Incremental Solid Modeling from Sparse and Omnidirectional Structure-from-Motion Data

Vadim Litvinov, Maxime Lhuillier

Incremental Solid Modeling from Sparse and Omnidirectional Structure-from-Motion Data

Vadim Litvinov
vadim.litvinov@univ-bpclermont.fr
Maxime Lhuillier
http://maxime.lhuillier.free.fr

Institut Pascal
Université Blaise Pascal
Aubière, France

Abstract

This paper introduces a sparse and incremental 2-manifold surface reconstruction method. It uses a sparse 3D point cloud generated by a Structure-from-Motion algorithm (SfM) as its main input as opposed to the more common dense algorithms. Furthermore, our method is incremental: the surface is updated for every new camera pose computed by SfM, and the update occurs in a small neighborhood of the new camera pose. Compared to the other surface reconstruction methods, our method has the advantage to have all these properties at the same time. The quality and execution time of the proposed algorithm is evaluated on a large scale (2.5 km.) real sequence taken in an urban environment, and the method is quantitatively evaluated on a synthetic urban scene.

1 Introduction

The majority of methods which reconstruct the surface of an environment from images have a dense information as input (dense 3D point cloud or almost every image pixels). In contrast to these dense methods, the direct estimation of a surface from the sparse point cloud estimated by Structure-from-Motion step (SfM) is under-explored. The main argument against such a sparse method is that the lack of points implies an inaccurate scene surface. However, a sparse method has advantages. First, the quality (accuracy, robustness) of a 3D point is expected to be better than that of a dense stereo method, thanks to the SfM machinery involving interest point detection and bundle adjustment. Second, the resulting simplified cloud provides a simplified surface at low complexity that applications could accept: (pre)visualization with a small hardware, robot localization, initialization of a dense stereo method.

In this paper, we introduce a sparse and incremental method which reconstructs a triangulated manifold surface from SfM data. The method is incremental since the surface is locally updated for every new camera pose (and its 3D points) estimated by SfM. This is interesting for applications which require a surface while reading the video sequence. A triangulated manifold surface (shortened as "manifold") is a list of triangles in 3D such that the neighborhood of every surface point is topologically a disk. This property is needed to define surface normal and curvature [2], then it is used by a lot of algorithms like surface refinement involving regularization (smoothing [10], dense stereo [4], ...) and others [2, 12].

The majority of sparse methods are based on sculpting in a 3D Delaunay triangulation. Sec. 2 presents and discusses these methods. Then our method is described in Sec. 3 (prerequisites) and 4 (algorithm). Last Sec. 5 provides experiments and we conclude in Sec. 6.

2 Sculpting Methods in a 3D Delaunay Triangulation

Let P be a set of 3D points sampled on an unknown surface. The 3D Delaunay triangulation T of P is a list of tetrahedra such that (1) the tetrahedra partition the convex hull of P, (2) their vertex set is P, (3) the circumscribing sphere of every tetrahedron does not contain a vertex in its interior. In our paper, a vertex/edge/triangle is a face of a T tetrahedron, and we use notation $|L|$ for the union of triangles (or tetrahedra) in list L.

In the sculpting methods, a list V of tetrahedra (in T) represents the reconstructed object whose volume is $|V|$. Border δV is the list of triangles which are included in exactly one tetrahedra of V. Then $|\delta V|$ should be a manifold and the target surface. In the early work [1], P has no bad point and we should have $P \subset |V|$. This method initializes $V = T$, then it selects and removes a tetrahedron Δ from V while the following conditions are met: Δ has a triangle in δV, $|\delta(V \setminus \{\Delta\})|$ is manifold, $P \subset |V \setminus \{\Delta\}|$. An efficient rule checks that $|\delta(V \setminus \{\Delta\})|$ is manifold. However, the genus of $|\delta V|$ is always zero, *i.e.* $|\delta V|$ is always homeomorphic to a 2-sphere without handle/hole. The genus can be greater than 0 in [6, 7] since several tetrahedra are removed at once from V.

In our Computer Vision context, SfM provides additional visibility knowledge R_i: every point $\mathbf{p}_i \in P$ is computed from camera locations $\mathbf{c}_j$ where $j \in R_i$. This implies that $|\delta V|$ should not intersect the rays (line segments) $\mathbf{c}_j\mathbf{p}_i$, $j \in R_i$ except at $\mathbf{p}_i$. The tetrahedra intersected by a ray are labeled "freespace", the others are "matter". Then the triangles between freespace and matter are good candidates to be in δV. Thanks to the visibility, we expect to reconstruct surfaces with fewer points than the methods referenced above. However, some bad points can occur since they are estimated from images, and we should not enforce $P \subset |V|$.

Several sculpting methods [5, 13, 15, 18, 20] use visibility knowledge and sparse point cloud P estimated from images. The manifold constraint on $|\delta V|$ is not enforced by [13, 15, 18]. Method [15] is incremental, but it defines directly V as the list of the matter tetrahedra, then the resulting surface $|\delta V|$ is not manifold. Method [18] is real-time (for small objects) and deals with image noise, but it is not incremental. In [13], a non-incremental method estimates a surface minimizing a cost involving visibility and photoconsistency. The manifold constraint is enforced in [5, 20]. Method [5] is limited to genus zero surface. Method [20] is incremental but the time complexity of one iteration can be too large: if the camera trajectory is a loop, this complexity is at least linear to the number of camera poses in the loop.

Our contribution is an incremental sculpting method which estimates a manifold from sparse SfM data, without genus restriction and without prohibitive complexity in presence of loop in the camera trajectory. Note that other sparse methods [11, 16, 19] exist, but they can not be classified in this Section and do not estimate a manifold in an incremental scheme.

3 Prerequisites

3.1 Manifold Tests

According to p. 723 of [9], $|\delta V|$ is manifold if for every vertex $\mathbf{v}$ of every δV triangle, the **general test** is successful: the triangles in δV including $\mathbf{v}$ can be ordered as $t_0, \cdots t_{k-1}$ such

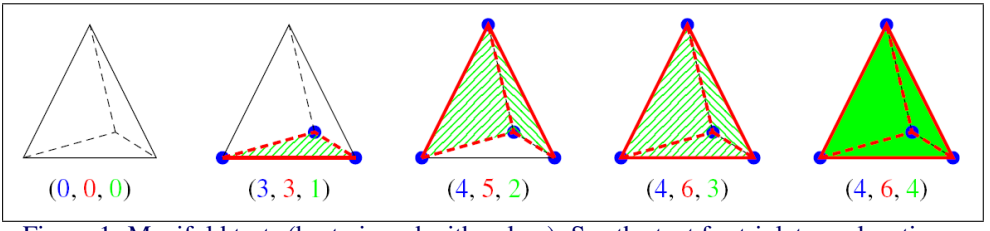

Figure 1: Manifold tests (best viewed with colors). See the text for triplets explanation.

that $t_i \cap t_{(i+1) \bmod k}$ is an edge, and such an edge is included in exactly two triangles t_i and t_j. In other words [8], the graph of the **v**-opposite edges in the δV triangles must be a cycle.

We complete the rule of [1] in our case where constraint $P \subset |V|$ is not used and obtain the **subtraction test**. Assume that $\Delta \in V$, $V \subseteq T$, and $|\delta V|$ is manifold. Then $|\delta(V \setminus \{\Delta\})|$ is manifold if the numbers of (vertices,edges,triangles) in $\Delta \cap |\delta V|$ is in $\{(0,0,0),(3,3,1),(4,5,2),(4,6,3),(4,6,4)\}$ (Fig. 1). Only cases (3,3,1) and (4,5,2) are in [1].

We also need the **addition test**, which is similar to the subtraction test (adding Δ to V is like subtracting Δ from $T \setminus V$). Assume that $\Delta \in T \setminus V$, $V \subseteq T$, and $|\delta V|$ is manifold. Then surface $|\delta(V \cup \{\Delta\})|$ is manifold if the numbers of (vertices,edges,triangles) in $\Delta \cap |\delta V|$ is in $\{(0,0,0),(3,3,1),(4,5,2),(4,6,3),(4,6,4)\}$ (Fig. 1).

If we would like to add a single tetrahedron Δ to V such that $|\delta V|$ is manifold, we can use the addition test or use the general test on triangles $\delta(V \cup \{\Delta\})$ at every vertex in $\Delta \cap |\delta V|$. In this case, the addition test is faster than using the general test. These tests can be converted to tests that are more convenient if T is implemented as the adjacency graph of the tetrahedra (*e.g.* see Appendix of [20] for the addition test).

3.2 Notations

At image (or time) $t+1$, our method has the following input:

- 3D Delaunay triangulation T_t;
- list F_t of freespace tetrahedra such that $F_t \subseteq T_t$;
- list O_t such that $O_t \subseteq F_t$ and $|\delta O_t|$ is manifold;
- list P_{t+1} of new Structure-from-Motion points, camera locations $\mathbf{c}_{t'}$ where $t' \leq t+1$;

and output:

- 3D Delaunay triangulation T_{t+1} by adding P_{t+1} in T_t;
- list F_{t+1} of freespace tetrahedra (by raytracing in T_{t+1}) such that $F_{t+1} \subseteq T_{t+1}$;
- list O_{t+1} such that $|\delta O_{t+1}|$ is manifold and O_{t+1} is the largest as possible in F_{t+1}.

Notations T, O, F are these tetrahedra lists if t does not need to be mentioned. The 3D ball centered at $\mathbf{x} \in \mathbb{R}^3$ with radius $r > 0$ is $B_r(\mathbf{x})$. Note that the vertex set of T_t is $P_0 \cup P_1 \cup \cdots \cup P_t$.

3.3 Assumptions and Consequences

The following assumptions are done for complexity reasons.

First we assume that $\mathbf{p}_i \in P_t$ and $t' \in R_i$ imply $t \in R_i$ and $t' \leq t$. This means that $\mathbf{p}_i$ is reconstructed from the t-th image and previous ones, but it is not reconstructed from next

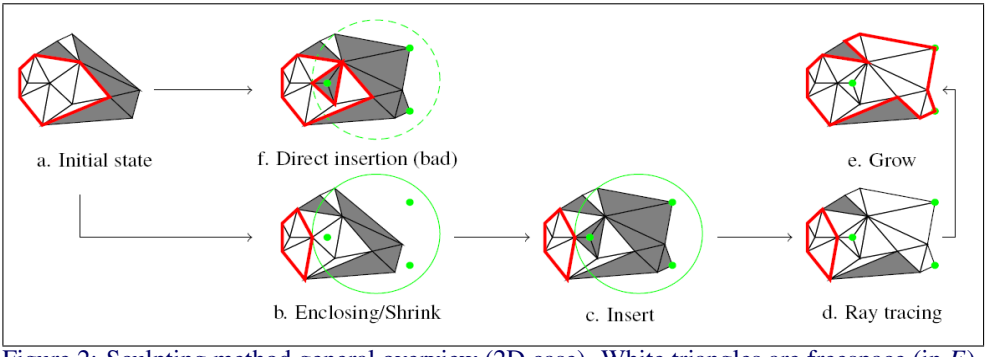

Figure 2: Sculpting method general overview (2D case). White triangles are freespace (in F), grey triangles are matter, red line is border δO, green dots are newly inserted points and the green circle encloses the tetrahedra that will be deleted by the addition of the green dots.

ones. This implies that ray $\mathbf{p}_i\mathbf{c}_t$ exists if $\mathbf{p}_i \in P_t$ and SfM does not update a reconstructed point once it is a vertex of T.

Second we assume that the lengths of all rays $\mathbf{p}_i\mathbf{c}_t$ are bounded by $r > 0$. This is almost a consequence of standard SfM point filtering, since the point uncertainty roughly increases as the square of the ray length [14]. Then P_t is included in $B_r(\mathbf{c}_t)$.

Third we assume that the diameters of the tetrahedra are bounded by $l > 0$. This condition is met if T is initialized by a large cartesian grid of Steiner vertices (*i.e.* extra vertices): the diameter of the circumscribing sphere of a tetrahedron is less than the diagonal length l of the (implicit) grid voxels. Since every tetrahedron created by the addition of a point has this point as vertex [5], the addition of P_{t+1} is a local update of T included in $B_{r+l}(\mathbf{c}_{t+1})$.

4 Our Incremental Sculpting Method

Assume that we add P_{t+1} in the Delaunay T_t as soon as the O_t computation is done. This destroys a list D of tetrahedra, which can contain tetrahedra in O_t (Fig. 2.f). The problem is the following: if we initialize $O_{t+1} = O_t \setminus D$, $|\delta O_{t+1}|$ can be non manifold, and there is no obvious method to update O_{t+1} such that $|\delta O_{t+1}|$ becomes manifold (inspired by [10], we could add new points in the Delaunay, but new tetrahedra will be destroyed and so on).

Here is our idea. List D is computed *without* adding P_{t+1} in T_t. Then we initialize $O_{t+1} = O_t$, shrink O_{t+1} such that $|\delta O_{t+1}|$ is maintained manifold by removing progressively tetrahedra from O_{t+1} until $O_{t+1} \cap D = \emptyset$. Now we add P_{t+1} in T_t without modifying neither O_{t+1} nor its border. If condition $O_{t+1} \cap D = \emptyset$ can not be meet exactly, a minority of points in P_{t+1} are not added in T_t to be sure that $|\delta O_{t+1}|$ is manifold. Last we trace rays to obtain the new freespace F_{t+1}, and grow O_{t+1} in F_{t+1} such that $|\delta O_{t+1}|$ is maintained manifold.

4.1 Step 1: Enclosing

We should calculate D without updating T to ensure that $|\delta O_{t+1}|$ is always a manifold. In practice, the results are better if we replace D by E such that $D \subseteq E \subseteq T_t$, then shrinking will be stopped when $O_{t+1} \cap E = \emptyset$ (which implies what we would like: $O_{t+1} \cap D = \emptyset$). We chose E such that it is the list of the tetrahedra in T_t included in a ball B including $|D|$ (Fig. 2.b). According to Sec. 3.3, we can use $B = B_{r+l}(\mathbf{c}_{t+1})$.

4.2 Step 2: O Shrinking

Now we would like to find O_{t+1} such that $O_{t+1} \subseteq O_t \setminus E$ (*i.e.* $O_{t+1} \subseteq O_t$ and $O_{t+1} \cap E = \emptyset$) and $|\delta O_{t+1}|$ is manifold. We initialize $O_{t+1} = O_t$ (Fig. 2.a) and we remove from O_{t+1} tetrahedra one-at-once (using the subtraction test) or several-at-once (using the general test). During this shrinking process, $|\delta O_{t+1}|$ is manifold and $O_{t+1} \subseteq F_t$ (Fig. 2.b). The selection of the removed tetrahedra and the stopping criteria of this process are detailed in Sec. 4.7. Actually, we can not prove that the resulting O_{t+1} meets exactly $O_{t+1} \cap D = \emptyset$, but we expect that the number of tetrahedra in $O_{t+1} \cap D$ is very small and the next step deals with this problem.

4.3 Step 3: Adding Points without Destroying δO

We initialize $T_{t+1} = T_t$ and $F_{t+1} = F_t$. For every point $\mathbf{p}$ in P_{t+1}, we use the following process.

Thanks to CGAL [3], we calculate the list $D(\mathbf{p})$ of tetrahedra of T_{t+1} which would be destroyed if we add $\mathbf{p}$ in T_{t+1}.

If $D(\mathbf{p}) \cap O_{t+1} = \emptyset$, we add $\mathbf{p}$ in T_{t+1} (Fig. 2.c). This does not update O_{t+1}, thus $|\delta O_{t+1}|$ is still a manifold. We also apply $F_{t+1} \leftarrow F_{t+1} \setminus D(\mathbf{p})$ *i.e.* the tetrahedra created by the addition of $\mathbf{p}$ are labelled matter. We still have $O_{t+1} \subseteq F_{t+1}$. The creation date $t+1$ is also given to the created tetrahedra.

If $D(\mathbf{p}) \cap O_{t+1} \neq \emptyset$, it is difficult to update O_{t+1} such that $|\delta O_{t+1}|$ is still manifold and T_{t+1} is still a 3D Delaunay triangulation. We do not add $\mathbf{p}$ in this case, which is rare since $O_{t+1} \cap D$ is expected to be very small (Sec. 4.2). We also apply $P_{t+1} \leftarrow P_{t+1} \setminus \{\mathbf{p}\}$.

4.4 Step 4: Ray-Tracing

Ray tracing labels tetrahedra in T_{t+1} as freespace, *i.e.* it increases list F_{t+1}. We do not trace all rays available at time $t+1$ for complexity reason, so the rays are selected before tracing.

First, we trace the rays of P_{t+1}, *i.e.* every ray $\mathbf{p}_i\mathbf{c}_j, j \in R_i$ of every point $\mathbf{p}_i$ in P_{t+1}, since this computation was not done before. Tracing ray $\mathbf{p}_i\mathbf{c}_j$ is a walk in the adjacency graph of the tetrahedra: the walk starts from $\mathbf{p}_i$ (which is a vertex of T), and we go from a tetrahedron to one of its four neighboring tetrahedra if the common (face) triangle is intersected by the ray. Every tetrahedron has an intersection counter which is incremented if it is in the walk.

Second, we collect in list R every ray of $P_{t'}$ (where $t' \leq t$) which *can* intersect a tetrahedron of N. List N is the list of the new tetrahedra, *i.e.* the tetrahedra which have creation date $t+1$ (Sec. 4.3). The other rays do not need to be considered. Then we trace every ray in R as explained before. Since this ray was already traced, we only increment the intersection counter for the tetrahedra in N (every ray is counted once). At the end, $O_{t+1} \subseteq F_{t+1}$ since the ray-tracing step adds new tetrahedra to the list F_{t+1} (Fig. 2.d).

Here is a method to compute R efficiently. Let X_N be a bounding box of the tetrahedra in N. Let $X_{t'}$ be a bounding box of the rays of $P_{t'}$. For every t' such that $t' \leq t$ and $X_N \cap X_{t'} \neq \emptyset$, we put in R the rays of $P_{t'}$ which intersect X_N.

4.5 Step 5: O Growing

We grow O_{t+1} in freespace F_{t+1} by adding tetrahedra one-at-once (addition test) and several-at-once (general test). As required in Sec. 3.2, $O_{t+1} \subseteq F_{t+1}$ and $|\delta O_{t+1}|$ is manifold (Fig. 2.e). This step is similar to that of the batch method in [20]. Here we give an overview of it.

First we apply a "One-Tetrahedron-at-Once Growing". A priority queue Q stores the tetrahedra in $F_{t+1} \setminus O_{t+1}$ which have a triangle in δO_{t+1} (we initialize Q with a tetrahedron in $F_{t+1} \cap E$). At each step, Q provides tetrahedron Δ with the largest ray intersection counter. We try to add Δ to O_{t+1} using the addition test. If this is successful, the tetrahedra in $F_{t+1} \setminus O_{t+1}$ which are adjacent to Δ are added to Q. The process stops when Q is empty. This growing is fast thanks to the addition test, but it can not change the $|\delta O|$ genus.

Second we apply a "Several-Tetrahedra-at-Once Growing" to allow genus changes. We find a vertex $\mathbf{v}$ in $|\delta O_{t+1}| \cap |E|$ such that all $\mathbf{v}$-incident tetrahedra are in F_{t+1}, and try to add to O_{t+1} those tetrahedra which are in $F_{t+1} \setminus O_{t+1}$ using the general test. If this is successful, we try to start one-tetrahedron-at once growings from these tetrahedra. The overall process stops when we can not find a successful $\mathbf{v}$.

4.6 Step 6: Post-Processing

First of all, we apply a genus refinement procedure ("Handle Removal") to $O_{t+1} \cap E$ as described in [21]. The only difference is that we don't add Steiner points in the middle of the edges such that T_{t+1} is Delaunay. Then we apply an incremental surface smoothing step [20].

4.7 Details on the Shrinking Step

We initialize $O_{t+1} = O_t$ and progressively remove tetrahedra in O_{t+1} such that $|\delta O_{t+1}|$ remains manifold until $O_{t+1} \cap E = \emptyset$. The O_{t+1} shrinking is an inverse of growing in Sec. 4.5. Let Q be the list (priority queue) of tetrahedra in $O_{t+1} \cap E$ which have a triangle in δO_{t+1}.

First we apply a "One-Tetrahedron-at-Once Shrinking". We remove from Q the tetrahedron Δ which has the smallest intersection counter. If $\Delta \notin O_{t+1}$ or Δ does not have a triangle in δO_{t+1}, we take another Δ in Q. Then we try to remove Δ from O_{t+1} such that $|\delta O_{t+1}|$ remains manifold using the subtraction test (Sec. 3.1). In case of success, we add to Q the tetrahedra of $O_{t+1} \cap E$ which are adjacent to Δ. We continue until $Q = \emptyset$. This shrinking is fast thanks to the subtraction test, but it can provide $E \cap O_{t+1} \neq \emptyset$ (*e.g.* if $E \cap O_{t+1} = \emptyset$ implies that $|\delta O_{t+1}|$ genus changes).

Second we apply a "Several-Tetrahedra-at-Once Shrinking" to allow genus changes. We find a vertex $\mathbf{v}$ which is both in a triangle of δO_{t+1} and in a tetrahedron of E, define L as the list of tetrahedra in $O_{t+1} \cap E$ which are incident to $\mathbf{v}$, apply $O_{t+1} \leftarrow O_{t+1} \setminus L$, and apply the general test for δO_{t+1} at every vertex of L. In case of success, we redefine a list Q with the adjacent tetrahedra of L, and redo the one-tetrahedron-at-once shrinking above. In case of failure, we apply $O_{t+1} \leftarrow O_{t+1} \cup L$ and try another $\mathbf{v}$. The overall process stops when we can not find a successful $\mathbf{v}$.

5 Experiments

5.1 Real Image Sequence

The City sequence is taken by a PointGrey Ladybug omnidirectional camera. Ladybug is a (non-central calibrated) rigid multicamera system consisting of six synchronized pinhole cameras each of which takes 1024×768 images at 15 fps. It is mounted on a car and is about 4 meters above the ground. The trajectory is 2.5 km long and includes a large loop (2.3 km).

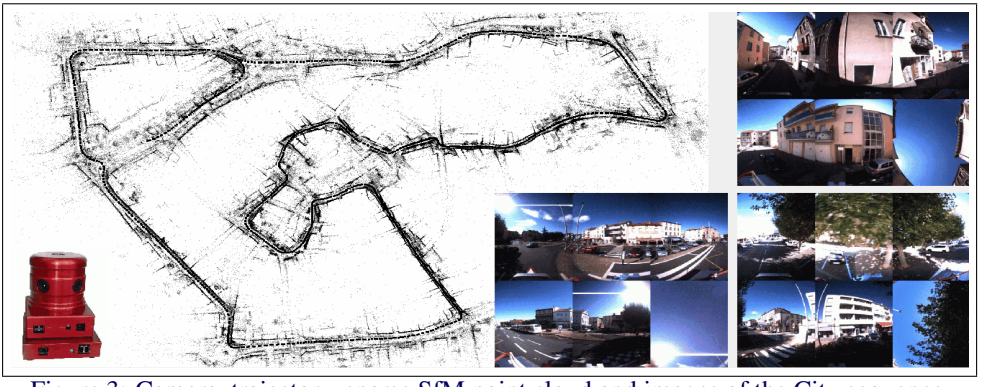

Figure 3: Camera, trajectory, sparse SfM point cloud and images of the City sequence.

First we provide details on SfM. A central approximation simplifies the calculation; we enforce the centers of the 6 cameras to be the same. Moreover, the size of the input images are reduced by a half to improve the computation speed. Then we adapt the incremental SfM in [17] based on local bundle adjustment. The loop is closed thanks to a global bundle adjustment. At the end of the SfM process, 1306 keyframes are selected in the 7735 available 6-tuples of images, 483k points are reconstructed from 2.64M Harris points detected and matched in the images. Thus the surface should be reconstructed with a small number of points (about 193 per meter). Fig. 3 shows a top view of the SfM result and sequence images.

Second we explain how to initialize our incremental method. Let t be the keyframe index (same notation as in Sec. 3 and 4). We construct the 3D Delaunay triangulation T_{40} from the points in P_t where $t \leq 40$, estimate freespace F_{40} by ray-tracing every ray in these P_t (Sec. 4.4), and obtain O_{40} by the growing step (Sec. 4.5) and handle removal [21]. The points of P_t are filtered as follows. Let $\mathbf{v}$ be the vertical direction ($||\mathbf{v}|| = 1$) and $s = \sum_{t=0}^{39} ||\mathbf{c}_{t+1} - \mathbf{c}_t||/39$. Every point $\mathbf{p}_i$ in P_t meets $-3s \leq (\mathbf{p}_i - \mathbf{c}_t).\mathbf{v}$ (we remove points below the ground surface), $||\mathbf{p}_i - \mathbf{c}_t|| \leq 15s$ (Sec. 3.3) and $\mathbf{p}_i$ has two rays forming an angle larger than 10° (standard filtering using aperture angle). The step of the grid of Steiner points is $15s$.

Third we apply the incremental method for $40 \leq t \leq 1306$. Fig. 4 shows the final surface as well as local views of the surface at several times t. The final surface has 528k triangles. The joint video shows the progressive surface reconstruction and the final surface (this video is also in http://www.youtube.com/watch?v=w1AQfvhGx5I).

The execution time for every keyframe is on the left of Fig. 5. We use a 4xIntel Xeon W3530 at 2.8 GHz (multi-threading is only used by the ray-tracing step). The most costly step is the handle removal (Sec. 4.6). This step and the (main part of the) other steps are only applied in E. Remind that E is the list of the tetrahedra in the enclosing ball B, which has a bounded radius (Sec. 4.1). According to Fig. 5 (right), the size of E is less than $120k$, which is quite smaller than $2066k$, the number of tetrahedra in the final 3D Delaunay triangulation.

The large loop (2.3 km) is closed at the very end of the sequence, where we observe a computation time increase. This is due to point increase in B: we reconstruct new points of the loop end at a location where points are already reconstructed at the loop beginning.

According to Sec. 4.2, we can not prove that O_{t+1} meets $O_{t+1} \cap D = \emptyset$ and we expect that $O_{t+1} \cap D$ is small enough to reject a minority of points (Sec. 4.3). In our experiments, $O_{t+1} \cap D = \emptyset$ is not met in only 59 keyframes out of 1306 (4.5%). For these keyframes, the maximum percentage of rejected points is 26.6% and 56 keyframes have less than 13.3% of rejected points. Then a minor amount of points can not be added in T_{t+1} due to $O_{t+1} \cap D \neq \emptyset$.

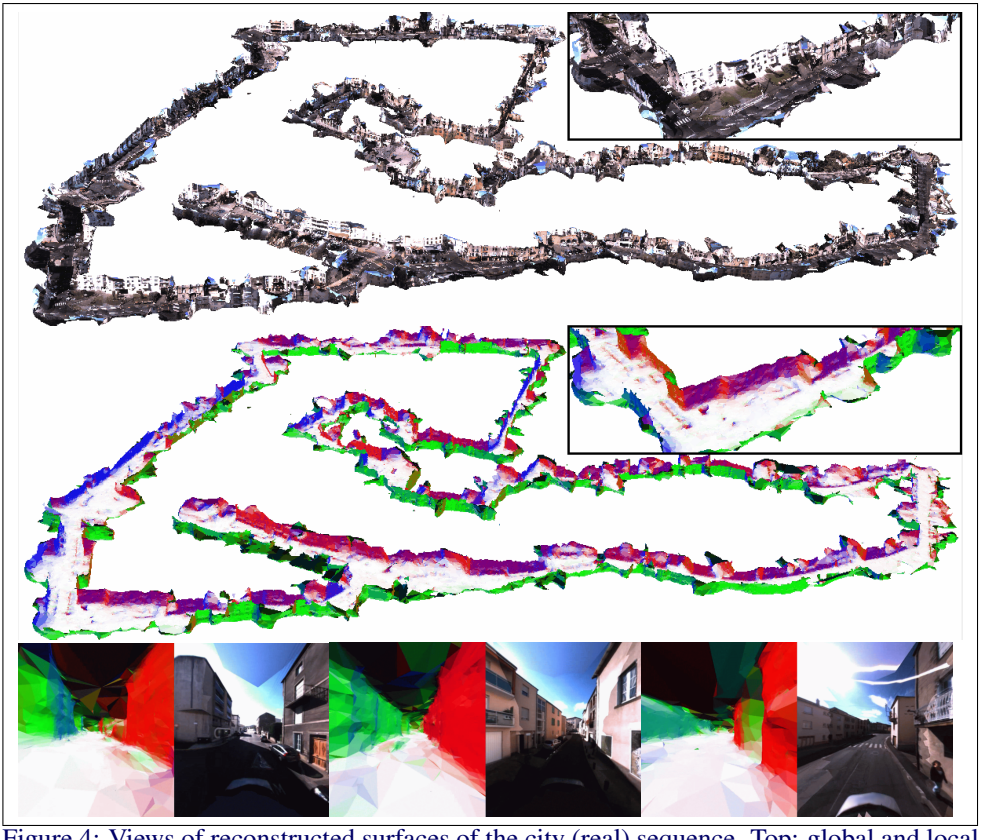

Figure 4: Views of reconstructed surfaces of the city (real) sequence. Top: global and local views of the final surface (the sky is removed to help visualization). Bottom: local views of the surface during computation. The triangle normals are colored: the ground is white, the walls are red-green-blue, the sky is black.

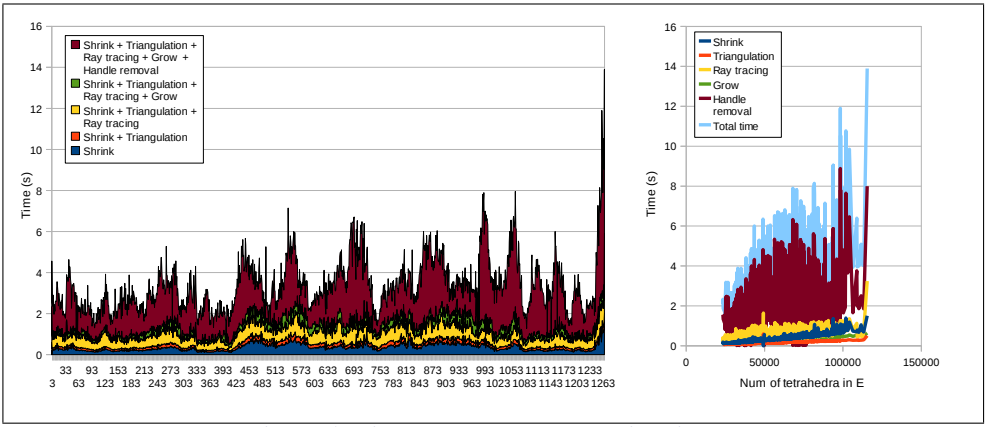

Figure 5: City sequence reconstruction times.

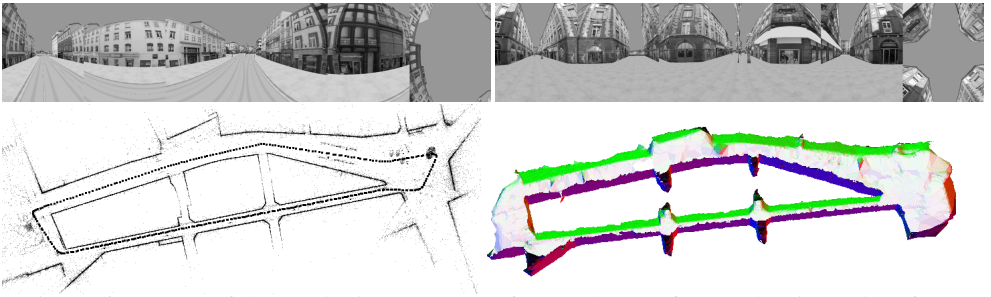

Figure 6: Images of the synthetic sequence, SfM results, the final and estimated surface.

5.2 Synthetic Image Sequence with Ground Truth

This Section provides the reconstruction error of our sparse scheme (central SfM + incremental surface reconstruction) applied in a synthetic urban scene, whose textures are extracted from real images taken in a city. The synthetic sequence has 1553 images generated by ray-tracing and the non-central camera in Sec. 5.1. The camera trajectory is a 621 m long and closed loop. The SfM reconstructs 346 keyframes and 145k points from 775k Harris points detected and matched in the images. Then we apply our incremental surface reconstruction and estimate the error between the final estimated surface (182k triangles) and the ground truth surface.

Now we explain how to estimate the reconstruction error. First, a pose-based registration is used to set both surfaces in the same coordinate system. The i-th keyframe has estimated location $\mathbf{c}_i$ and ground truth location $\mathbf{c}_i^g$; the latter is the mean of the 6 camera centers in the ground truth coordinate system. Then we estimate the similarity transformation S minimizing $E(S) = \sum_{i=0}^{345} ||S(\mathbf{c}_i) - \mathbf{c}_i^g||^2$, and use S to map the estimated surface in the ground truth coordinate system. Second, a ray-tracing approach is used to compute the error. Let $\mathbf{p}$ be a pixel in the i-th keyframe. Let $\mathbf{p}_e$ be the first intersection of the estimated surface and the ray defined by $\mathbf{p}$ and the estimated i-th pose. Let $\mathbf{p}_g$ be the first intersection of the ground truth surface and the ray defined by $\mathbf{p}$ and the ground truth pose of the i-th keyframe. If both $\mathbf{p}_e$ and $\mathbf{p}_g$ exist, we define error $e(\mathbf{p}) = ||\mathbf{p}_e - \mathbf{p}_g||$.

The pose-registration provides $\sqrt{E(S)/346} = 14$ cm. Then we uniformly sample pixels in the keyframe sequence and examine the distribution of $e(\mathbf{p})$ for 769k pixels. The x-quantile q_x is the real such that x percents of the e values are less than q_x. We have $q_{10} = 12$, $q_{20} = 16$, $q_{30} = 18$, $q_{40} = 22$, $q_{50} = 28$, $q_{60} = 48$, $q_{70} = 76$, $q_{80} = 112$, $q_{90} = 174$, and 3% of the e values are larger than 600 (all numbers are given in centimeters).

6 Conclusion

In this paper, we have introduced a new incremental 2-manifold surface reconstruction method taking a sparse SfM data as input. Compared to the previous similar algorithms, it has the advantage to produce a manifold without genus restrictions and without prohibitive complexity in case of large loops in the input trajectory. It has been experimented on a large scale real urban scene as well as on a set of synthetic data with ground truth.

Several improvements of our method are subject of future work. First, the choice of a better (smaller) enclosing area where almost all calculations are done would decrease the computation time. Second, our implementation and matching method should be improved. Moreover, several lines of investigation exists to enhance the quality of the reconstructed

surface. In particular, the usage of the other type of the input primitives than the interest points should be explored. We could also think about surface denoising methods that make a better usage of the Structure-from-Motion properties and scene priors.

Acknowledgements

Thanks to CNRS, UBP, FEDER and Auvergne Région for funding. Thanks to the CRISTAL project for providing the ground truth VRML model used in the quantitative evaluation.

References

[1] J.D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Transaction on Graphics*, 3(4):266–286, 1984.

[2] M. Botsch, L. Kobbelt, M. Pauly, and P. Alliez an B. Levy. *Polygon mesh processing*. AK Peters, 2010.

[3] CGAL. Computational geometry algorithm library. www.cgal.org.

[4] C. Hernandez Esteban and F. Schmitt. Silhouette and stereo fusion 3D object modeling. *Computer Vision and Image Understanding*, 96(3):367–392, 2004.

[5] O. Faugeras, E. Le Bras Mehlman, and J.D. Boissonnat. Representing stereo data with the Delaunay triangulation. *Artificial Intelligence*, pages 41–47, 1990.

[6] L. De Floriani, P. Magillo, and E. Puppo. Managing the level of detail in 3D shape reconstruction and presentation. In *Proc. ICPR*, pages 389–391, 1988.

[7] A. Gezahegne. Surface reconstruction with constrained sculpting, 2005. Master of Science Thesis, University of California Davis.

[8] P. Giblin. *Graphs, surfaces and homology*. Cambridge University Press, Third Edition., 2010.

[9] J. Goodman and J.O. Rourke (editors). *Handbook of discrete and computational geometry*. Second Edition, CRC Press., 2004.

[10] A. Gueziec, G. Taubin, F. Lazarus, and B. Horn. Cutting and stitching: converting sets of polygons to manifold surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 7(2):136–151, 2001.

[11] A. Hilton. Scene modelling from sparse 3D data. *Image and Vision Computing*, 23: 900–920, 2005.

[12] C.M. Hoffmann. *Geometric and solid modeling: an introduction*. Morgan Kaufmann Publishers Inc., 1989.

[13] P. Labatut, J.P. Pons, and R. Keriven. Efficient multiview reconstruction of large-scale scenes using interest points, Delaunay triangulation, and graph cut. In *Proc. ICCV*, 2007.

[14] M. Lhuillier. A generic error model and its application to automatic 3d modeling of scenes using a catadioptric camera. *International Journal of Computer Vision*, 91(2): 175–199, 2011.

[15] D. Lovi, N. Birkbeck, D. Cobzas, and M. Jagersand. Incremental free-space carving for real-time 3D reconstruction. In *Proc. 3DIMPVT*, 2012.

[16] D.D. Morris and T. Kanade. Image-consistent surface triangulation. In *Proc. CVPR*, pages 332–338, 2000.

[17] E. Mouragnon, M.Lhuillier, M. Dhome, F. Dekeyer, and P. Sayd. Generic and real-time structure from motion. In *Proc. BMVC*, 2007.

[18] Q. Pan, G. Reitmayr, and T. Drummond. ProFORMA: probabilistic feature-based on-line rapid model acquisition. In *Proc. BVMC*, 2009.

[19] C.J. Taylor. Surface reconstruction from feature based stereo. In *Proc. ICCV*, 2003.

[20] S. Yu and M. Lhuillier. Incremental reconstruction of a manifold surface from sparse visual mapping. In *Proc. 3DIMPVT*, 2012.

[21] S. Yu and M. Lhuillier. Genus refinement of a manifold surface reconstructed by sculpting the 3D-Delaunay triangulation of structure-from-motion points. In *Proc. ICPR*, 2012.