Genus refinement of a manifold surface reconstructed by sculpting the 3d-Delaunay triangulation of Structure-from-Motion points

Shuda YU and Maxime LHUILLIER
Institut Pascal, UMR 6602, CNRS/UBP/IFMA, Aubière, France
Maxime.Lhuillier@univ-bpclermont.fr

Abstract

The automatic surface reconstruction from an image sequence is still an active research topic. Recently, a method was designed to reconstruct a 2-manifold surface from the sparse cloud of points generated by Structure-from-Motion (SfM). This method reconstructs outdoor scenes from hundreds of catadioptric images. It is based on sculpting the 3d Delaunay triangulation of the points by ray-tracing. Our paper improves the resulting surface by removing spurious handles, which occur frequently.

1 Introduction

Recently, a method [11] was designed to reconstruct a 2-manifold from the sparse cloud of points generated by SfM. This method is interesting for two reasons. First, a surface directly estimated from the SfM result would be ideal for both time and space complexities since a dense stereo step is not used. Second, a 2-manifold would be useful for initializing a more accurate (but more costly) surface reconstruction such as surface deformation optimizing photo-consistency.

Every point in a 2-manifold (i.e. 2d topological manifold surface) has a surface neighborhood which has the disk topology. The genus of a 2-manifold is the number of its handles, e.g. a sphere has genus 0 and a torus has genus 1. Since our 2-manifold is triangulated, each triangle is exactly connected by its 3 edges to 3 other triangles, the surface has neither hole nor self-intersection, and it cuts $\mathbb{R}^3$ into *free-space* and *matter* regions.

The method in [11] was discussed with others which work with the same input: a sparse cloud P of SfM points. Most of these methods have the following steps. First, the 3d Delaunay triangulation T of P is build [2]: T partitions the convex hull of P, and the circumsphere of every tetrahedron in T does not contain any vertex of T within it. Second, all tetrahedra in T are labeled either *free-space* or *matter* with the help of the rays provided by SfM. A ray is a line segment connecting a 3d point to a viewpoint used to reconstruct the point. A tetrahedron is *free-space* if it is intersected by a ray, otherwise it is *matter*. Third, the labeling is used to generate the surface. Several methods [9, 8, 5] directly consider the target surface as the list of triangles separating the *free-space* and *matter*. Unfortunately, the resulting surface can be non-manifold. In [4] ([11], respectively), a 2-manifold is extracted thanks to a region growing procedure in the list of *matter* (*free-space*, respectively) tetrahedra.

However, spurious handles could exist on the 2-manifold (they are called "arks" in [11]). Fig. 1 shows an example: the oblique handle on the left connects a small wall to the ground. This handle does not exist on the true scene surface and it should be removed. Spurious handles also occur in other contexts [3].

Figure 1. Spurious handle and its removal

Previous methods [10, 12] remove the spurious handles of 3d models. They don't use the visibility data (*free-space* or *matter*) and remove handles assuming that (1) the spurious handles are small and (2) the handles in the real world are large. Here we would like to remove the most visually critical spurious handles (they can be large) and we detect them thanks to the visibility.

The paper summarizes the scene modelling algorithm (Sec. 2), describes our spurious handle removal method (Sec. 3) and gives the experiments (Sec. 4).

2 Automatic Scene Modelling (Summary)

Our scene modelling method has two steps: geometry estimation [6] and surface estimation [11].

Geometry Estimation Our camera model is a central catadioptric camera with a symmetry axis. First, SfM simultaneously estimates the camera poses (rotation+translation) and a sparse cloud of 3d points thanks to approximate values of the intrinsic parameters (radial distortion coefficients). These points are the reconstructions of Harris points detected and matched in the images. Then, all poses, points and intrinsic parameters are refined by an adequate bundle adjustment assuming that the radial distortion coefficients are constant in the whole sequence.

Surface Estimation The principle is in Section 1: 3d Delaunay triangulation, ray-tracing, and 2-manifold extraction. Here we add some details.

In order to generate a 2-manifold, a region-growing approach is applied. The idea is to grow iteratively an *outside* region O in the *free-space* region by maintaining the manifold property on the border of O during the growing [11]. Before the growing, we assign *inside* to all tetrahedra. The final manifold is the list of triangles separating the *outside* tetrahedra and *inside* tetrahedra.

Lastly, post-processing is used to improve the quality of the surface. It includes peak removal, surface denoising using discrete Laplacian, triangle removal in the sky, and surface texturing.

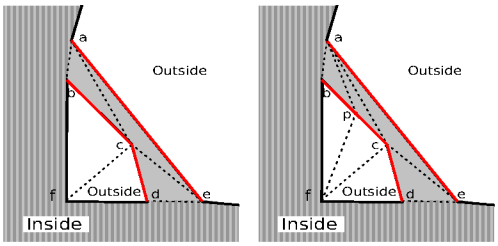
3 Removal of the Spurious Handles

Our contribution is a new handle removal method. First, it detects the visually critical handles in the 2-manifold (Section 3.2). Then it inserts Steiner points in the Delaunay in such a way that the surface is still a 2-manifold (Section 3.3). At last, a local region-growing algorithm is applied for every vertex on the critical handles to add new tetrahedra into the *outside* region (Section 3.4). Section 3.1 summarizes our previous method.

3.1 Previous Method

The end of the experiment Section in [11] also describes a method to remove the spurious handles using Steiner points (called "artificial points" in [11]). The adding of Steiner points (i.e. extra points) is a standard method in Computational Geometry to generate mesh with good properties [1]. The previous handle removal method in [11] inserts randomly a few Steiner points in the neighbourhood of the camera trajectory to avoid long tetrahedra which are potentially in handles. This is done in the 3d Delaunay triangulation step (before ray tracing and region growing). We observed that this method usually works well for sparse SfM point cloud [6] but not for quasi-dense point clouds [7]: many large handles are not removed. Besides, another idea is the adding of a regular grid of vertices in the 3d Delaunay triangulation to bound the size of the tetrahedra. This idea is not a good solution since the Steiner points should be dense enough and it degrades significantly the surface quality.

3.2 Detection of the Critical Handle Edges



(a) *The polygon* **abcde** *forms an handle, and the edges* **bc**, **cd** *and* **ae** *are handle edges.*

(b) *A Steiner point* **p** *is inserted on* **bc**. *Triangle* **bfc** *is split to triangle* **bfp**, **pfc**. *Triangle* **abc** *is split to triangle* **abp**, **pca**

Figure 2. A handle in the 2d case. Bold black and red lines = manifold, red lines = handle edges, dashed lines = triangles of Delaunay, white region = *outside*, grey region = *inside-freespace*, grey and striped region = *inside-matter*.

First we introduce definitions. The tetrahedra have an adjacency graph: the graph vertices are the tetrahedra, the graph edges are the triangles between two tetrahedra. A handle is a connected set of tetrahedra in this graph, which are both *free-space* and *inside*. A handle edge is an edge of a tetrahedron in a handle, such that the edge is on the 2-manifold and all tetrahedra incident to the edge are *free-space*. Fig. 2(a) shows a handle and several handle edges in the 2d case for legibility. In the 2d case, surfaces, tetrahedra and triangles are replaced by curves, triangles and edges, respectively. Let $\mathbf{c}_k$ be the 3d location of the k-th image of the image sequence.

Second, the following method is used to detect the spurious handles which are large enough to be visually critical. Let e_i be a handle edge. Let $\mathbf{v}_l$ and $\mathbf{v}_r$ be both ends of e_i. If there is viewpoint $\mathbf{c}_k$ such that the

angle $\widehat{\mathbf{v}_l\mathbf{c}_k\mathbf{v}_r}$ is greater than a threshold α, we say that e_i is visually critical. By doing so, the list of all critical handle edges e_i are detected. Let T_i be the list of the tetrahedra having e_i as edge. Now we would like to insert in the *outside* region the tetrahedra in T_i.

3.3 Insertion of the Steiner Points

This step adds a Steiner point at the middle of every critical handle edge e_i to change the adjacency graph of the tetrahedra at the spurious handles. Such an "edge splitting" improves the results of our handle removal method. However, an edge splitting modifies the Delaunay triangulation: old tetrahedra are destroyed and new tetrahedra are created. The manifold property of the *outside* region may be lost due to the destruction of tetrahedra and the states (*outside* or *inside*) of the new tetrahedra should be defined.

Our solution is the following: we maintain the manifold property by adding Steiner points without enforcing the empty circumsphere property of the Delaunay triangulation. Fig. 2(b) shows an example. When a point is added on the middle of edge e_i, every tetrahedron in T_i is split into two tetrahedra which share a same triangle. By assigning the same *inside/outside* state of each tetrahedron to both its split tetrahedra, the new triangulated surface between *inside* and *outside* tetrahedra is still a 2-manifold.

3.4 Local Region-Growing

The adding of Steiner points changes the graph of the tetrahedra and provides new tetrahedra for the region-growing process. Here we use a local region-growing approach which is more time consuming than that in [11], but which is more adapted to the removal of spurious handles.

For each vertex $\mathbf{v}_i$ on split critical handle edges, we define the list of tetrahedra L_i incident to $\mathbf{v}_i$ which are *free-space* and *inside*. Then a first local region-growing process is applied for the list L_i. If the local region-growing of L_i fails, a second local region-growing process is tried individually for each tetrahedron of L_i. The pseudo-code is given by Algorithm 1.

Here, the local region-growing process from the seed list S (S can be a list of tetrahedra L_i or only one tetrahedron Δ) tries to grow the *outside* region by the tetrahedra of S and their neighbours. At first, all tetrahedra of S are forced to be *outside* and the number s of singular vertices due to the enforcement is calculated. A surface vertex is singular if it has no surface neighbourhood which is topologically a disk (this can be checked by considering all surface triangles having

```
For each vertex v_i on the (split) handle edges :
    Get the list L_i of tetrahedra incident to v_i
    which are inside and free-space;
    bool b=Local region-growing (L_i);
    if false==b then
        For each tetrahedron Δ ∈ L_i :
            Local region-growing (Δ);
```

Algorithm 1: Spurious handles removal using local region-growing

the vertex [2, 11]). Then at each time, a *free-space* and *inside* tetrahedron is picked from the neighbours of the forced tetrahedra, and it is forced to be *outside*. This enforcement is retained only if it decreases s. The local region-growing stops when no more enforcement can decrease s. If the final s is 0, then the local region growing succeeds. Otherwise it fails, and all forced tetrahedra should be re-labeled to be *inside*. The pseudo-code is given by Algorithm 2. Now we provides two important technical details.

```
Inputs  : seed list S, list O of outside tetrahedra
Outputs: a boolean, list O of outside tetrahedra
// Notation: δO is the border of O (triangle list)
O ← O ∪ S;
Let s be the number of singular vertices of δO;
Create a queue Q of tetrahedra, Q = ∅;
For each tetrahedron Δ in S :
    For each 4-neighbour tetrahedron Δ' of Δ :
        if Δ' is inside and free-space then
            Q ← Q ∪ {Δ'};
while Q ≠ ∅ do
    Pick a tetrahedron Δ out of Q;
    O ← O ∪ {Δ};
    Let s' the number of singular vertices of δO;
    if s' < s then
        S ← S ∪ {Δ};
        s = s';
        For each 4-neighbor tetrahed. Δ' of Δ :
            if Δ' is inside and free-space then
                Q ← Q ∪ {Δ'};
    else
        O ← O \ {Δ};
    if 0 == s then
        return true;
O ← O \ S; // failure since 0 < s
return false;
```

Algorithm 2: Local region-growing

In this algorithm, the most time consuming part is

the calculation of the number s (or s') of singular vertices in δO; O is the list of outside tetrahedra and δO is its border (i.e. the list of triangles between *outside* and *inside* tetrahedra). At first glance, s could be computed by checking if every vertex of δO is singular after every O update. However, this is very inefficient. Since O is incrementally updated by a very small number of tetrahedra, the neighborhood of a very small number of vertices of δO (those of the added tetrahedra) are modified. So we compute the number of singular vertices of the added tetrahedra before and after the O update. We obtain two numbers s_{before} and s_{after} and we update s by $s = s + s_{after} - s_{before}$. At the beginning, $s = 0$ since δO is a 2-manifold.

Lastly, the algorithm involves a priority queue (heap) Q of *free-space* tetrahedra. The tetrahedra are sorted and selected by using the same priority score as in [11]: the number of rays which intersects the tetrahedra.

4 Experiment

Experiments are done with a sequence of 208 images taken by a catadioptric camera around a church (three images are shown in Fig. 3). The catadioptric camera is a convex mirror mounted in front of a perspective camera. The calculations are done using an Intel(R) Core(TM)2 Duo E8500 @ 3.16GHz. We first give a sum up of the results of the automatic 3d modelling algorithm without handle removal, then the results of the handle removal algorithm are presented.

Two 3d models are reconstructed. The first one, called ChurchS, is obtained by applying the geometry and surface estimation algorithms described in Section 2. The second one, called ChurchQ, is obtained from the quasi-dense point cloud obtained by [7] (the SfM step is the same, the points are the vertices of the final list of triangles after local model selection and redundancy removal) and then the surface estimation algorithm in Section 2 is applied. For ChurchS (resp. ChurchQ), $76k$ (resp. $339k$) 3d points are reconstructed and shown in Fig. 3. Then $59k$ (resp. $316k$) points are selected in the Delaunay by the angle filter method in [11] (using $\epsilon = 5$). Lastly, $190k$ (resp. $1132k$) tetrahedra are *free-space* and 86% (resp. 88%) of the *free-space* tetrahedra are *outside*. The 2-manifold has $94k$ (resp. $472k$) triangles.

Now we apply our handle removal method on both models. First, 1277 (resp. 1306) critical handle edges are detected for ChurchS (resp. ChurchQ) using $\alpha = 5$. For each edge, a Steiner point is added in the middle of the edge. Then our local region-growing is applied on every vertex of the new handle edges. Finally, 9521 (resp. 56628) tetrahedra are added to the *outside* region. This increases the *outside/free-space* ratio from 86% (resp. 88%) to 91% (resp. 93%). This ratio is a quality measure of our algorithm, which is 100% in the ideal case. Fig. 4 shows the results without handle removal, the results of the handle removal in [11] and the results with our handle removal algorithm. We see that our handle removal method removes most of the critical parts of the scene (which are not physically plausible), and the resulting surface is better than the 2-manifold without handle removal and the results of handle removal in [11]. Global views of the results with our handle removal algorithm are shown in Fig. 5. The computation time for ChurchS (resp. ChurchQ) is about 20s (resp. 97s), including 3s (resp. 8.5s) as our handle removal time.

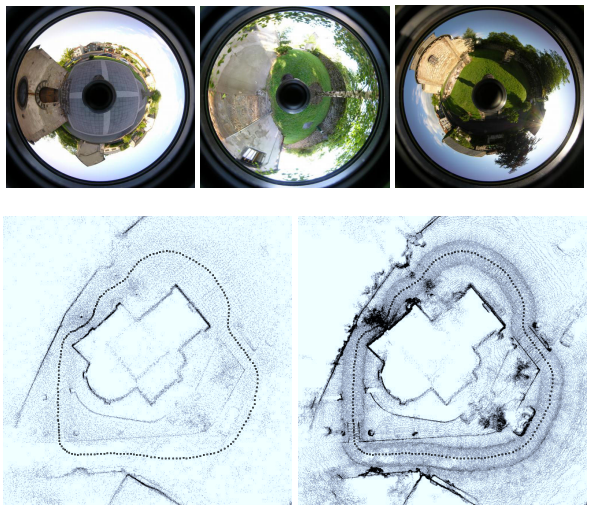


Figure 3. Three images of the sequence (top), sparse (bottom-left) and quasi-dense (bottom-right) clouds of points.

In other experiments, our handle removal method is tested without adding Steiner points to check if these points are useful. We observe that few tetrahedra are added to the *outside* regions so that many spurious handles are not removed. Thus we can see that adding Steiner points is not optional to change the adjacency graph of the tetrahedra at the spurious handles.

Last, our results depend on α: the smaller α is, the more handles are detected, but the higher the processing time is. In our experiments, we find that if a too small α (< 5 degrees) is used, the surface quality is not significantly improved.

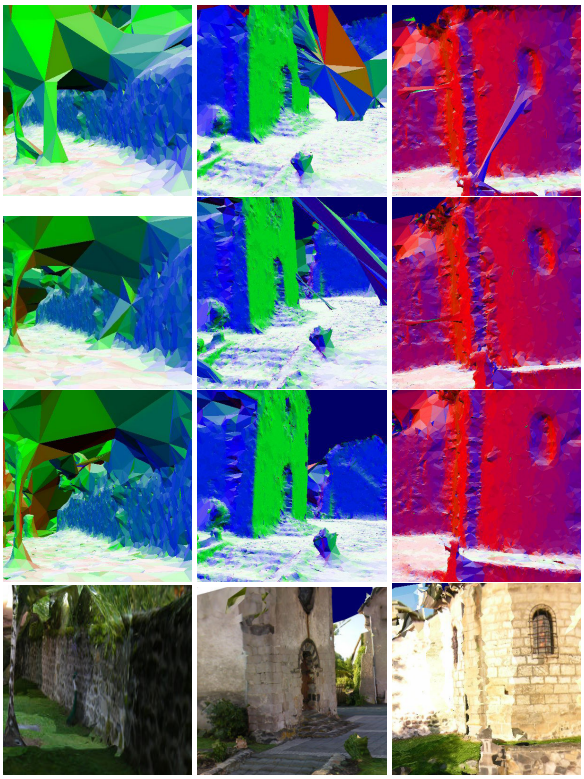


Figure 4. Column 1: ChurchS surface. Columns 2 and 3: ChurchQ surfaces. Row 1: no handle removal. Row 2: the handle removal in [11]. Rows 3 and 4: the handle removal of the paper. In rows 1-3, colours encode the triangle normals.

5 Conclusion

The paper summarizes an automatic scene modelling system from a sparse SfM point cloud, which generates a 2-manifold surface by the *outside* region growing in the *free-space*. We explain a drawback of this method (spurious handles), introduce a post-processing to remove these handles, and compare favourably the results with a previous handle removal method. The idea is based on the use of Steiner points (which do not destroy the 2-manifold) and several local region-growing tried in the spurious handles.

Several improvements could be done in future work. The image contours could be reconstructed and integrated in the Delaunay to improve the surface. Besides, the surface smoothing algorithm could be improved to reduce better the noise of the surface.

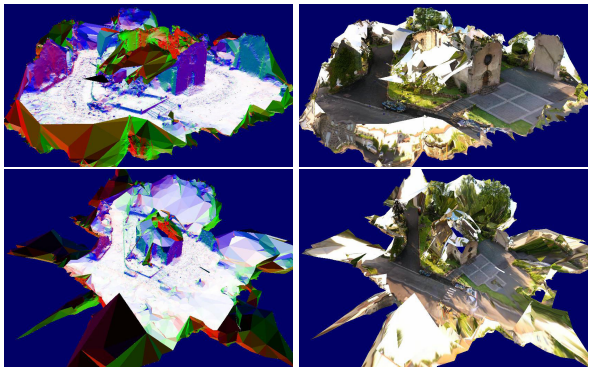


Figure 5. Global views of ChurchQ (top) and ChurchS (bottom).

References

[1] M. Bern, D. Eppstein, and J. Gilbert. Provably good mesh generation. *Journal of Computer and System Sciences*, 48(3):384–409, 1994.

[2] J.-D. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Trans. Graph.*, 3(4):266–286, 1984.

[3] R. Chaine. A geometric convection approach of 3-d reconstruction. *Eurographics Symposium on Geometry Processing*, pages 218–229, 2003.

[4] O. Faugeras, E. Bras-Mehlman, and J. Boissonnat. Representing stereo data with the delaunay triangulation. *Artificial Intelligence*, 44(12):41 – 87, 1990.

[5] P. Labatut, J.-P. Pons, and R. Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. *International Conference on Computer Vision*, 2007.

[6] M. Lhuillier. Automatic scene structure and camera motion using a catadioptric system. *Computer Vision and Image Understanding*, 109(2):186–203, 2008.

[7] M. Lhuillier. A generic error model and its application to automatic 3d modeling of scenes using a catadioptric camera. *International Journal of Computer Vision*, 28(8):1335–1340, 2011.

[8] D. Lovi, N. Birkbeck, D. Cobzas, and M. Jagersand. Incremental free-space carving for real-time 3d reconstruction. In *3DPVT'2010*.

[9] Q. Pan, G. Reitmayr, and T. Drummond. Proforma: Probabilistic feature-based on-line rapid model acquisition. *British Machine Vision Conference*, 2007.

[10] Z. Wood, H. Hoppe, M. Desbrun, and P. Schröder. Removing excess topology from isosurfaces. *ACM Transactions on Graphics (TOG)*, 23(2):190–208, 2004.

[11] S. Yu and M. Lhuillier. Surface reconstruction of scenes using a catadiopric camera. In *Lecture notes in computer sciences 6930, also in Mirage'11*.

[12] Q. Zhou, T. Ju, and S. Hu. Topology repair of solid models using skeletons. *Visualization and Computer Graphics, IEEE Transactions on*, 13(4):675–685, 2007.