in both cases since $f(O)$ increases. Last we improve the result (without g change) thanks to Sh., i.e. we try to start a local Shelling from every tetrahedron $\Delta \in F \setminus O$.

5.3.3. Critical Handle Removal (C.H.R.)

A critical handle H meets several conditions ([25]). First, $H \subseteq F \setminus O$. Second H is "critical": there is a critical edge $\mathbf{ab} \in L_\alpha$ such that $\mathbf{ab} \in c(H)$. Third, there is a plane π such that $\pi \cap \mathbf{ab} \neq \emptyset$ and π "cuts" H: $H \subseteq T_\pi$ where $T_\pi = \{\Delta \in T, \pi \cap \Delta \neq \emptyset\}$. Last H is "surrounded" by O in π: we have $\Delta \in O$ if $\Delta \in T_\pi \setminus H$ is adjacent to another tetrahedron $\Delta' \in H$.

The detection of H is done as follows. For every edge $\mathbf{ab} \in L_\alpha$, we try several planes π that are orthogonal to **ab** or horizontal. (These directions are expected for cross-sections of H by π.) Then we search H by growing in $T_\pi \cap (F \setminus O)$ and starting by $H = T_{\mathbf{ab}} \cap (F \setminus O)$. If the surrounding condition is not

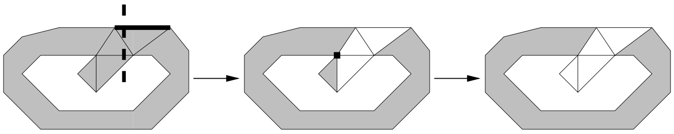

Figure 11: Our critical handle removal in the 2D case. O is white and $F \backslash O$ is the set of triangles filled in gray. Left: a critical edge e (bold edges) is detected in $c(\partial O)$, π is the dashed line, H is the set of three gray triangles cut by π. Middle: a singular vertex (black dot) appears when applying $O \leftarrow O \cup H$. Right: the singular vertex is removed by repairing (growing) O.

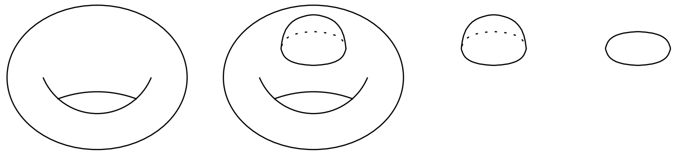

Figure 12: Notations for a shelling blocking. From left to right: O, O', $O' \setminus O$ is homeomorphic to $\{\mathbf{x} \in \mathbb{R}^3, ||\mathbf{x}|| \leq 1\}$, $c(O) \cap c(O' \setminus O)$ is a disc.

meet at the growing end, the detection of H fails and another pair $(\mathbf{ab}, \pi)$ is tried.

Once H is computed, we use F.R. as in C.E.R above: first apply $O \leftarrow O \cup H$, then try to remove the singular vertices in ∂O thanks to Algorithm 3 of [22] using input $G = H$. Fig. 11 shows an example of C.H.R.

At this point, we can remove remaining critical handles using C.H.R. of [22] (summary in Sec. 5.1.2), but we would like to moderate its g increase (due to small handles) and its use of Steiner vertices. Let M be the set of tetrahedra obtained as the union of the remaining handles detected as in the beginning of Sec. 5.3.3. We simply use the C.H.R. summarized in Sec. 5.1.2 by replacing its input set L_α by $L_\alpha \cap c(M)$.

Last we note that C.H.R. is preceded by S.G. in [25]. The summary in Sec. 2.2.2 reminds us that S.G. is useful to escape from local extremum of function f. Since our experiments show that g is almost constant by using S.G., we also use S.G. before C.H.R. (and after C.E.R) in method 2. Details on S.G. are given in Appendix D.

6. Escape from local extrema due to Shelling blocking

First Sec. 6.1 asserts that Shelling alone cannot generate as many surfaces as expected, then Sec. 6.2 provides examples. Last Sec. 6.3 explains how to solve (partly) this problem.

6.1. Definition of Shelling blocking

Let O and O' be tetrahedron sets such that $\emptyset \neq O \subset O' \subseteq F$, ∂O is manifold, $|\partial O|$ can be continuously deformed to $|\partial O'|$ by moving in $|O' \setminus O|$ using an isotopy. (It is a continuous function $h : |\partial O| \times [0, 1] \to |O' \setminus O|$ such that $h(|\partial O|, 0) = |\partial O|$, $h(|\partial O|, 1) = |\partial O'|$, and $\mathbf{x} \mapsto h(\mathbf{x}, t)$ is homeomorphism $\forall t$.) Thus $\partial O'$ is manifold with the same genus as ∂O. Let n be the number of the tetrahedra in $O' \setminus O$. We could expect to obtain O' from O by a greedy Shelling algorithm: let $O_0 = O$, then choose tetrahedron series $\Delta_i \in O' \setminus O_{i-1}$ for i varying from 1 to n and set $O_i = \{\Delta_i\} \cup O_{i-1}$ (Sec. 2.2.2). However, this is not always possible. In this case, we say that we have a *Shelling blocking*.

A Shelling blocking implies that there is $i \leq n$ such that no tetrahedron $\Delta_i \in O' \setminus O_{i-1}$ meets all required constraints to define O_i. In other words, $\partial(O_{i-1} \cup \{\Delta_i\})$ has a singular vertex or Δ_i has no triangle face in ∂O_{i-1} for every tried $\Delta_i \in O' \setminus O_{i-1}$. This may look surprising since Shelling in the 2D case does not have such a blocking according to [6]. (Replace tetrahedra by triangles, 2-manifold by 1-manifold, as in the top line of Fig. 2.)

6.2. Examples of Shelling blocking

In a first example, $\partial O'$ is a sphere (i.e. $\partial O'$ is homeomorphic to $\{\mathbf{x} \in \mathbb{R}^3, ||\mathbf{x}|| = 1\}$.) and we set $O = \{\Delta\}$ with a tetrahedron $\Delta \in O'$. According to [47, 20], there are three cases. Case 1: Shelling can success and can fail (This depends on its tetrahedron choices.), e.g. if O' is convex and large enough. Case 2: Shelling always fails, whatever its tetrahedron choices. (An example O' has only 12 vertices and 25 tetrahedra.) Case 3: there is never Shelling blocking, e.g. if the number of vertices in $c(O')$ is less than 9.

In a second example, ∂O is a manifold, $\partial(O' \setminus O)$ is a sphere, and $c(O) \cap c(O' \setminus O)$ is a disc. (i.e. it is homeomorphic to $\{\mathbf{x} \in \mathbb{R}^2, ||\mathbf{x}|| \leq 1\}$.) Fig. 12 shows an example where ∂O is a torus. Appendix E explains how to obtain a Shelling blocking in this context if $O' \setminus O$ has only 5 tetrahedra.

6.3. Escape from Shelling blocking using other operations

T.E., C.H.R., F.R., C.E.R. and S.G. in Secs. 2.2.2 and 5.3.2 can escape from Shelling blocking, although they are not originally designed to do that. This means that they can reach O' from O if we have a Shelling blocking. (Note that S.G. is not designed to meet $O \subset O'$.) Appendix E.1 shows an example: C.E.R. can escape from the last Shelling blocking in Sec. 6.2.

Here we present a method called *Unlock* to escape from remaining blockings. F.R. is used with two differences compared to [22]. First Repair is replaced by Repair2 (in Sec. 4.3.2) which is better. Second Repair2 is used without involving visually critical edges to select locations where it is applied. We apply Repair2 to a lot of small H. (Reminder: first force $O \leftarrow O \cup H$ then repair O in the neighborhood of H.) We try $H = \{\Delta\}$ for every tetrahedron $\Delta \in F \setminus O$ and $H = T_{\mathbf{v}} \cap (F \setminus O)$ for every vertex $\mathbf{v} \in c(F \setminus O)$. However, this generates high topological noise. A solution is to cancel the successful F.R. tentatives as in Sec. 5.2, but this is time consuming.

We prefer to enforce a supplementary constraint on the tried H before F.R.: the graph whose vertices are in $c(H) \cap c(O)$ and whose edges are in $c(\partial O)$ must be non-empty and connected. (The complete edge set is stored in an adjacency list before all F.R. for efficiency.) Experiment shows that this heuristic constraint greatly reduces topological noise and we explain it in an example. Assume that $H = \{\Delta\}$ and $F \setminus O$ is like a coin: a cylinder with large diameter and small thickness (Fig. 13). The constraint avoids almost all cases where Δ has a vertex in one side and another vertex in the other side of the coin. If F.R. is applied and is successful in such a case (i.e. without the constraint), it creates a hole connecting both sides and the genus of ∂O increases.

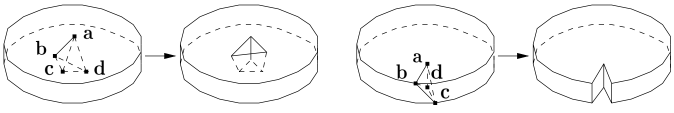

Figure 13: Without (left) and with (right) the supplementary constraint in Unlock. We have $H = \{\mathbf{abcd}\}$, $\mathbf{abcd} \in F \setminus O$, $F \setminus O$ forms a coin, O is all around the coin, **ab** is in the top side and **cd** is in the bottom side. Left: there are no edges in $c(\partial O)$ connecting **ab** and **cd**, then Repair2 creates a torus and g increases. Right: **bc** is in the edge of the coin, then Repair2 does not change g.

7. Experiments

7.1. Sparse input point cloud from images

We reconstruct complete environments thanks to videos taken by helmed-held omnidirectional multi-cameras and spherical cameras. First the multi-camera is self-calibrated ([29]), and synchronized if needed, using global shutter approximation. This involves a generic Structure-from-Motion ([28]) based on local bundle adjustment, which provides keyframes and matched Harris points that are used latter. We also reduce the accumulated drift of the trajectory thanks to detection and closure of loops inspired by [45, 44]: detection using a vocabulary tree of the matched points, closure using global bundle adjustment initialized by pose graph optimizations.

Then the sparse cloud of points is completed for surface reconstruction. For every keyframe, we detect Canny curves and Harris points, match them to those of the previous keyframe ([21] and the epipolar constraint. A Canny point is ignored if the angle between its curve tangent and the epipolar line at this point is less than a threshold ($\pi/4$). Both Canny and Harris points are reconstructed by ray intersection (using RANSAC and Levenberg-Marquardt methods). We only retain one reconstructed Canny pixel over four consecutive ones in a curve.

Last a pre-filtering is applied to remove gross outliers in the point cloud. Point $\mathbf{p}_i$ is reconstructed in keyframes located at $\mathbf{c}_j$ where $j \in V_i$ (Sec. 2.1). The point is rejected if V_i has no more than two elements or all apical angles are small: $\widehat{\mathbf{c}_{j_1}\mathbf{p}_i\mathbf{c}_{j_2}} < \epsilon$ where $\epsilon > 0$ is a threshold and $j_k \in V_i$. This roughly means that we reject a 3D point if the ratio of its uncertainty by its depth is greater than a threshold ([18]). The choice of ϵ is a trade-off: small values for selecting the scene background and large values for reducing noise in the foreground. We also reject points that are too far in the sky or below the ground as follows. Let $\mathbf{v}$ be an estimate of the vertical direction. (It is computed assuming that the multi-camera trajectory is roughly horizontal.) We reject $\mathbf{p}_i$ if there is $j \in V_i$ such that altitude $\mathbf{v}^\top \mathbf{p}_i$ is one of the ten smallest or ten largest values in $\{\mathbf{v}^\top \mathbf{p}_k, j \in V_k\}$.

7.2. Dataset, parameter setting, and initialization

We take four 1280×960 videos at 100Hz by walking during 16 minutes in an university campus using a DIY multi-camera. The multi-camera has four GoPro Hero3 cameras enclosed in a cardboard and mounted on a helmet (top of Fig. 14). The trajectory is about 1500m. long, 3459 keyframes are selected, reconstruction and pre-filtering provide 2.8M points and 13.6M rays. The scene includes streets, cars, facades and vegetation.

We use $\epsilon = \pi/18$ (i.e. 10° for point selection in Sec. 7.1), $\alpha = \pi/16$ (selection of critical edges in Sec. 5.1.2), $w_0 = \pi/2$ (P.R. parameter in Sec. 5.1.3). The bottom of Fig. 14 shows the points and the keyframe locations estimated by structure-from-motion (Sec. 7.1) and the surface using the method of [22] (Sec. 5.1). The surface has 4.2M triangles before sky removal.

All experimented methods have the same T, F and initial O by Sh. There are 17.6M tetrahedra in T; 53% of them are in F. 30% of the vertices in $c(\partial F)$ are singular in ∂F. Let O/F be the ratio of the number of tetrahedra in F that are in O. Let c and g be the number of connected components and genus of ∂O. Sh. provides $O/F = 83.29\%$, $c = 1$ and $g = 0$ in 13s. (We use a I7-5500U 1600MHz DDR3L laptop.)

7.3. Comparison of manifold tests

Here we compare the computation times of the surface reconstruction of [22] (summary in Sec. 5.1) by using different manifold tests: the new test based on directed-edges (introduced in Sec. 3) and the old test based on connected components (introduced by [22]). Both tests check that a vertex is regular in ∂O; their algorithms are detailed in Appendix A.

They obviously provide the same surface. There are three successive operations after Sh.: T.E., C.H.R. and P.R. Using the new test, their times are 3.8s, 27s and 15.6s. Using the old test, their times are 3.9s, 34s and 16.5s. The main acceleration is for the most time expensive operation C.H.R. These tests are done 30M times. Sh. does not change since it has its own test. We also check that the number of edges in D is small (this is useful for the complexity of the new test according to Sec. 3.3): the mean is 8.2, the standard deviation and maximum are 5.9 and 189. The new test is used in the next sections.

7.4. Comparison of repairs for singularity removals

Now we compare the method of [22] by trying different repairs used by C.H.R.: the new Repair2 based on singularity analysis (introduced in Sec. 4) and the old Repair that adds one tetrahedron at once (introduced by [22]). We remind that C.H.R. tries to add a lot of sets $H \subseteq F \setminus O$ to O by using a repair to remove the resulting singular vertices.

Before the use of repair, $O/F = 83.54\%$. Repair2 provides $O/F = 85.31\%$ in 10s. Repair provides $O/F = 85.21\%$ in 27s. Repair2 has the best ratio and is quite faster than Repair. We note that O/F does not increases a lot, but this is not a reason to omit C.H.R. since it removes visually critical handle. (Fig. 15 shows examples.) Repair2 is used in the next sections.

7.5. Comparisons of topologies

Here we experiment three methods using Repair2 and the new manifold test: M0 ([22], see also Sec. 5.1), M1 (Sec. 5.2) and M2 (Sec. 5.3). We remind that M1 is M0 with a C.H.R. that can be canceled, M2 is M0 such that T.E. is replaced by C.E.R. and C.H.R. is replaced by S.G. and more selective C.H.R.

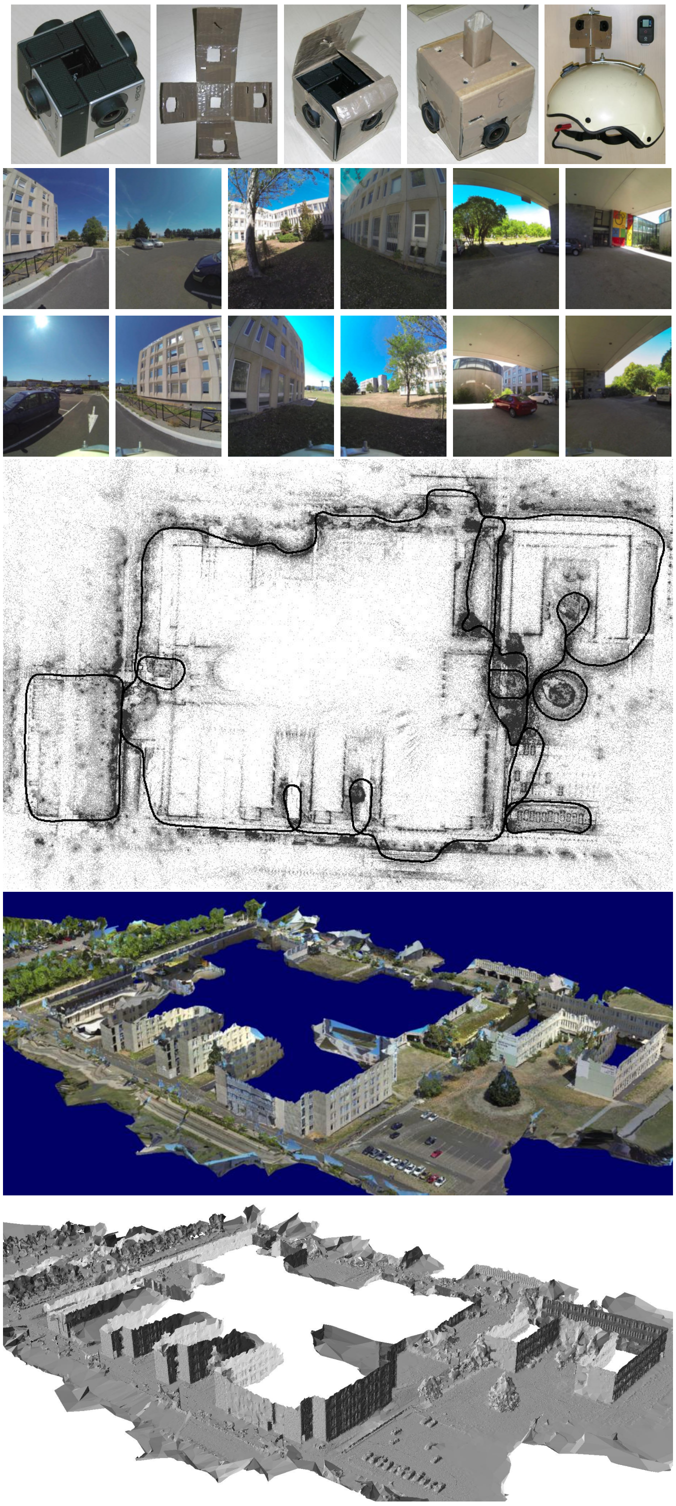

Figure 14: Top: four GoPro Hero 3 cameras enclosed in a cardboard box and resulting images of our multi-camera in the campus. Middle: top view of the structure-from-motion result including loop closure. Bottom: surface by [22] (best viewed in colors).

Figure 15: Examples of removals of spurious handles by [22] without (left) and with (middle and right) C.H.R. involving Repair2.

7.5.1. *Quantitative comparisons*

First we check assertions in Sec. 5 about g using Tab. 1. For M0, g is multiplied by about 4.6 due to C.H.R. For M1, g decreases by 35% thanks to C.H.R. For M2, C.E.R. provides g that is about 20% greater than that of T.E. The genus g is the same by S.G. and almost the same by C.H.R. Furthermore, P.R. has similar effects in all cases: g decreases by 10%-16%. At the end, both M1 and M2 provide quite smaller g than M0. (g is divided by 7.6 and 3.7.)

Second we see that there is a trade-off between O/F (which grows if the maximized visibility score function $f(O)$ grows) and small g (i.e. simplified topology): the larger the ratio O/F, the larger the genus g. Indeed, $M0 > M2 > M1$ for both O/F and g. The trade-off still holds if we use an alternative definition of T.E. by [24], but there are differences: this T.E. provides a quite larger g than the original T.E., which implies that $M0 > M1 > M2$ for both O/F and g (more details in Appendix F).

Last we check the assumption of Theorem 5 used by M1. It is not 100% meet: the number of connected components of O (which is not c) is 41 after C.H.R. of M1. However, it is 100% meet if we slightly modify C.H.R. (Add condition "$\mathbf{v}_i \in c(\partial O)$" in line 1 of Algorithm 2 of [22].); then we obtain a very similar M1's surface (only 281 different triangles) with same c and g.

7.5.2. *Removing holes due to bad or lacking points*

First we remind what is a "hole" in our context. Bad input points are remaining after the pre-filtering in Sec. 7.1. Their rays can intersect tetrahedra that become free-space and outside although they should be matter and inside. These tetrahedra form holes. More precisely, there are two kinds of holes:

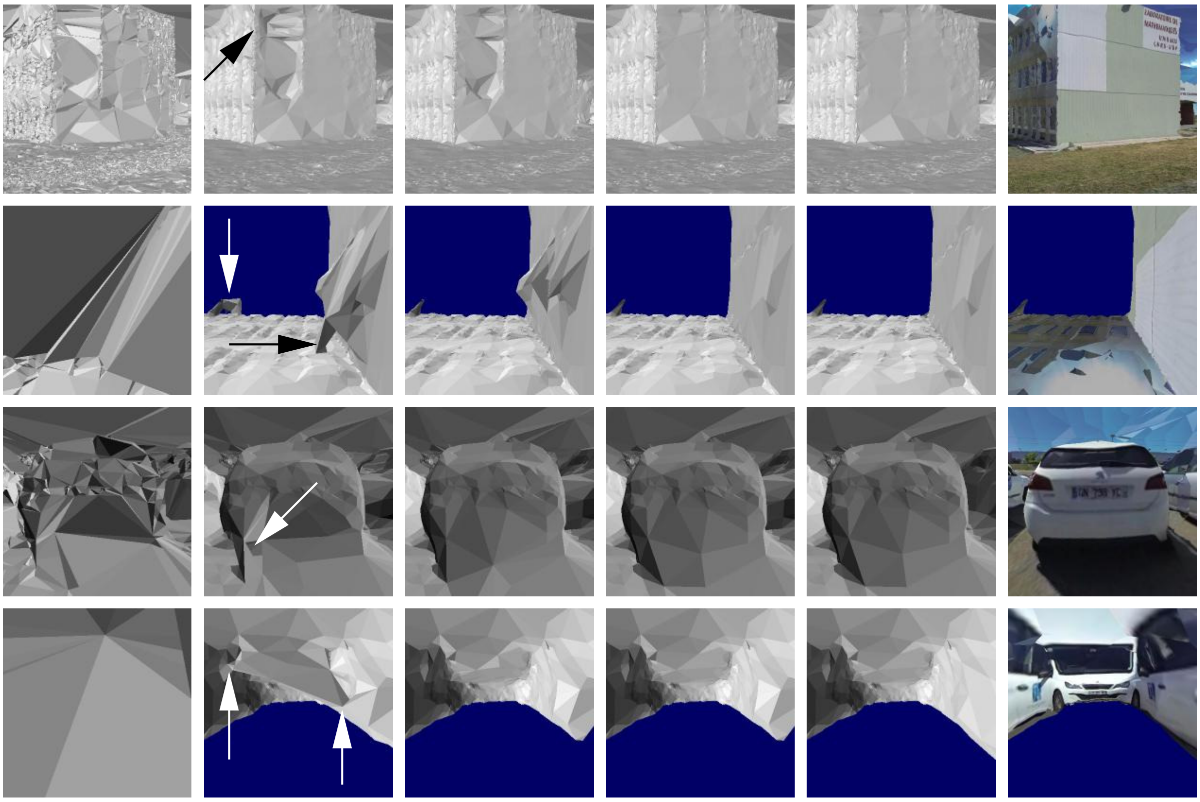

Figure 16: Handles of M0 and their removals by M1, M2 and M3. From left to right: ∂F, M0, M1, M2, M3, and texture of M3. There are views from outside and from inside of a building corner (rows 1&2) and a car (rows 3&4). The arrows are pointing to handles removed by M1/M2/M3.

Table 1: Labeling, surface topology (number of connected components and genus) and times of methods M0-2. Reminders: both M0 and M1 successively apply Sh.-T.E.-C.H.R.-P.R., M2 successively applies Sh.-C.E.R.-S.G.-C.H.R.-P.R.

step	method	O/F	c	g	time
T.E.	M0	83.54%	368	377	3.9s
C.H.R.	M0	85.31%	483	1759	10.4s
P.R.	M0	79.87%	56	1549	15.6s
T.E.	M1	83.54%	368	377	3.9s
C.H.R.	M1	84.75%	499	243	16.5s
P.R.	M1	79.14%	56	204	16s
C.E.R.	M2	84.90%	154	454	4.9s
S.G.	M2	84.99%	155	454	5.1s
C.H.R.	M2	85.02%	171	459	5.6s
P.R.	M2	79.45%	55	411	15.6s

concavities and handles. We explain them for a simple example: a wall in a city such that both planar sides of the wall are seen by the camera trajectory (the $\mathbf{c}_j$). In the first case, a concavity deforms one side of the wall without topology change, e.g. if the wall includes a bad point. Such holes can be removed by P.R. if they are peaks (more details in Appendix C of [22]). In the second case, a handle connects both wall sides, e.g. if a ray of a bad point crosses the wall.

Second we show examples of holes surrounded by handles (second case) that are removed by our lowered genus constraint. In both Figs. 16 and 17, holes surrounded by handles of M0 (column 2) are removed by M1 (column 3) and/or M2 (column 4). M2 usually has the best visual result since M1 sometimes fills large holes that should not be (where there is a lot of free-space F), i.e. its surface connects important scene components that should not be. For example in row 2 of Fig. 17, the space between the tree and the notice board and the ground is in F but the M1 surface immediately connects these three scene components. In row 2 of Fig. 16, we also see that large blunders in F are greatly reduced by the manifold constraint (See M0 result.) and further reduced by the low genus constraint. (See M1 and M2 results.)

7.6. Details on C.E.R, S.G. and C.H.R. used in method M2

We note that O/F (Tab. 1) does not increase a lot by operations C.E.R., S.G. and C.H.R. However, this is not a reason to omit them in M2. Fig. 18 shows examples of visual artifacts that occur in the final surface if we omit one of them. If C.E.R. is omitted, a porch forms a blind alley and there is a visual artifact adjacent to a pillar of the porch (row 1). If S.G. is omitted, two adjacent cars are connected by the surface (row 3). If C.H.R. is omitted, spurious handles occur between another post and the ground and a building (row 5). Fig. 18 also shows a top view of all improvements (tetrahedra moved from $F \setminus O$ to O) done by each of these three operations, which are non negligible.

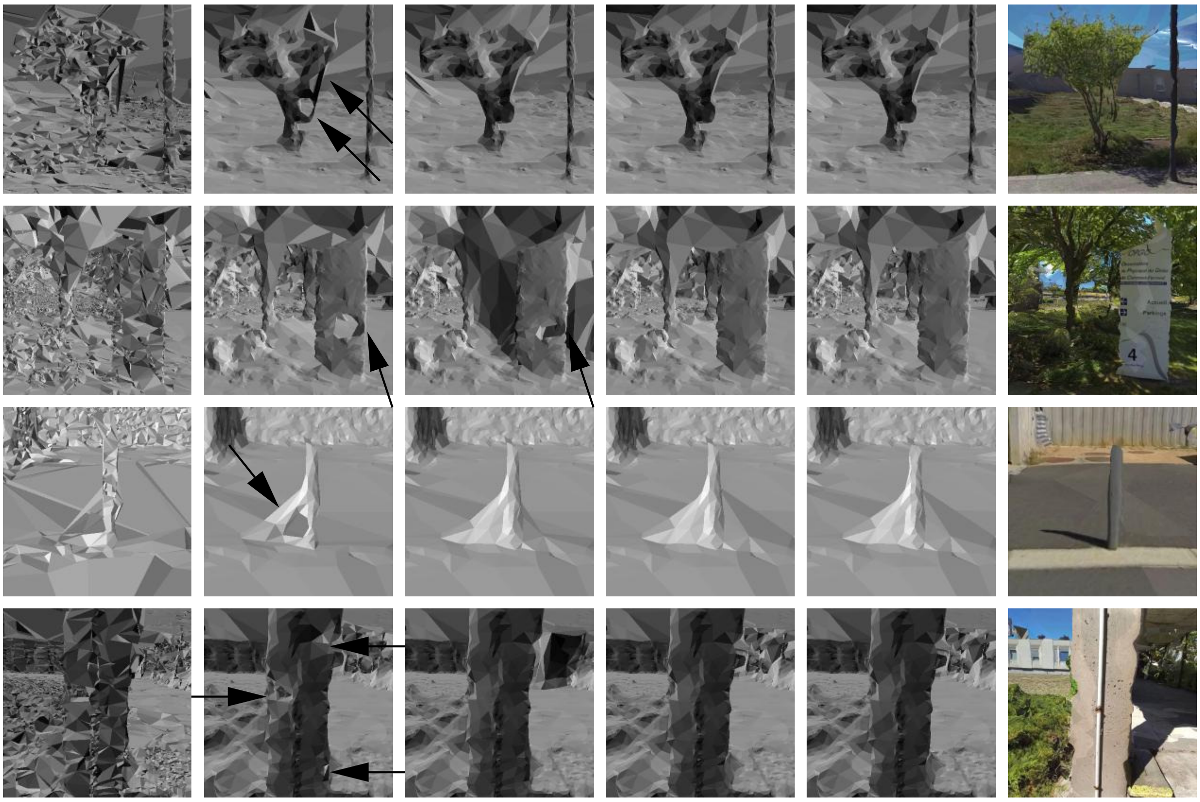

Figure 17: Handles of M0 and their removals by M1, M2 and M3. From left to right: ∂F, M0, M1, M2, M3, and texture of M3. There are views of a small tree (row 1), a notice board under a tree (row 2), an anti-parking post (row 3) and a pillar (row 4). The arrows are pointing to handles removed by M1/M2/M3.

7.7. Unlock shelling blocking

Now we study a method M3, which is M2 with an additional step Unlock (Sec. 6.3) done after C.H.R. and before P.R. Fig. 19 shows examples of visual artifacts of M2 that are removed by M3. It also shows a top view of all improvements (tetrahedra moved from $F \setminus O$ to O) done by Unlock, which are non negligible. The ratio O/F increases from 85.02% to 85.39% thanks to Unlock in 15.3s; the g of M3 is 517. (25% greater than the g of M2.) We also check two assertions in Sec. 6.3. If Unlock does not have the supplementary constraint, there is a lot of topological noise: g of M3 is multiplied by 14 ($g = 7279$). If we replace the supplementary constraint (before every use of Repair2) by testing $\delta c - \delta g \geq 0$ as in Sec. 5.2.2 (after every successful use of Repair2), g of M3 decreases by 21% ($g = 408$), the computation time of Unlock is multiplied by 2.1, and $O/F = 85.38\%$.

The video **Video.mp4** in the supplementary material shows walkthroughs in the surface generated by M3.

7.8. Comparison of accuracies

We compare the accuracies of M0-3's surfaces with respect to a ground truth surface: a synthetic urban scene with textures of real images taken in a city. First a multi-camera video (4 GoPro cameras) is generated by ray-tracing of the scene. The camera trajectory is a 621m long closed loop around buildings. All methods have the same input generated as in Sec. 7.1: 601 keyframes and 687k points. Fig. 20 shows the images at a single location, estimated and ground truth surfaces.

Table 2: Mean, standard deviation and quartiles of geometric error $e(\mathbf{p})$ in cm, c and g for the methods M0-3.

M.	$\bar{e}$	σ_e	70%	80%	90%	c	g
M0	34.73	103	14	26	76	3	82
M1	34.12	101	14	26	78	3	2
M2	32.89	100.8	14	24	70	3	17
M3	32.93	100.8	14	24	70	3	33

A geometric error e between the estimated and ground truth surfaces is computed as follows. Let $\mathbf{c}_i$ and $\mathbf{c}_i^g$ be the estimated and ground truth locations of the multi-camera at the i-th keyframe. First we set the estimated surface in the coordinate system of the ground truth thanks to the similarity transformation Z that minimizes

$$E(Z) = \sum_{i=0}^{600} \|Z(\mathbf{c}_i) - \mathbf{c}_i^g\|^2. \tag{19}$$

We obtain $\sqrt{E(Z)/601} = 9.1cm$. Then $e(\mathbf{p})$ is defined by the distance between the point $\mathbf{p}$ and the ground truth surface; $\mathbf{p}$ is randomly and uniformly sampled in the estimated surface.

Tab. 2 shows c, g and quartiles of $e(\mathbf{p})$. The quartiles below 70% are the sames for all methods. We see that M0-3 have similar accuracies (M2 and M3 are slightly better than M1, which

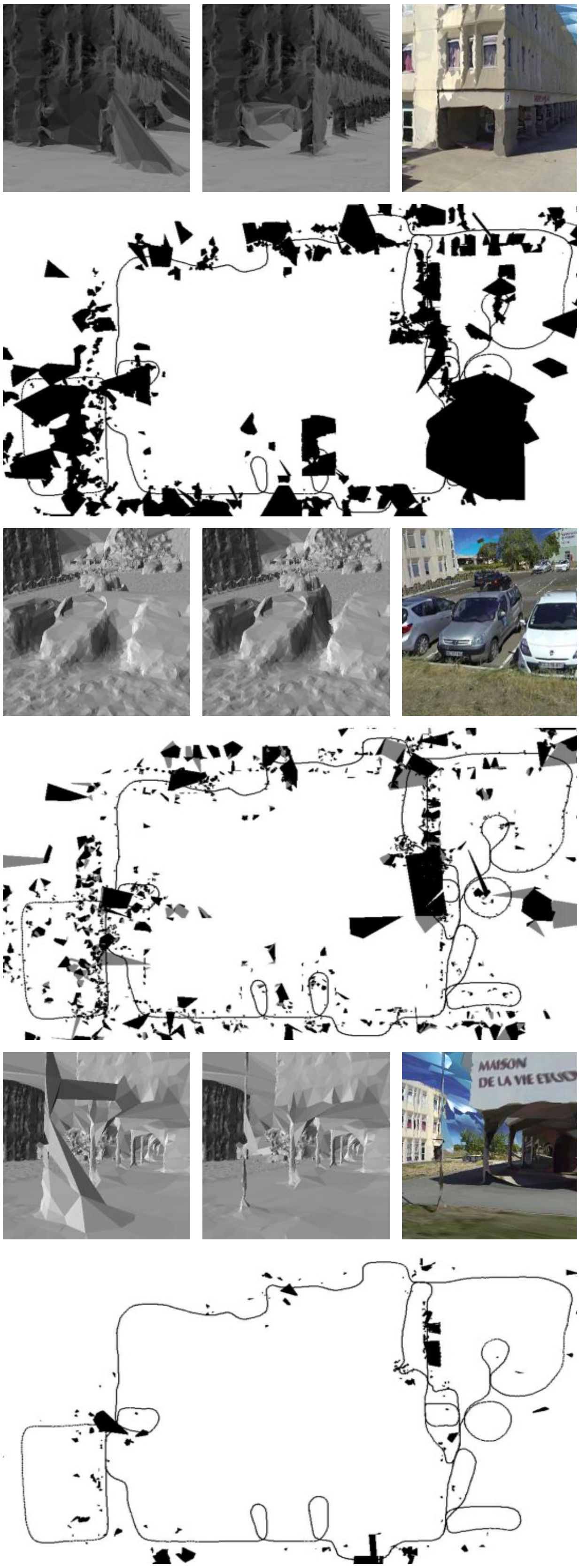

Figure 18: Improvements done by C.E.R (top), S.G. (middle) and C.H.R. (bottom) involved in method M2. In each case, there is a top view of tetrahedra switched from $F \setminus O$ to O (in black) or from O to $F \setminus O$ (in grey, S.G. only), superimposed by the camera trajectory. There are also local views of the surface obtained without and with the operation.

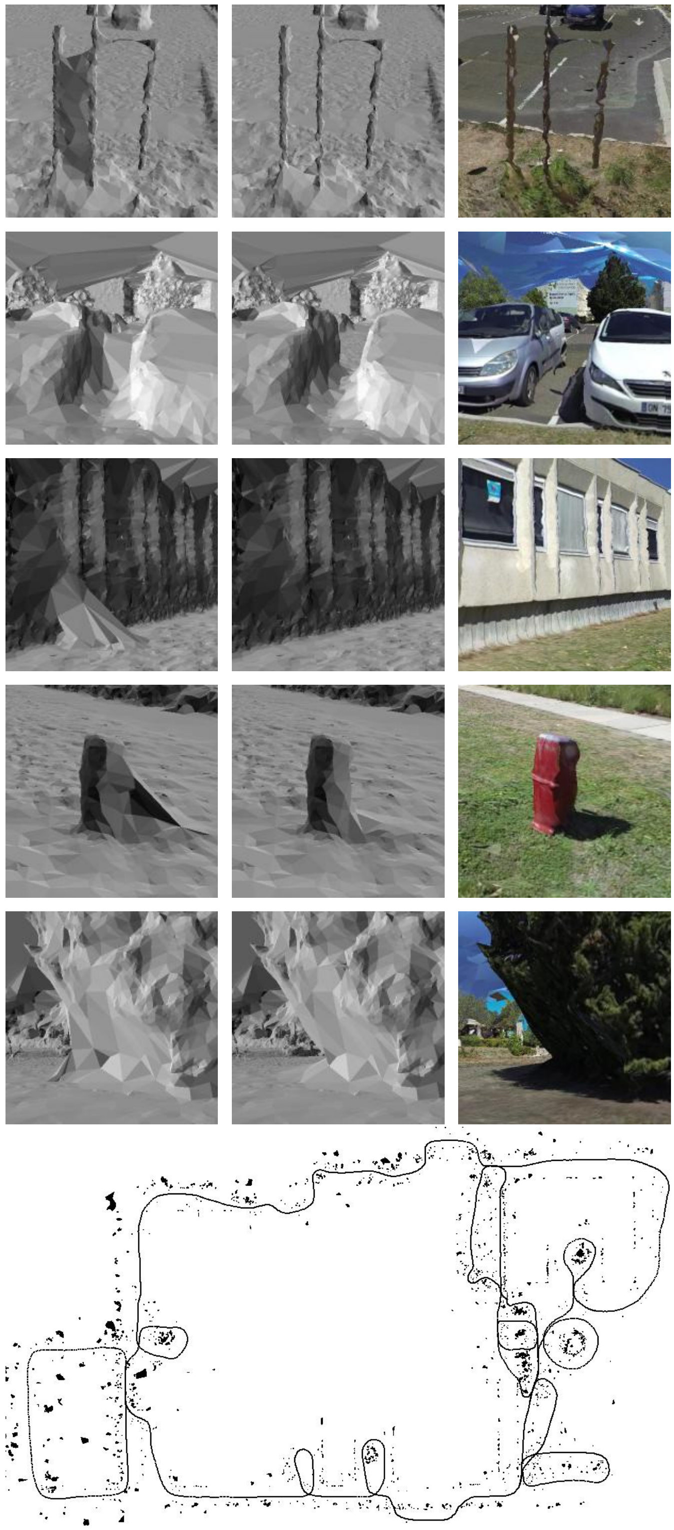

Figure 19: Improvements done by Unlock involved in method M3. Top and middle: local views of the surface without (left) and with (middle and right) Unlock showing three wood posts, two cars, facade and ground, fire hydrant, three and ground. Bottom: a top view of tetrahedra switched from $F \setminus O$ to O (in black) by Unlock, superimposed by the camera trajectory.

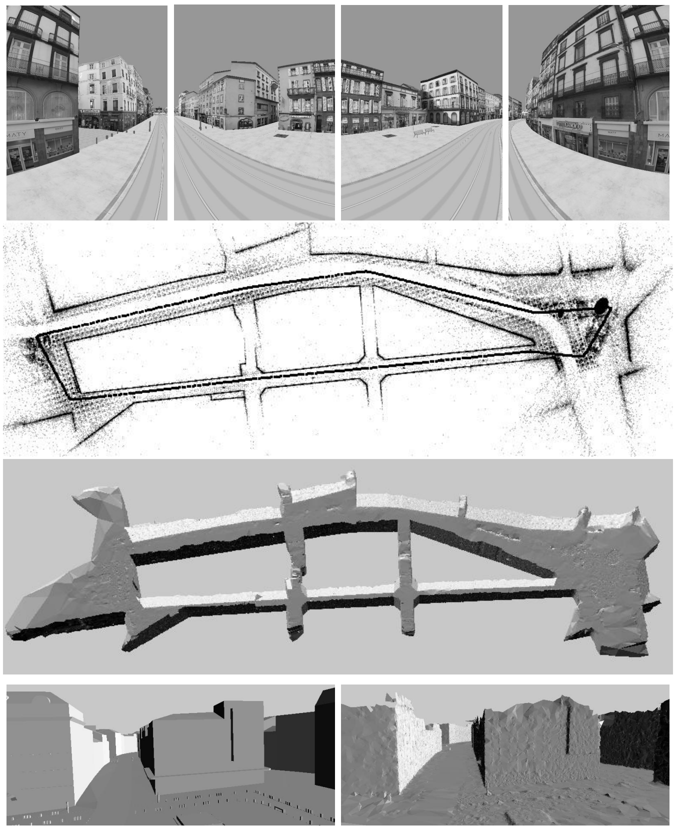

Figure 20: Experiments using a synthetic urban scene. Top: images at a location. Middle: top views of SfM result and M3 surface. Bottom: local views of ground truth and M3 surfaces.

Table 3: Genuses obtained by the methods M0-3 and values of α.

α	0.1	0.15	0.2	0.25	0.3
M0	3774	2269	1531	1156	945
M1	194	199	207	220	225
M2	920	580	406	308	237
M3	1019	669	511	417	353

in turn is slightly better than M0.) using a similar number of triangles (1.09M triangles) and times between 36s and 40s. The main difference between methods is the genus g. Its ground truth is 3 since the streets form 3 independent loops and the buildings do not have handles. We see that the topological noise of M0 is quite reduced by M1, M2 and M3. M1 provides the best g but its surface approximates the two streets in the scene center by (completely) blind alleys although they are not. These streets are incompletely blind alleys but have spurious handles in M0, M2 and M3 cases.

7.9. Varying α

The angle α is an important parameter since the set of the critical edges depends on α (L_α in Eq. 16) and operations C.E.R, C.H.R., S.G. take L_α as input. (Sh., T.E. and P.R. do not.)

Figure 21: Varying α for the method M3: $\alpha = 0.3$ on the left and $\alpha = 0.1$ on the middle and right. There are views of a fence in front of a building, a light post between two streets, and tall grass under trees.

First we detail a link between g and α. By varying α around its default value $\pi/16 \approx 0.2$, Tab. 3 shows that g increases if α decreases for M0, M2 and M3. Indeed, new handles can be generated at the neighborhood of every critical edge in L_α where operations C.E.R./C.H.R. are applied and the number of edges in L_α increases if α decreases. In contrast to this, g slightly decreases if α decreases for M1. There is a reason: the cancellations of F.R. involved in C.H.R. of M1 prohibit g increments. (This is the only differences between M0 and M1.) In all cases, if α decreases from 0.3 to 0.1., c increases in range $[47, 70]$ and the computation time of C.H.R. is multiplied by about 3.5.

Second Fig. 21 shows surfaces differences for M3 between $\alpha = 0.3$ and $\alpha = 0.1$. We see that a large value of α simplifies the topology, e.g. holes/handles are removed in a fence (top). However, this also degrades the O growing at some locations, e.g. near a light post (middle) and below trees (bottom). As in Sec. 7.5.1, there is a trade-off between visibility consistency and topology simplicity.

7.10. Comparison with a graph-cut method

First we discuss a graph-cut method (named GC) of [40]. GC takes ours 3D Delaunay triangulation T as input, then computes a closed surface $S \subset c(T)$ that minimizes a cost function $E_{vis} + \lambda E_{qual}$ composed of a visibility term E_{vis} and[2] a surface quality term E_{qual}, last GC has a Laplacian smoothing. On the one hand, we observe that thin structures (e.g. posts) and foliages are removed from S if the weighting λ increases. On

[2]Addendum: using $\alpha_{vis} = 1$.

the other hand, S is less robust to "bad" points if λ decreases. These "bad" points can also includes a few good ones which are not enough numerous to generate a descent shape, e.g. a few good points inside a building or a car if the multi-camera is outside. We choose $\lambda = 2$ as a trade-off. We also note that a good surface quality in the sense of [40] implies a low number of singular vertices. Indeed, the percentage of singular vertices of S decreases if λ increases: 1.5% if $\lambda = 1$, 0.26% if $\lambda = 2$, 0.07% if $\lambda = 3$. The computation time of the graph-cut optimization also increases (from 25s to 64s) and the number of triangles in S decreases (from 5M to 3.5M).

Second we compare GC and our method M3 using the default value $\alpha = \pi/16$. The number of triangles is 4.2M for M3 and 4.6M for GC. Rows 1-2 in Fig. 22 show that the GC surface is noisier than that of M3 in textured walls. This is coherent with an accuracy evaluation using the dataset in Sec. 7.8: the mean error of M3 is lower than that of GC. (They are 0.33m and 0.43m respectively.) Furthermore, the most complete foliages and trunks of trees are usually obtained by M3 (examples in rows 3-4). However we observe in rows 5-6 that GC is usually the most robust with respect to "bad" points inside buildings. (In row 6, M3 generates a spurious handle connecting two windows at a building corner.) We also note that GC has the best result in row 7 by forcing to O several thin matter tetrahedra that connect roofs of two cars.

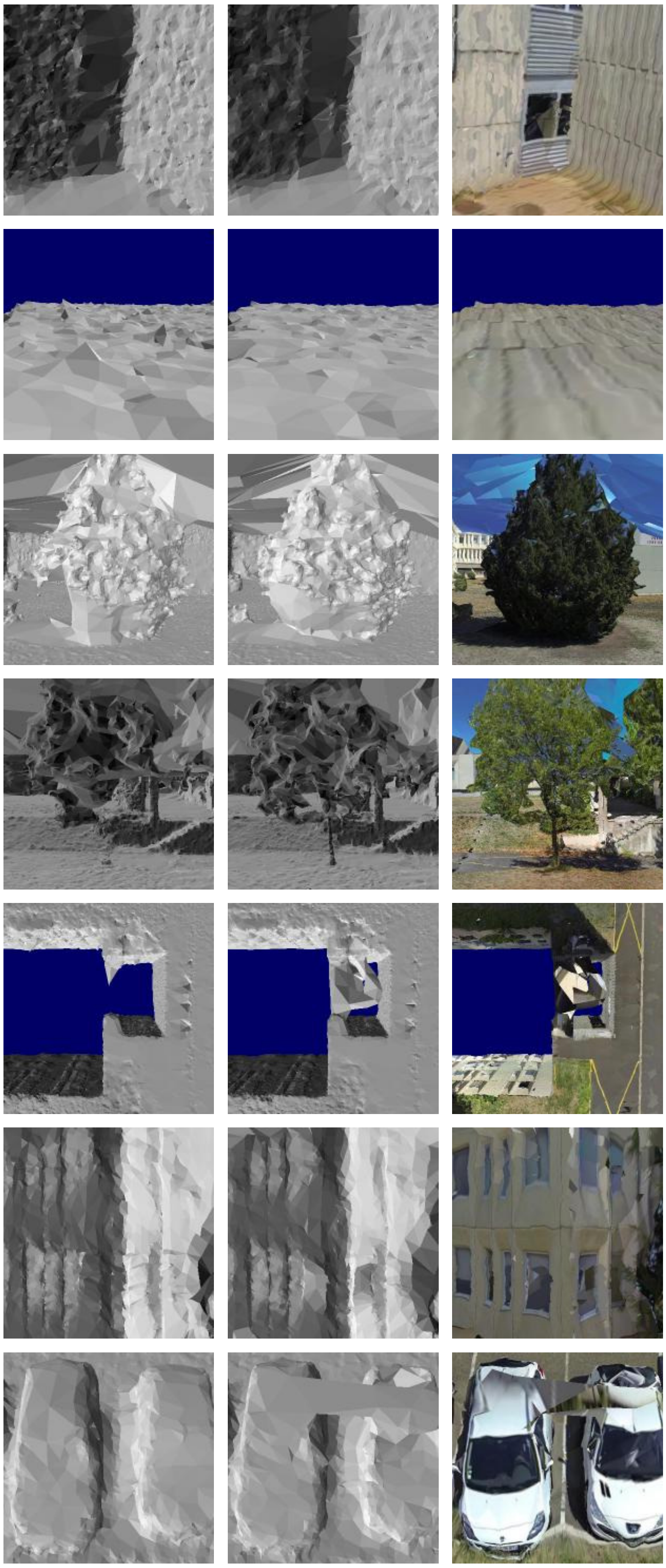

Figure 22: Differences between GC (left) and M3 (middle and right). M3 is the best in rows 1-4 and GC is the best in rows 5-7 (details in the text). All views are from the outside except those in rows 2 and 5.

8. Conclusion

Topology constraints (manifoldness, low genus, connectedness) are under-explored in Computer Vision methods that estimate a surface given points reconstructed from images, although these constraints are useful for both surface estimation and applications. This article presents the first surface reconstruction methods that simultaneously enforce visibility consistency and low genus. Starting from a previous work enforcing manifoldness, we quantitatively reduce the genus and show surface improvements including hole removals. A simple modification removes topological noise due to an operation of the original method. A second method is more involved but adjusts the topology simplification thanks to an user parameter during the visibility consistency optimization. Other contributions include an acceleration of the manifold test by using the orientability of the 3D Delaunay triangulation (of the input points), a more efficient removal of surface singularities which improves escapes from local extrema. This removal is based on a study of non-manifoldness near a vertex or an edge of the surface. The local extrema are partly due to limitations of an operation called "shelling" in Combinatorial Topology.

We experiment in a context that we believe useful for applications. First the input point cloud is sparse. This is useful in several contexts including large scale scenes, limited computational resources and initialization of dense stereo. Second the points are reconstructed from videos taken by several consumer cameras (or a spherical camera) mounted on a helmet and by walking (or biking) in complex environments.

Future work includes improvements to escape from local extrema of our optimization problem, e.g. by finding a (provably) set of local operations that reach all manifold surfaces embedded in the 3D Delaunay triangulation. Furthermore, we should improve contour reconstruction, use of connectedness, surface smoothing and visibility score function. Our contributions can also be applied to the incremental surface reconstruction. (We only experiment batch surface reconstruction.) Last a dense stereo refinement should add surface details and reduce reconstruction errors.

Acknowledgments

Thanks to CNRS, Université Clermont Auvergne and Institut Pascal for funding Maxime Lhuillier. Thanks to the CRISTAL project for providing the ground truth VRML model used in the quantitative evaluation.

Appendix A. Algorithms of manifold tests

Here we use shortened notations: t_i^Δ is the face triangle of a tetrahedron $\Delta = \mathbf{v}_0^\Delta\mathbf{v}_1^\Delta\mathbf{v}_2^\Delta\mathbf{v}_3^\Delta$ that does not include $\mathbf{v}_i^\Delta$, i.e. t_i^Δ is $\mathbf{v}_i^\Delta$-opposite in Δ. In our implementation, every tetrahedron $\Delta \in T^\infty$ (Sec. 4.1) stores four booleans $t_i^\Delta \in \partial O$ in addition to a boolean $\Delta \in O$. This redundancy accelerates all tests by reducing the use of indices or pointers to neighboring tetrahedra.

Appendix A.1. Our manifold test

We remind that $\pi \in A_4$ iff $(\pi 0, \pi 1, \pi 2, \pi 3) \in \tilde{A}_4$ where

$$\begin{aligned}\tilde{A}_4 = \{ \quad & (0,1,2,3),(0,2,3,1),(0,3,1,2),\\ & (1,2,0,3),(1,0,3,2),(1,3,2,0),\\ & (2,3,0,1),(2,0,1,3),(2,1,3,0),\\ & (3,0,2,1),(3,2,1,0),(3,1,0,2) \quad \}.\end{aligned} \tag{A.1}$$

Algorithm 3 presents (in C style) our directed edge-based manifold test (Sec. 3.3). In the first step (collect the directed edges), lines 3-4 select $\Delta \in T_\mathbf{v} \cap O$, then lines 5-20 describe every $\pi \in A_4$ as follows. Lines {5,9,13,17} select the four cases of $\pi 0 \in \{0,1,2,3\}$. Lines 6-8 are the three sub-cases of $\pi 0 = 0$ based on tests $t_{\pi 1}^\Delta \in \partial O$ where $\pi 1 \in \{1,2,3\}$, then $(\mathbf{v}_{\pi 2}^\Delta, \mathbf{v}_{\pi 3}^\Delta)$ is collected in every sub-case such that π is even. The other cases of $\pi 0$ are similar. In the second step (Check that D is a directed cycle.), we permute the edges in E to check that D is a closed path, then check that every vertex is traversed only once by this path.

Algorithm 3. *Directed edge-based manifold test for* $\mathbf{v}$

01: // collect the directed edges of D in a table
02: Let E be a table of vertex indices, and let n=0;
03: **for** each tetrahedron $\Delta \in T_\mathbf{v}$
04: **if** ($\Delta \in O$) {
05: **if** ($\mathbf{v}_0^\Delta$==$\mathbf{v}$) {
06: **if** ($t_1^\Delta \in \partial O$) { E[n++]= $\mathbf{v}_2^\Delta$; E[n++]= $\mathbf{v}_3^\Delta$; }
07: **if** ($t_2^\Delta \in \partial O$) { E[n++]= $\mathbf{v}_3^\Delta$; E[n++]= $\mathbf{v}_1^\Delta$; }
08: **if** ($t_3^\Delta \in \partial O$) { E[n++]= $\mathbf{v}_1^\Delta$; E[n++]= $\mathbf{v}_2^\Delta$; }
09: } **else if** ($\mathbf{v}_1^\Delta$==$\mathbf{v}$) {
10: **if** ($t_2^\Delta \in \partial O$) { E[n++]= $\mathbf{v}_0^\Delta$; E[n++]= $\mathbf{v}_3^\Delta$; }
11: **if** ($t_0^\Delta \in \partial O$) { E[n++]= $\mathbf{v}_3^\Delta$; E[n++]= $\mathbf{v}_2^\Delta$; }
12: **if** ($t_3^\Delta \in \partial O$) { E[n++]= $\mathbf{v}_2^\Delta$; E[n++]= $\mathbf{v}_0^\Delta$; }
13: } **else if** ($\mathbf{v}_2^\Delta$==$\mathbf{v}$) {
14: **if** ($t_3^\Delta \in \partial O$) { E[n++]= $\mathbf{v}_0^\Delta$; E[n++]= $\mathbf{v}_1^\Delta$; }
15: **if** ($t_0^\Delta \in \partial O$) { E[n++]= $\mathbf{v}_1^\Delta$; E[n++]= $\mathbf{v}_3^\Delta$; }
16: **if** ($t_1^\Delta \in \partial O$) { E[n++]= $\mathbf{v}_3^\Delta$; E[n++]= $\mathbf{v}_0^\Delta$; }
17: } **else if** ($\mathbf{v}_3^\Delta$==$\mathbf{v}$) {
18: **if** ($t_0^\Delta \in \partial O$) { E[n++]= $\mathbf{v}_2^\Delta$; E[n++]= $\mathbf{v}_1^\Delta$; }
19: **if** ($t_2^\Delta \in \partial O$) { E[n++]= $\mathbf{v}_1^\Delta$; E[n++]= $\mathbf{v}_0^\Delta$; }
20: **if** ($t_1^\Delta \in \partial O$) { E[n++]= $\mathbf{v}_0^\Delta$; E[n++]= $\mathbf{v}_2^\Delta$; }
21: }
22: }
23: **if** (n==0) **return** 1; // if $\mathbf{v} \notin c(\partial O)$
24: **if** (n< 6) **return** 0; // too short for a directed cycle
25: // does E form a closed path ?
26: **for** (i=0; i+3< n; i= i+2)
27: **if** (E[i+1]!= E[i+2]) {
28: **for** (j= i+4; j+1< n; j=j+2)
29: **if** (E[i+1]==E[j]) {
30: Swap(E[i+2], E[j]); Swap(E[i+3], E[j+1]);
31: **break**;
32: }
33: **if** (j+1≥ n) **return** 0;
34: }
35: **if** (E[n-1]!= E[0]) **return** 0;
36: // do we have distinct vertices in the path ?
37: **for** (i= 0; i+1< n; i= i+2)
38: **for** (j= i+2; j+1< n; j= j+2)
39: **if** (E[i]==E[j])
40: **return** 0;
41: **return** 1;

end

Furthermore, it can be shown that

- Theorem 1 is true if we replace T by T^∞ in its statement
- we can replace "$\Delta \in O$" by "$\Delta \in T^\infty \setminus O$" in Eqs. 8 and 10, then Theorem 2 is still true.

Thus the directed-edge based test in Algorithm 3 is the same if we replace $\Delta \in T_\mathbf{v}$ by $\Delta \in T_\mathbf{v}^\infty$ in line 03 (Store $T_\mathbf{w}^\infty$ in a table for every $\mathbf{w} \in c(T^\infty)$.) and replace $\Delta \in O$ by $\Delta \notin O$ in line 04.

Appendix A.2. A previous manifold test

Algorithm 4 is a previous test (the so-called Graph-based Vertex Test of [22]), that we experiment and compare to Algorithm 3. It checks that the vertex $\mathbf{v}$ is regular in ∂O by a traversal of the adjacency graph of $T_\mathbf{v}^\infty$ (proof in Theorem 3 of [19]). Every tetrahedron Δ has indices (or pointers) to its four adjacent tetrahedra Δ_i in T^∞. (The notation Δ_i meets $\Delta_i \cap \Delta = t_i^\Delta$.)

Algorithm 4. *Tetrahedron-based manifold test for* $\mathbf{v}$

// every tetrahedron Δ has boolean $f_\Delta = 0$
for each tetrahedron $\Delta \in T_\mathbf{v}^\infty$

```
f_Δ= 1;
c= 0;
let P be a table of tetrahedra indices, and let n= 0;
for each tetrahedron Δ' ∈ T_v^∞
  if (f_Δ') {
    f_Δ'= 0;
    P[n++]= Δ';
    while (n) {
      Δ= P[--n];
      if (v_0^Δ! = v && t_0^Δ ∉ ∂O && f_Δ0)
          { f_Δ0 = 0; P[n++]= Δ0; }
      if (v_1^Δ! = v && t_1^Δ ∉ ∂O && f_Δ1)
          { f_Δ1 = 0; P[n++]= Δ1; }
      if (v_2^Δ! = v && t_2^Δ ∉ ∂O && f_Δ2)
          { f_Δ2 = 0; P[n++]= Δ2; }
      if (v_3^Δ! = v && t_3^Δ ∉ ∂O && f_Δ3)
          { f_Δ3 = 0; P[n++]= Δ3; }
    }
    c++;
  }
return (c<3); // c<2 if v ∉ c(∂O)
```

end

Appendix B. Proof of Theorem 2

First we show Lemma 1.

Proof. First assume that edge $\mathbf{ab} \in U$ and show that triangle $\mathbf{abv} \in \partial O$. Thus $(\mathbf{a}, \mathbf{b}) \in D$ or $(\mathbf{b}, \mathbf{a}) \in D$. In both cases, Eq. 8 implies that $\mathbf{abv} \in \partial O$.

Second assume that $\mathbf{abv} \in \partial O$ and show that $\mathbf{ab} \in U$. There is a tetrahedron $\Delta \in O$ such that $\mathbf{abv}$ is a face of Δ. Let $\mathbf{x}$ be a vertex such that $\Delta = \mathbf{vxab}$. Since there are exactly two possible orientations, $\overline{(\mathbf{v}, \mathbf{x}, \mathbf{a}, \mathbf{b})} = o(\Delta)$ or $\overline{(\mathbf{v}, \mathbf{x}, \mathbf{b}, \mathbf{a})} = o(\Delta)$. Since $\overline{X} = o(\Delta) \iff X \in o(\Delta)$, Eq. 8 implies $(\mathbf{a}, \mathbf{b}) \in D$ in the first case and $(\mathbf{b}, \mathbf{a}) \in D$ in the second case. In both cases, $\mathbf{ab} \in U$. □

Then we show Lemma 2.

Proof. According to [3, 19], a vertex $\mathbf{v} \in c(\partial O)$ is regular in ∂O iff the $\mathbf{v}$-opposite edges in the triangles of ∂O (having $\mathbf{v}$ as vertex) form a cycle. Now we use Lemma 1 and obtain the result. □

We also need the two following lemma.

Lemma 6. *If* $(\mathbf{x}, \mathbf{y}) \in D$, $(\mathbf{y}, \mathbf{x}) \notin D$.

Proof. If $(\mathbf{x}, \mathbf{y}) \in D$, there are a tetrahedron $\Delta \in O$ and a vertex $\mathbf{v}_1$ such that $\overline{(\mathbf{v}, \mathbf{v}_1, \mathbf{x}, \mathbf{y})} = o(\Delta)$ and $\mathbf{vxy} \in \partial O$. Assume (reductio ad absurdum) that $(\mathbf{y}, \mathbf{x}) \in D$. There are a tetrahedron $\Delta' \in O$ and a vertex $\mathbf{v}'_1$ such that $\overline{(\mathbf{v}, \mathbf{v}'_1, \mathbf{y}, \mathbf{x})} = o(\Delta')$. Since $\mathbf{vxy} \in \partial O$, $\mathbf{vxy}$ is included in a single tetrahedron of O. Thus $\Delta = \Delta'$ and $\mathbf{v}_1 = \mathbf{v}'_1$. Now we have $\overline{(\mathbf{v}, \mathbf{v}_1, \mathbf{y}, \mathbf{x})} = \overline{(\mathbf{v}, \mathbf{v}_1, \mathbf{x}, \mathbf{y})}$, which is impossible. □

Lemma 7. *Let* $\mathbf{uva}_1\mathbf{a}_2 \cdots \mathbf{uva}_k\mathbf{a}_{k+1}$ *be a series of k distinct tetrahedra in T. If* $\overline{(\mathbf{v}, \mathbf{u}, \mathbf{a}_1, \mathbf{a}_2)} = o(\mathbf{uva}_1\mathbf{a}_2)$, $\overline{(\mathbf{v}, \mathbf{u}, \mathbf{a}_k, \mathbf{a}_{k+1})} = o(\mathbf{uva}_k\mathbf{a}_{k+1})$.

Proof. First we show that if tetrahedra $\Delta = \mathbf{v}_0\mathbf{v}_1\mathbf{v}_2\mathbf{v}_3$ and $\Delta' = \mathbf{v}_0\mathbf{v}_1\mathbf{v}_2\mathbf{v}'_3$ are consistently oriented,

$$\overline{(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)} = o(\Delta) \Rightarrow \overline{(\mathbf{v}_1, \mathbf{v}_0, \mathbf{v}_2, \mathbf{v}'_3)} = o(\Delta'). \tag{B.1}$$

We have $\overline{(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}'_3)} = o(\Delta')$ or $\overline{(\mathbf{v}_1, \mathbf{v}_0, \mathbf{v}_2, \mathbf{v}'_3)} = o(\Delta')$. Assume (reductio ad absurdum) that the former is true. Thanks to Theorem 1, Δ and Δ' are consistently oriented by using orientations $o(\Delta)$ and $o(\Delta')$: there are $\pi \in A_4$ and $\pi' \in A_4$ such that $(\mathbf{v}_{\pi 0}, \mathbf{v}_{\pi 1}, \mathbf{v}_{\pi 2}) = (\mathbf{v}_{\pi' 1}, \mathbf{v}_{\pi' 0}, \mathbf{v}_{\pi' 2})$. Thus $\pi 0 = \pi' 1$, $\pi 1 = \pi' 0$, $\pi 2 = \pi' 2$ and $\pi 3 = \pi' 3$, i.e. $(\pi')^{-1} \circ \pi = (0\ 1) \notin A_4$ (impossible).

Then we show that

$$\begin{aligned} &\overline{(\mathbf{v}, \mathbf{u}, \mathbf{a}_i, \mathbf{a}_{i+1})} = o(\mathbf{vua}_i\mathbf{a}_{i+1}) \Rightarrow \\ &\overline{(\mathbf{v}, \mathbf{u}, \mathbf{a}_{i+1}, \mathbf{a}_{i+2})} = o(\mathbf{vua}_{i+1}\mathbf{a}_{i+2}). \end{aligned} \tag{B.2}$$

We have $\overline{(\mathbf{u}, \mathbf{v}, \mathbf{a}_{i+1}, \mathbf{a}_i)} = \overline{(\mathbf{v}, \mathbf{u}, \mathbf{a}_i, \mathbf{a}_{i+1})} = o(\mathbf{vua}_i\mathbf{a}_{i+1})$. Now we use Eq. B.1 with adjacent tetrahedra $\mathbf{uva}_{i+1}\mathbf{a}_i$ and $\mathbf{uva}_{i+1}\mathbf{a}_{i+2}$, which are consistently oriented, and obtain $\overline{(\mathbf{v}, \mathbf{u}, \mathbf{a}_{i+1}, \mathbf{a}_{i+2})} = o(\mathbf{vua}_{i+1}\mathbf{a}_{i+2})$. □

Last we show Theorem 2.

Proof. Assume that D is a directed cycle and show that $\mathbf{v}$ is regular in ∂O. U is a cycle since D is a directed cycle. Then Lemma 2 implies that $\mathbf{v}$ is regular in ∂O.

Conversely, assume that $\mathbf{v}$ is regular in ∂O and show that D is a directed cycle. Thanks to Lemma 2, U is a cycle. Since D cannot include two inverse edges (Lemma 6), we only need to show that there are edges $(\mathbf{a}, \mathbf{u})$ and $(\mathbf{u}, \mathbf{b})$ in D for every vertex $\mathbf{u} \in c(U)$.

Let a vertex $\mathbf{u} \in c(U)$. Since T is a 3D Delaunay triangulation, there are distinct vertices $\mathbf{a}_i$ such that

$$T_{\mathbf{uv}} = \{\mathbf{uva}_0\mathbf{a}_1, \mathbf{uva}_1\mathbf{a}_2, \mathbf{uva}_2\mathbf{a}_3, \cdots \mathbf{uva}_k\mathbf{a}_{k+1}\}, \tag{B.3}$$

with a possible exception $\mathbf{a}_0 = \mathbf{a}_{k+1}$. Thanks to Lemma 1, if $\mathbf{uva}_i\mathbf{a}_{i+1} \in O$ and $\mathbf{uva}_j \in \partial O$ and $j \in \{i, i+1\}$, $\mathbf{ua}_j \in U$. Since U is a cycle, the number of such $\mathbf{uva}_j$ is exactly 2. Thus we have l and m such that

$$T_{\mathbf{vu}} \cap O = \{\mathbf{vua}_i\mathbf{a}_{i+1}, l \le i < i+1 \le m\} \tag{B.4}$$

and $\mathbf{vua}_l \in \partial O$ and $\mathbf{vua}_m \in \partial O$. Fig. B.23 shows $T_{\mathbf{vu}} \cap O$ if $l = 1$ and $m = 4$. Now we have two cases : $o(\mathbf{vua}_l\mathbf{a}_{l+1})$ is $\overline{(\mathbf{v}, \mathbf{u}, \mathbf{a}_l, \mathbf{a}_{l+1})}$ or $\overline{(\mathbf{u}, \mathbf{v}, \mathbf{a}_l, \mathbf{a}_{l+1})}$.

In case 1, $\overline{(\mathbf{v}, \mathbf{u}, \mathbf{a}_l, \mathbf{a}_{l+1})} = o(\mathbf{vua}_l\mathbf{a}_{l+1})$ and we show that $(\mathbf{u}, \mathbf{a}_l) \in D$ and $(\mathbf{a}_m, \mathbf{u}) \in D$. According to Eq. 8 and since $\overline{(\mathbf{v}, \mathbf{a}_{l+1}, \mathbf{u}, \mathbf{a}_l,)} = \overline{(\mathbf{v}, \mathbf{u}, \mathbf{a}_l, \mathbf{a}_{l+1})}$ and $\mathbf{vua}_l \in \partial O$, $(\mathbf{u}, \mathbf{a}_l) \in D$. Thanks to Lemma 7 and $\overline{(\mathbf{v}, \mathbf{u}, \mathbf{a}_l, \mathbf{a}_{l+1})} = o(\mathbf{vua}_l\mathbf{a}_{l+1})$, we have $\overline{(\mathbf{v}, \mathbf{u}, \mathbf{a}_{m-1}, \mathbf{a}_m)} = o(\mathbf{vua}_{m-1}\mathbf{a}_m)$. Since $\overline{(\mathbf{v}, \mathbf{a}_{m-1}, \mathbf{a}_m, \mathbf{u})} = \overline{(\mathbf{v}, \mathbf{u}, \mathbf{a}_{m-1}, \mathbf{a}_m)}$ and $\mathbf{vua}_m \in \partial O$, Eq. 8 implies $(\mathbf{a}_m, \mathbf{u}) \in D$.

In case 2, $\overline{(\mathbf{u}, \mathbf{v}, \mathbf{a}_l, \mathbf{a}_{l+1})} = o(\mathbf{vua}_l\mathbf{a}_{l+1})$ and we show that $(\mathbf{a}_l, \mathbf{u}) \in D$ and $(\mathbf{u}, \mathbf{a}_m) \in D$. According to Eq. 8 and since $\overline{(\mathbf{v}, \mathbf{a}_{l+1}, \mathbf{a}_l, \mathbf{u})} = \overline{(\mathbf{u}, \mathbf{v}, \mathbf{a}_l, \mathbf{a}_{l+1})}$ and $\mathbf{vua}_l \in \partial O$, $(\mathbf{a}_l, \mathbf{u}) \in D$. Thanks to Lemma 7 and $\overline{(\mathbf{u}, \mathbf{v}, \mathbf{a}_l, \mathbf{a}_{l+1})} = o(\mathbf{vua}_l\mathbf{a}_{l+1})$, we have $\overline{(\mathbf{u}, \mathbf{v}, \mathbf{a}_{m-1}, \mathbf{a}_m)} = o(\mathbf{vua}_{m-1}\mathbf{a}_m)$. Since $\overline{(\mathbf{v}, \mathbf{a}_{m-1}, \mathbf{u}, \mathbf{a}_m)} = \overline{(\mathbf{u}, \mathbf{v}, \mathbf{a}_{m-1}, \mathbf{a}_m)}$ and $\mathbf{vua}_m \in \partial O$, Eq. 8 implies $(\mathbf{u}, \mathbf{a}_m) \in D$. □

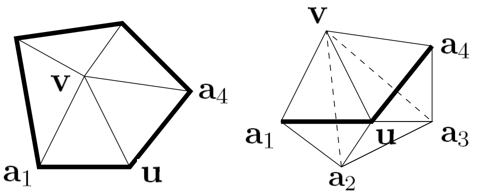

Figure B.23: Notations for Theorem 2 proof with $l = 1$ and $m = 4$. Left: the triangles in ∂O including the vertex $\mathbf{v}$. Here $\mathbf{v}$ is regular in ∂O. Right: the tetrahedra in $T_{\mathbf{uv}} \cap O$. We have $\mathbf{uva}_i\mathbf{a}_{i+1} \in O$ where $1 \leq i < 4$, $\mathbf{vua}_1 \in \partial O$, $\mathbf{vua}_4 \in \partial O$. In both cases, the bold edges are the U edges, i.e. the $\mathbf{v}$-opposite edges in triangles of ∂O.

Appendix C. Proof of Lemma 5

First let a triangle $t \in \partial(O \cup \tilde{H}) \setminus \partial O$ and show $t \in \partial\tilde{H}$. Let Δ be the only tetrahedron in $O \cup \tilde{H}$ including t. (i.e. t is a face of Δ.) Since $t \notin \partial O$, $\Delta \notin O$. Thus Δ is the only tetrahedron in $\tilde{H}$ including t: $t \in \partial\tilde{H}$.

Second let a triangle $t \in \partial O \setminus \partial(O \cup \tilde{H})$ and show $t \in \partial\tilde{H}$. Let Δ be the only tetrahedron in O including t. If Δ is the only tetrahedron in $O \cup \tilde{H}$ including t, $t \in \partial(O \cup \tilde{H})$, which is impossible. Thus there is another tetrahedron $\Delta' \in O \cup \tilde{H}$ including t. Since Δ is unique, $\Delta' \notin O$. Thus $\Delta' \in \tilde{H}$. Furthermore $\tilde{H} \subseteq F \setminus O$ and $\Delta \in O$ imply $\Delta \notin \tilde{H}$. We obtain $t \in \partial\tilde{H}$.

Last let a triangle $t \in \partial\tilde{H}$ and show $t \in J$. Let Δ be the only tetrahedron in $\tilde{H}$ including t. If Δ is the only tetrahedron in $O \cup \tilde{H}$ including t, $t \in \partial(O \cup \tilde{H})$. Since $\Delta \in \tilde{H} \subseteq F \setminus O$, $\Delta \notin O$. Thus $t \notin \partial O$. We obtain $t \in \partial(O \cup \tilde{H}) \setminus \partial O \subseteq J$. Otherwise, there is another tetrahedron $\Delta' \in O \cup \tilde{H}$ including t. Since Δ is unique, $\Delta' \notin \tilde{H}$. Thus $\Delta' \in O$. Since $\Delta \notin O$, $t \in \partial O \setminus \partial(O \cup \tilde{H}) \subseteq J$.

Appendix D. Shrink-Grow (S.G.) algorithm

Algorithm 5 is Shrink-Grow. (S.G. is summarized in Sec. 2.2.2.) This algorithm is repeated until $f(O)$ does not change or a maximum number of iterations is reached. It uses a function Growing(Δ) which updates O by Shelling using Δ as the first tetrahedron that we try to add to O (i.e. Algorithm 1 of [22] with inputs $Q_0 = \{\Delta\}$ and $O \neq \emptyset$). The function Growing also returns the set of tetrahedra that it adds to O.

Algorithm 5. *Shrink-Grow (one iteration)*

Let L_α be defined by Eq. 16;
Let $G_\alpha = \{\Delta \in F, c(\Delta) \cap L_\alpha \neq \emptyset\}$;
for each vertex $\mathbf{v} \in c(\partial O) \cap c(G_\alpha)$ {
  $L_{sub} = (T_\mathbf{v} \cap O)$;
  $L_{seed} = (T_\mathbf{v} \setminus O) \cap G_\alpha$;
  $O \leftarrow O \setminus L_{sub}$;
  if ($L_{seed} \neq \emptyset$ && $L_{sub} \neq \emptyset$ &&
    every vertex in $c(L_{sub})$ is regular in ∂O) {
    $L_{add} = \emptyset$;
    for each tetrahedron $\Delta \in L_{seed}$
      $L_{add} \leftarrow L_{add} \cup$ Growing(Δ);
    $R_{sub} = \sum_{\Delta \in L_{sub}} f(\Delta)$; // f is defined in Sec. 2.2.1
    $R_{add} = \sum_{\Delta \in L_{add}} f(\Delta)$;
    if ($R_{sub} > R_{add}$) // abandon if $f(O)$ decreases
      { $O \leftarrow O \setminus L_{add}$; $O \leftarrow O \cup L_{sub}$; }
  } **else** $O \leftarrow O \cup L_{sub}$; // abandon if ∂O is not manifold
}

end

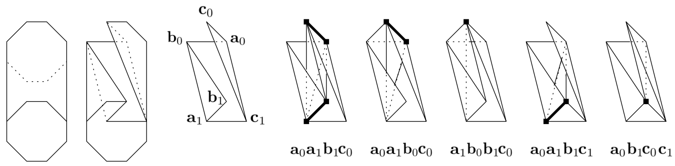

Figure E.24: Illustration for Theorem 6. From left to right: $O \cup P$, O, $c(O) \cap c(P)$, $c(O) \cap c(P)$ and a tetrahedron $\Delta \in P$ and all singularities of $\partial(O \cup \{\Delta\})$ in the 5 cases. Bold edges and black dots are singular. In this example, $O \cup P$ is a cylinder with an octagonal base.

Appendix E. Shelling blocking using five tetrahedra

Let $P \subseteq F$ be defined (as in Sec. 6.1 of [20]) by

$$\{\mathbf{a}_0\mathbf{a}_1\mathbf{b}_1\mathbf{c}_0, \mathbf{a}_0\mathbf{a}_1\mathbf{b}_0\mathbf{c}_0, \mathbf{a}_1\mathbf{b}_0\mathbf{b}_1\mathbf{c}_0, \mathbf{a}_0\mathbf{a}_1\mathbf{b}_1\mathbf{c}_1, \mathbf{a}_0\mathbf{b}_1\mathbf{c}_0\mathbf{c}_1\} \quad \text{(E.1)}$$

using distinct vertices $\mathbf{a}_0, \mathbf{a}_1, \mathbf{b}_0, \mathbf{b}_1, \mathbf{c}_0$ and $\mathbf{c}_1$. We summarize P: it has one "internal" tetrahedron $\mathbf{a}_0\mathbf{a}_1\mathbf{b}_1\mathbf{c}_0$ since every triangle face of $\mathbf{a}_0\mathbf{a}_1\mathbf{b}_1\mathbf{c}_0$ is included in another tetrahedron of P having vertex $\mathbf{b}_0$ or $\mathbf{c}_1$. Furthermore,

$$\begin{aligned} \partial P \quad = \quad & \{\mathbf{a}_0\mathbf{b}_0\mathbf{c}_0, \mathbf{a}_1\mathbf{b}_1\mathbf{c}_1\} \cup \{\mathbf{a}_0\mathbf{a}_1\mathbf{b}_0, \mathbf{a}_1\mathbf{b}_0\mathbf{b}_1\} \cup \\ & \{\mathbf{b}_0\mathbf{b}_1\mathbf{c}_0, \mathbf{b}_1\mathbf{c}_0\mathbf{c}_1\} \cup \{\mathbf{a}_0\mathbf{c}_0\mathbf{c}_1, \mathbf{a}_0\mathbf{a}_1\mathbf{c}_1\}. \quad \text{(E.2)} \end{aligned}$$

is homeomorphic to a sphere. (∂P is an octahedron.) Assume that $O \subseteq F$ such that ∂O is a manifold and

$$c(O) \cap c(P) = c(\{\mathbf{a}_0\mathbf{a}_1\mathbf{b}_0, \mathbf{a}_1\mathbf{b}_0\mathbf{b}_1, \mathbf{a}_0\mathbf{c}_0\mathbf{c}_1, \mathbf{a}_0\mathbf{a}_1\mathbf{c}_1\}). \quad \text{(E.3)}$$

Let $O' = O \cup P$. Since $O \cap P = \emptyset$, $P = O' \setminus O$. Using these choices, all conditions in Sec. 6.2 are meet: we have $\emptyset \neq O \subset O' \subseteq F$, ∂O is a manifold, $\partial(O' \setminus O)$ is a sphere, and $c(O) \cap c(O' \setminus O)$ is a disc.

Theorem 6. *If $\Delta \in P$, then $\partial(O \cup \{\Delta\})$ has a singular vertex.*

According to Fig. E.24 and Theorem 6 (proof in file **Proofs.pdf** of the supplementary material), the Shelling cannot start from O by using tetrahedra in $O' \setminus O$. Theorem 16 of [20] provides other examples of Shelling blocking.

Appendix E.1. Escape thanks to Critical Edge Removal

Here we check that the growing from O to O' can be done thanks to C.E.R. (Sec. 5.3.2): apply C.E.R. to the edge $\mathbf{a}_0\mathbf{a}_1$ if $\mathbf{a}_0\mathbf{a}_1 \in L_\alpha$. Let $O_0 = O$. The Force step of C.E.R. provides

$$O_1 = O_0 \cup \{\mathbf{a}_0\mathbf{a}_1\mathbf{b}_1\mathbf{c}_0, \mathbf{a}_0\mathbf{a}_1\mathbf{b}_0\mathbf{c}_0, \mathbf{a}_0\mathbf{a}_1\mathbf{b}_1\mathbf{c}_1, \}. \quad \text{(E.4)}$$

The Repair(2) step can choose $O_2 = O_1 \cup \{\mathbf{a}_1\mathbf{b}_0\mathbf{b}_1\mathbf{c}_0\}$ and $O_3 = O_2 \cup \{\mathbf{a}_0\mathbf{b}_1\mathbf{c}_0\mathbf{c}_1\}$, or $O_2 = O_1 \cup \{\mathbf{a}_0\mathbf{b}_1\mathbf{c}_0\mathbf{c}_1\}$ and $O_3 = O_2 \cup \{\mathbf{a}_1\mathbf{b}_0\mathbf{b}_1\mathbf{c}_0\}$. Let V_i be the set of the singular vertices of ∂O_i.

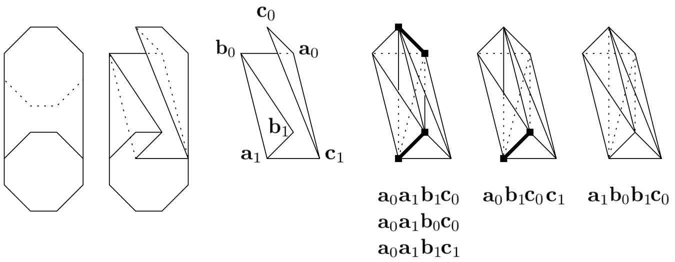

Figure E.25: Illustration for Theorem 7. From left to right: $O_3 = O \cup P$, $O_0 = O$, $c(O) \cap c(P)$, $c(O) \cap c(P)$ and $O_1 \setminus O_0$ and V_1, $c(O) \cap c(P)$ and $O_2 \setminus O_0$ and V_2, $O_3 \setminus O_0$ and V_3. Bold edges are singular and black dots are in V_i. In this example, $O \cup P$ is a cylinder with an octagonal base.

Table F.4: We redo Tab. 1 by using T.E. of [24] instead of T.E. of [22]. The former generates a lot of topological noise compared to the latter.

step	method	O/F	c	g	time
T.E.	M0	84.41%	978	2137	20s
C.H.R.	M0	85.44%	1034	3072	9.3s
P.R.	M0	80.01%	59	2672	15.9s
T.E.	M1	84.41%	978	2137	20s
C.H.R.	M1	85.16%	1049	1892	11.9s
P.R.	M1	79.65%	60	1616	15.8s

Theorem 7. *We have* $\emptyset = V_3 \subseteq V_2 \subseteq V_1$.

According to Fig. E.25 and Theorem 7 (proof in file **Proofs.pdf** of the supplementary material), the Repair(2) step of C.E.R. is successful.

Appendix F. Use T.E. of [24] instead of T.E. of [22]

In this paper, we use T.E. of [22]: if $\mathbf{v} \in c(\partial O)$ and $T_\mathbf{v} \subseteq F$ and $\partial(O \cup T_\mathbf{v})$ is manifold, then update $O \leftarrow O \cup T_\mathbf{v}$. However there is another T.E. in algorithm IV.2 of [24]: if $\mathbf{v} \in c(\partial O)$ and $\partial(O \cup (T_\mathbf{v} \cap F))$ is manifold, then update $O \leftarrow O \cup (T_\mathbf{v} \cap F)$. Since the latter is less constrained than the former, it provides the largest O growing. (See Tab. F.4.) However the latter generates a lot of topological noise (g is multiplied by 5.7.) and its computation time is 5 times larger than that of the former. As a consequence, the g of M0 increases by 72% and the g of M1 is multiplied by 4. M2 is unchanged since it does not use T.E. As in Sec. 7.5.1, we see a trade-off between visibility consistency and topology simplicity: $M0 > M1 > M2$ for both O/F and g.

References

[1] Amenta, N., Bern, M., 1999. Surface reconstruction by voronoi filtering. Discrete Computational Geometry 22.

[2] Bodis-Szomoru, A., Riemenschneider, H., Van Gool, L., 2017. Efficient edge-aware surface mesh reconstruction for urban scenes. Computer Vision and Image Understanding 157.

[3] Boissonnat, J., 1984. Geometric structures for three-dimensional shape representation. ACM Transactions on Graphics 3.

[4] Boissonnat, J., Devillers, O., Pion, S., Teillaud, M., Yvinec, M., 2002. Triangulations in cgal. Computational Geometry: Theory and Applications 22.

[5] Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Levy, B., 2010. Polygon mesh processing. AK Peters.

[6] Danaraj, G., Klee, V., 1978. A representation of 2-dimensional pseudo-manifolds and its use in the design of a linear-time shelling processing, in: Algorithmic aspects of combinatorics, vol. 2 of Ann. Discrete Math.

[7] Dyn, N., Hormann, K., Kim, S., Levin, D., 2000. Optimizing 3d triangulations using discrete curvature analysis, in: Mathematical Methods for Curves and Surfaces.

[8] Faugeras, O., Le Bras-Mehlman, E., Boissonnat, J., 1990. Representing stereo data with the delaunay triangulation. Artificial Intelligence 44.

[9] Fraundorfer, F., Schindler, K., Bischof, H., 2006. Piecewise planar scene reconstruction from sparse correspondences. Image and Vision Computing 24.

[10] Gueziec, A., Taubin, G., Lazarus, F., Horn, B., 2001. Cutting and stitching: converting sets of polygons to manifold surface. IEEE Transactions on Visualization and Computer Graphics 7.

[11] Hagbi, N., El-Sana, J., 2010. Carving for topology simplification of polygonal meshes. Computer-Aided Design 42.

[12] Hernandez Esteban, C., Schmitt, F., 2004. Silhouette and stereo fusion of 3d object modeling. Computer Vision and Image Understanding 96.

[13] Hilton, A., 2005. Scene modeling from sparse 3d data. Image and Vision Computing 23.

[14] Hoppe, C., Klopschitz, M., Donoser, M., Bischof, H., 2013. Incremental surface extraction from sparse structure-from-motion point clouds, in: British Machine Vision Conference.

[15] Hornung, A., Kobbelt, L., 2006. Robust reconstruction of watertight 3d models from non-uniformly sampled point clouds without normal estimation, in: Eurographics Symposium on Geometry Processing.

[16] Jancosek, M., Pajdla, T., 2011. Multi-view reconstruction preserving weakly-supported surfaces, in: IEEE Conference on Computer Vision and Pattern Recognition.

[17] Kobbelt, L., Vorsatz, J., Labsik, U., Seidel, H., 1999. A shrink wrapping approach to remeshing polygonal surfaces, in: Eurographics.

[18] Lhuillier, M., 2011. A generic error model and its application to automatic 3d modeling of scenes using a catadioptric camera. International Journal of Computer Vision 91.

[19] Lhuillier, M., 2015. 2-manifold tests for 3d delaunay triangulation-based surface reconstruction. Journal of Mathematical Imaging and Vision 51.

[20] Lhuillier, M., 2017. Overview of shelling for 2-manifold surface reconstruction based on 3d delaunay triangulation. Journal of Mathematical Imaging and Vision 59.

[21] Lhuillier, M., Quan, L., 2002. Match propagation for image-based modeling rendering. IEEE Transaction on Pattern Analysis and Machine Intelligence 24.

[22] Lhuillier, M., Yu, S., 2013. Manifold surface reconstruction of an environment from sparse structure-from-motion data. Computer Vision and Image Understanding 117.

[23] Ling, Y., Shen, S., 2017. Building maps for autonomous navigation using sparse visual slam features, in: IEEE/RSJ International Conference on Intelligent Robots and Systems.

[24] Litvinov, V., 2015. Incremental reconstruction of a complex scene using omnidirectional camera. Ph.D. thesis. Universite Blaise Pascal. France.

[25] Litvinov, V., Lhuillier, M., 2014. Incremental solid modeling from sparse structure-from-motion data with improved visual artifact removal, in: IAPR International Conference on Pattern Recognition.

[26] Lovi, D., Birkbeck, N., Cobzas, D., Jagersand, M., 2010. Incremental free-space carving for real-time 3d reconstruction, in: International Symposium on 3D Data Processing, Visualization and Transmission.

[27] Morris, D., Kanade, T., 2000. Image-consistent surface triangulation, in: IEEE Conference on Computer Vision and Pattern Recognition.

[28] Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., Sayd, P., 2007. Generic and real-time structure from motion, in: British Machine Vision Conference.

[29] Nguyen, T., Lhuillier, M., 2017. Self-calibration of omnidirectional multi-cameras including synchronization and rolling shutter. Computer Vision and Image Understanding 162.

[30] Oswald, M., Stuhmer, J., Cremers, D., 2014. Generalized connectivity constraints for spatio-temporal 3d reconstruction, in: European Confer-

ence on Computer Vision.

[31] Pan, Q., Reitmayr, G., Drummond, T., 2009. Proforma: probabilistic feature-based online rapid model acquisition, in: British Machine Vision Conference.

[32] Piazza, E., Romanoni, A., Matteucci, M., 2018. Real-time cpu-based large-scale three-dimensional mesh reconstruction. IEEE Robotics and Automation Letters 3.

[33] Romanoni, A., Delaunoy, A., Matteucci, M., Pollefeys, M., 2016. Automatic 3d reconstruction of manifold meshes via delaunay triangulation and mesh sweeping, in: IEEE Winter Conference on Applications of Computer Vision.

[34] Rossignac, J., Cardoze, D., 1999. Matchmaker: manifold brep for non-manifold r-sets, in: ACM Symposium on Solid Modeling and Applications.

[35] Sakkalis, T., Shen, G., Patrikalakis, N., 2000. Representational validity of boundary representation models. Computer-Aided Design 32.

[36] Sugiura, T., Torii, A., Okutomi, M., 2013. 3d surface extraction using incremental tetrahedra carving, in: ICCV Workshop.

[37] Taylor, C., 2003. Surface reconstruction from feature based-stereo, in: International Conference on Computer Vision.

[38] Teillaud, M., 1999. Three dimensional triangulation in cgal, in: Abstracts Europeans Workshop on Computational Geometry.

[39] Vogiatzis, G., Torr, P., Cipolla, R., 2003. Bayesian stochastic mesh optimization for 3d reconstruction, in: British Machine Vision Conference.

[40] Vu, H., Labatut, P., Pons, J., Keriven, R., 2012. High accuracy and visibility-consistent dense multiview stereo. IEEE Transaction on Pattern Analysis and Machine Intelligence 34.

[41] Werner, T., Zisserman, A., 2002. New techniques for automated architecture reconstruction from photographs, in: European Conference on Computer Vision.

[42] Wood, Z., Hoppe, H., Desbrun, M., Schroder, P., 2004. Removing excess topology from isosurfaces. ACM Transactions on Graphics 23.

[43] Wu, C., Agarwal, S., Curless, B., Seitz, S., 2012. Schematic surface reconstruction, in: IEEE Conference on Computer Vision and Pattern Recognition.

[44] Younes, G., Asmar, D., Shammas, E., Zelek, J., 2017. Keyframe-based monocular slam: design, survey and future directions. Robotics and Autonomous Systems 98.

[45] Yousif, K., Bab-Hadiashar, A., Hoseinnezhad, R., 2015. An overview to visual odometry and visual slam: applications to mobile robotics. Intelligent Industrial Systems 1.

[46] Zhou, Q., Ju, T., Hu, S., 2007. Topology repair of solid models using skeletons. IEEE Transactions on Visualization and Computer Graphics 13.

[47] Ziegler, G., 1998. Shelling polyhedral 3-balls and 4-polytopes. Discrete Computational Geometry 19.

[48] Zomorodian, A., 2005. Topology for computing. Cambridge University Press, New York, NY.

[49] Zou, M., Holloway, M., Carr, N., Ju., T., 2015. Topology-constrained surface reconstruction from cross-sections. ACM Transaction of Graphics 34.