Incremental Solid Modeling from Sparse Structure-from-Motion Data with Improved Visual Artifacts Removal

Vadim Litvinov and Maxime Lhuillier
Institut Pascal, UMR 6602, CNRS/UBP/IFMA, Aubière, France
http://maxime.lhuillier.free.fr

Abstract—In the recent years, a family of 2-manifold surface reconstruction methods from a sparse Structure-from-Motion points cloud based on 3D Delaunay triangulation was developed. This family consists of batch and incremental variations which include a step that remove visual artifacts. Although been necessary in the term of surface quality, this step is slow compared to the other parts of the algorithm and is not well suited to be used in an incremental manner. In this paper, we present two other methods for removing visual artifacts. They are evaluated and compared to the previous one in the incremental context where the need of new methods is the highest. Taken separately, they provide medium results, but used together they are as good as the old method in the terms of surface quality, and at the same time, processing time is almost three times smaller.

Keywords—3D shape recovery, Stereo and multiple view geometry, Reconstruction and camera motion estimation

I. INTRODUCTION

Despite being around for quite a long time, surface reconstruction is still an active research topic. In the last few years, several methods were developed to perform solid modeling from a sparse Structure-from-Motion (SfM) point cloud, both in batch [1] and incremental [2] frameworks. The main interest of this family of methods is their time and space complexities due to the fact that a dense stereo step is unnecessary. Furthermore, this series of algorithms produces a 2-manifold output which is useful to initialize a more precise, but heavy, dense reconstructions or simply allow to perform efficient surface smoothing. Other surface reconstruction methods [3]–[5] work in a similar (sparse) context but produce non manifold surfaces.

For memory, a surface has the 2-manifold property if and only if every point of it has a neighborhood which is topologically a disk, i.e. each triangle of the surface is exactly connected by its three edges to three other triangles, the surface has neither holes nor self-intersections and it divides the space into separate regions. If the surface is 2-manifold, its genus is defined as the number of its handles. For example, the genus of a sphere is 0, that of a torus is 1, etc.

The incremental method [2] considered in this paper is a sculpting method in a 3D Delaunay triangulation. It takes a sparse cloud of 3D points sampled on an unknown surface along with their visibility information produced by SfM. The 3D Delaunay triangulation $T$ of $P$ is a partition of the convex hull of $P$ such as the circumscribing sphere of every tetrahedron does not contain a vertex in its interior. All the tetrahedra of $T$ are labeled free-space or matter using the visibility constraints. A ray is defined as a line segment between a camera and a point observed by this camera. A tetrahedron is free-space if it is intersected by at last one ray.

The list of triangles separating free-space and matter can, at first glance, be directly considered as the output surface, but the resulting surface is not guaranteed to be a manifold. To enforce the topology constraint, a region growing approach as in [6] can be used, but alone it is limited to genus 0 surfaces. So methods [1], [2] expand it by a topology extension step.

However, the region growing and topology extension steps alone suffer from a visual artifacts problem as illustrated on Fig. 1. The artifact that connects the wall of the building to the ground on the left part of the figure does not exist in reality and should be removed as shown on the central part. This is only one example of situations when the problem arise. It can also occur in other contexts [7].

There were several previous attempts to solve this problem. First of all, computational geometry only approaches [8], [9] that use scanner data consider that spurious handles (a particular, but very usual kind of artifact) are usually small by contrast to the real world handles that would be large. Unfortunately this is not always true in our case. Another idea [1] is to use the visibility information provided by SfM.

The artifact removal method in Sec. 4 of [1] is efficient in terms of resulting surface quality, but, unfortunately, it is the slowest part of the calculation if the input is large enough. Another limitation is the fact that this algorithm has an artificial (Steiner) points addition step that breaks Delaunay property of the triangulation. Although this is not really a problem in the batch case [1], this is not recommended in the incremental framework [2] where the triangulation is updated by vertex additions after every artifact removal step.

![Fig. 1. A visual artifact (left) and its removal (middle and right). The triangle orientations are colored (white: ground, red-green-blue: vertical, black: sky).](image-url)
Indeed, the Delaunay triangulation has theoretical warranties for surface reconstruction in both geometric senses. We also benefit by the fast and incremental 3D Delaunay triangulation in the CGAL library.

The main contribution of this paper is two other visual artifacts removal methods that are faster than the previous one and don’t require Steiner points addition. Another contribution is a comparison between these methods applied in the incremental framework.

Sec. II summarizes the batch and incremental surface reconstruction methods. We remind previous visual artifacts removal method in Sec. III-A and then explain in detail the two new methods proposed by this paper in the remaining of Sec. III. Finally, we compare these methods in Sec. IV.

II. SUMMARY OF SOLID MODELING METHODS

In this section, we summarize the batch and incremental solid modeling methods. The first and common step of the two algorithms is Harris points extraction and their 3D reconstruction by a Structure-from-Motion algorithm. Let \( P \) be a set of 3D points reconstructed by SfM, \( C \) a list of camera locations. Let \( R \) be the list of rays defined as follow: \( \forall c_i \in C, \forall p_j \in P \), segment \( c_i p_j \in R \) if and only if \( c_i \) has observed \( p_j \).

\[ A. \text{ Batch surface reconstruction method} \]

Here is a brief overview of the batch method in [1]. First of all, a 3D Delaunay triangulation \( T \) of \( P \) is constructed (\( T \) is a list of tetrahedra). For all \( \Delta \in T \), we define \( I(\Delta) \) as the number of intersections between \( \Delta \) and the segments of \( R \). Each tetrahedron \( \Delta \in T \) is labeled free-space or matter: \( \Delta \) is free-space if and only if \( I(\Delta) \neq 0 \). We compute the list \( F \subseteq T \) of the free-space tetrahedra.

As the next step, we define another partition of \( T \). Let \( O \subseteq F \) such as the border \( \partial O \) of \( O \) is 2-manifold (\( \partial O \) is the list of triangles which are faces of exactly one tetrahedron in \( O \)). The tetrahedra in \( O \) are outside and the others are inside.

Initially \( O = \emptyset \). We begin by the region growing step: for each \( \Delta \in F \), we add \( \Delta \) to \( O \) if \( \partial O \) remains manifold. This property is checked using a very fast test. We begin by adding tetrahedra with highest \( I(\Delta) \). This way, we begin by adding tetrahedra with a highest chance of being free-space.

Then, we use the topology extension step to allow the genus of the resulting surface be higher than zero. We add the tetrahedra \( \Delta \in F \) by packs of tetrahedra incident to a common vertex \( v \in P \). To ensure the manifold property, in this configuration, a slower but more general test is used.

We repeat these two steps until no more tetrahedra can be added to \( O \). The resulting surface is the border \( \partial O \) of \( O \). Some post-processing steps are then applied to achieve better surface quality (more details in [1]), including the artifact removal method which is summarized in Sec. III-A for paper clarity.

\[ B. \text{ Incremental surface reconstruction method} \]

To experiment our artifacts removal methods in an incremental context, we use the algorithm described in [2].

Let \( P_t \) be the set of points computed at time \( t \) that will not be modified by SfM after \( t \) (\( p \in P_t \) and \( c_i p \in R \) imply \( i \leq t \); \( i = t \) exists). At the keyframe/time \( t + 1 \), the algorithm has the following input: \( T_t \) is a 3D Delaunay triangulation of \( P_t \), \( F_t \subseteq T_t \) is a list of free-space tetrahedra of \( T_t \) and \( O_t \subseteq F_t \) is a list of outside tetrahedra such that \( \partial O_t \) is a 2-manifold.

The goal of the algorithm is to update the surface with a set of new SfM points \( P_{t+1} \). They can not directly be added into the triangulation \( T_t \) because this can modify \( O_t \) and break the manifold property of its border. So we compute a ball \( B \) centered on \( c_{t+1} \) with the radius big enough to enclose \( P_{t+1} \) and neighboring tetrahedra (see Fig. 2). Then we remove tetrahedra from \( O_t \) one-by-one and by packs in the similar manner than Sec. II-A until \( \partial O_t \) is still a manifold and we get as close as possible to \( O_t \cap B = \emptyset \). This is the shrinking step.

As the next step, we safely add \( P_{t+1} \) into \( T_t \) without modifying \( O_t \). We obtain \( T_{t+1} \), then, we update the free-space matters of tetrahedra and obtain \( F_{t+1} \subseteq T_{t+1} \).

Finally, to compute the new surface, we initialize \( O_{t+1} = O_t \) and we apply the region growing and topology extension algorithms described in Sec. II-A. To enhance the resulting surface quality, some post-processing steps are applied locally to the updated region (see [2] for details).

III. VISUAL ARTIFACTS REMOVAL METHODS

Before we begin the discussion about different artifacts removal methods, we should precisely define what we seek to remove. Using the definitions in Sec. II, the 3D Delaunay triangulation \( T \) is encoded by an adjacency graph \( \Gamma_T \) in which the nodes are the tetrahedra of \( T \) and the edges are the triangles between two tetrahedra. In the same manner, for any set \( S \subseteq T \) of tetrahedra, \( \Gamma_S \subseteq \Gamma_T \) is the corresponding adjacency graph.

We define a visual artifact \( A \) by \( A \subseteq F \setminus O \) and \( \Gamma_A \) is connected, i.e. a set of tetrahedra that is included in the inside volume \((A \cap O = \emptyset)\), but not in the scene matter \((A \subseteq F)\). Unfortunately, detecting all the artifacts in the resulting model using this definition alone would be too slow to be useful in practice. So we define a visually critical edge [1] as a segment \( ab \) such as \( a \in P, b \in P \) and \( 3c \in C \) such as \( abc > \alpha \) where \( \alpha \) is a user defined threshold (\( \alpha = 5 \) deg in this paper). Then, we define a visually critical artifact as a visual artifact which has (at least) a tetrahedron containing a visually critical edge.

The artifacts removal methods discussed in this section seek to remove as many visually critical artifacts as they can. We do not use the term spurious handle as in [1], because it is too restrictive. Our algorithms deal with more than just "handles."
A. Summary of previous method

In this section, we briefly discuss the visual artifacts removal method described in [1]. As its entry, this algorithm takes the list $E_o$ of the visually critical edges.

First of all, the method splits each edge of $E_o$ in its middle by adding a Steiner point (i.e. artificial point which wasn’t seen by any camera). Each edge is split into two parts and the resulting edges are stored in a list $E_{split}$. Each tetrahedron including the edge is also split in two and the two resulting tetrahedra are assigned the same number of intersections and status as the initial one.

This way, we reduce the size of tetrahedra and hope to locally unlock region growing in the neighborhood of this edge. Furthermore, $\partial O$ is still manifold. The drawback is that the triangulation is not guaranteed to be Delaunay anymore.

After this first step, we create a list $V_c$ of the end vertices of the edges of $E_{split}$. For each vertex $v \in V_c$, we apply a force/repair cycle as following.

We begin by the force step. Let $L_v$ be the list of the tetrahedra incident to $v$. We define $G = (L_o \cap F) \setminus O$. So $G$ is a set of tetrahedra forming a visual artifact. We force them into outside: $O \leftarrow O \cup G$. If the border of $O$ remains manifold, we get what we wanted. If it doesn’t, it contains some vertices whose neighborhood is not topologically a disk. These vertices are called singular vertices in the remaining of this paper. We call $n$ the number of these singular vertices. So, we try to repair $O$ using the next step.

Then, we execute the repair step. We seek to add a bunch of free-space tetrahedra to $O$ such that $\partial O$ become manifold once again. To achieve this goal, we apply a local region growing algorithm to $O$ in $\hat{F}$ starting in the neighborhood of $G$ and which decreases $n$. The algorithm stops if a number of iterations $g_o$ (fixed by the user) has been reached or no more tetrahedra can be added to $O$. If the repair step succeeds (i.e. $n = 0$; $\partial O$ is manifold), we was able to remove a visual artifact and proceed to the next vertex. Otherwise, we restore $O$ to the previous state.

B. New visual artifacts removal method

As was previously noted, the method in Sec. III-A is good to improve the surface quality, but unfortunately it has two problems. The first is the need of Steiner points for good results, which break the Delaunay property of the triangulation.

The second problem is that artifacts removal method is slow (48%-72% of the total processing time according to Tab. 2 of [1]). So, we propose a faster visual artifacts removal method that doesn’t require these Steiner points.

The region growing described in Sec. II is a greedy optimization algorithm that maximizes

$$f(O) = \sum_{\Delta \in O} I(\Delta)$$  \hspace{1cm} (1)

in the discrete search space of tetrahedra lists $O$ such that $O \subseteq F$ and $\partial O$ is a 2-manifold. In [1], we have a steepest descent heuristic for function $f(O)$ (by region growing of $O$) which can get stuck to a local maximizer of $f(O)$. A visual artifact (e.g. spurious handle) can be seen in this situation. The basic idea of our new method is to remove some tetrahedra from $O$ and so to decrease $f(O)$ to kick the algorithm out of its local extrema.

As in the previous section, we call $E_o$ the list of visually critical edges. Let $G_o \subseteq F$ such that every $\Delta \in G_o$ has an edge in $E_o$. Then, for every vertex $v$ of both $\partial O$ and $G_o$, we force neighboring tetrahedra out of $O$ and we try local region growing beginning from neighboring tetrahedra included in $G_o$. If the final value of $f(O)$ is greater than the initial one, we are able to escape from a local maximum, otherwise we revert everything to the initial state and try another vertex $v$. See Fig. 3 for an example and the algorithm below. Once we tried all $\partial O$ vertices, we complete the result using region growing and topology extension (Sec. II-A) restricted to the tetrahedra included in $B$ (Sec. II-B).

The algorithm in pseudo-code is as follows:

1: function REGION_GROWING($\Delta_o$)  \hspace{1cm} \triangleright Local region growing in $G_o$
2: let $Q$ be a priority queue of tetrahedra based on $I(\Delta)$
3: $L_{add} \leftarrow \emptyset$
4: while $Q \neq \emptyset$
5: \hspace{1cm} $\Delta \leftarrow \text{POP}(Q)$
6: \hspace{1cm} if $\Delta \notin O$ and $\partial(O \cup \{\Delta\})$ is manifold then
7: \hspace{1cm} $O \leftarrow O \cup \{\Delta\}$
8: \hspace{1cm} $L_{add} \leftarrow L_{add} \cup \{\Delta\}$
9: \hspace{1cm} for all $\Delta'$ neighbor of $\Delta$ do
10: \hspace{1cm} \hspace{1cm} \text{if } \Delta' \in G_o \text{ and } \Delta' \notin O \text{ then}
11: \hspace{1cm} \hspace{1cm} \text{PUSH}(Q, \Delta')$
12: \hspace{1cm} end if
13: end for
14: end if
15: end while
16: return $L_{add}$
17: end function

19: procedure ARTIFACTS_REMOVING  \hspace{1cm} \triangleright The main artifacts suppression algorithm
20: $s_{old}, s \leftarrow 0$
21: repeat
22: $s_{old} \leftarrow s$
23: \hspace{1cm} for all vertex $v$ of both $\partial O$ and $G_o$ do
24: \hspace{1cm} $N_v \leftarrow$ all the tetrahedra incident to $v$
25: \hspace{1cm} $L_{sub} \leftarrow O \cap N_v$
26: \hspace{1cm} $L_{seed} \leftarrow (G_o \cap N_v) \setminus O$
27: \hspace{1cm} $L_{add} \leftarrow \emptyset$
28: \hspace{1cm} end for
29: until $s_{old} = s$
30: end repeat

33: Procedure for ARTIFACTS_REMOVING
34: \hspace{1cm} for all $\Delta \in E_{split}$ and $\Delta \notin O$ do
35: \hspace{1cm} $L_v \leftarrow \Delta$
36: \hspace{1cm} end for
37: end procedure

Fig. 3. Escape from local extremum thanks to our new artifacts removal (2D case). White tetrahedra are outside, gray tetrahedra are inside. The number in tetrahedron $\Delta$ is $I(\Delta)$. The green dot is the vertex considered by the algorithm and thick red lines are critical edges. Left: $O$ before removal. Middle: force neighboring tetrahedra out of $O$. Right: local growing of $O$. 

$$f(O_3) > f(O_1) > f(O_2)$$
C. New spurious handle removal method

We also propose a visual artifacts removal algorithm that is a specialization of Sec. III-A method. The latter is slow mainly because it consists in many attempts to remove a small pack of tetrahedra and the majority of these attempts are unsuccessful. So, the idea is to remove bigger packs of tetrahedra by placing ourselves in a less general context.

Instead of trying to remove all the visually critical artifacts, we seek to remove a particular kind of artifact: the handle. The fourth column of Fig. 6 shows an example. This is the most visible kind of artifact for the human eye.

The basic idea of the algorithm is to remove the handle (by contrast to Sec. III-A where we remove all the tetrahedra including a vertex) and then try to restore the manifold property of $O$ by a local region growing. To do that, we begin as usual, by computing the list $E_\alpha$ of visually critical edges. For each edge $ab \in E_\alpha$, we check if it is contained in a handle.

We define a plane $\pi$ perpendicular to $ab$ and intersecting segment $ab$ in some point. In practice we try several planes intersecting $ab$ in $\frac{2a+b}{2}, \frac{a+3b}{4}$ and $\frac{a+2b}{4}$. Let $L_\alpha$ be the list of the tetrahedra intersected by $\pi$. Let $N_{ab}$ be the list of the tetrahedra including edge $ab$. A handle $H$ is a set of tetrahedra forming a visual artifact, so $H \subseteq F \setminus O$. With this definition, we begin to form our handle by $H \leftarrow (N_{ab} \cap L_\pi \cap F) \setminus O$.

Let $N_H$ be the list of the tetrahedra directly adjacent to the set $H$ (i.e. $\forall \Delta \in N_H, \Delta$ has a 4-neighbor in $H$). We iteratively grow $H$ by performing $H \leftarrow H \cup (N_H \cap L_\pi \cap F) \setminus O$ until no more tetrahedra can be added. Then we check that the final $H$ is surrounded by $O$ in plane $\pi$ (see Fig. 4), i.e. $\forall \Delta \in (N_H \cap L_\pi) \setminus O, \Delta \in O$.

Once we have detected a handle $H$, we try to remove it as in Sec. III-A. First we force $H$ in $O$ (i.e. $O \leftarrow O \cup H$), then we try to restore the 2-manifold property of $O \partial O$ using the repair step initialized by the tetrahedra in $(N_H \cap F) \setminus O$. If we succeed, we remove an artifact, otherwise we undo everything and try another edge in $E_\alpha$.

IV. EXPERIMENTS

A. Dataset

To perform a quantitative evaluation of the visual artifacts removal methods discussed in this paper, we use the SfM dataset in [2]. The video is a 2.5 km. long trajectory taken in a general urban environment by a PointGrey Ladybug omnidirectional camera. Our SfM is based on local bundle adjustment [13] and is refined by a global bundle adjustment to close the large loop (2.3 km). The SfM selects 1306 keyframes out of 7735 frames and detects 2,648 Harris interest points. 483k 3D points are reconstructed and shown in Fig. 5. Thus, we only have 193 points per meter to reconstruct the surface.

B. How to compare the artifact removal methods?

Every removal method is integrated in the surface post-processing step of the incremental surface reconstruction method [2] (before smoothing). To compare the methods in terms of output surface quality, we manually count the visually critical artifacts (Sec. III) remaining on the final surface. Fig. 6 shows examples of artifacts that are manually counted.
We evaluate five visual artifacts removal methods: None (no removal method), III.A (the method in [2] summarized in Sec. III-A), III.B (escape from local extremum using the method in Sec. III-B), III.C (handle removal using the method in Sec. III-C), III.B&C (use III.C after III.B). The results are summarized in Tab. I. The removal methods III.A, III.B, III.C and III.B&C provide similar improvements (increases) of the objective function $f$. The differences between them are small (less than 0.034%), but we see that III.A is slightly better than III.B and III.C taken separately. Furthermore, we see that the combination of III.B and III.C is slightly better than III.A, and this is confirmed by both the number of tetrahedra in $F' \setminus O$ (union of all visual artifacts) and the number of artifacts that are manually detected. Fig. 6 compares the results of all methods at four locations in the reconstruction and is consistent with these comparisons. We note that III.B has important visual artifacts (as the one in the left of this figure) although it has a good (small) $F' \setminus O$, and III.C can correct them in spite of its greater $F' \setminus O$. Then we think that the combination III.B&C is a good choice.

A difference between the four methods is the calculation time. Indeed, III.B, III.C and III.B&C are significantly faster than III.A since their mean time per keyframe is about 2.5-4 times smaller (Tab. I and Fig. 7). We also see that III.A needs the largest number of tests that check the manifold property at a vertex.

D. More details on III.B&C Result

Fig. 8 shows a global view of the 3D models reconstructed by the method in [2] including the artifact removal method III.B&C. At the end of the process, the surface has 600k triangles and the 3D Delaunay has 2.09M tetrahedra. The computation times for every keyframe is in Fig. 9. We use a 4xIntel Xeon W3530 at 2.8Ghz. The surface reconstruction time for every keyframe has mean 1.87 s. and maximal value 5.39 s. The largest times are at the sequence end, where the large loop is closed. As in [2], this is due to the fact that we reconstruct new points of the loop end at a location where points are already reconstructed at the loop beginning. The joint video shows the progressive surface reconstruction and the final surface.

We observe differences with [2]: the artifact removal step (previously named handle removal) is not the most costly step anymore, and the global times are roughly divided by 1.5-2.
These improvements are the consequences of both our artifact removals (our contribution) and reasons in the Appendix.

V. CONCLUSION

This paper improves the visual artifact removal step of a recent incremental method which reconstructs a 2-manifold surface from sparse Structure-from-Motion data. We replace this step by the same method restricted on particular cases of visual artifacts (handles) and preceded by another method which escapes from the local extrema of the objective function. Then the processing time of the step is greatly reduced without loss of surface quality. We experiment on a 2.5km. long video sequence taken by an omnidirectional camera moving in an urban environment.

Future work includes improvements of calculation time and surface quality, e.g. reconstruct and integrate the image contours in the 3D Delaunay triangulation, improve the interest point matching, investigate other objective cost functions.

REFERENCES


APPENDIX

Corrections are applied to the implementation in [2]. An implementation bug is corrected and the shrinking step is now slower but more accurate than before. We remember that the ideal result of shrinking meet \( O \cap B = \emptyset \) where \( B \) is a ball where almost all computations are done. Now this condition is not meet in only one of the 1306 keyframes (before: 4.5% of the keyframes). As a consequence, the step of the regular Steiner points grid can be reduced to 10 times mean distance between consecutive camera locations (before: 15 times). We remember that this grid is useful to reduce the size of \( B \) and these Steiner points do not break Delaunay property [2].

Moreover, the topology extension step of shrinking and growing is re-implemented in a multi-threaded fashion. As was briefly explained in Sec. II, this step consists mainly in trying to add or remove a pack of tetrahedra to the outside region. In the new implementation, we simply perform many tests simultaneously, so these steps are greatly accelerated.

ACKNOWLEDGEMENTS

Thanks to CNRS, FEDER and Auvergne Région for funding.