ON THE WAY TO WATER-TIGHT MESH

Rui Liu, Darius Burschka, Gerd Hirzinger

Institute of Robotics and Mechatronics, German Aerospace Center (DLR)
Oberpfaffenhofen, 82234 Wessling, Germany.
Rui.Liu@dlr.de

KEY WORDS: Modelling, Laser scanning, Cultural Heritage, Robotics, Automation, Segmentation, Reconstruction, Algorithms

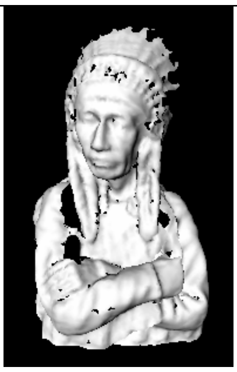
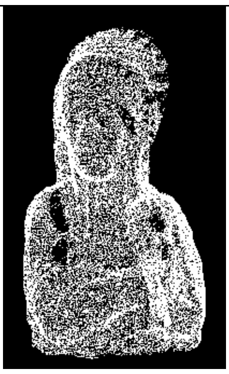
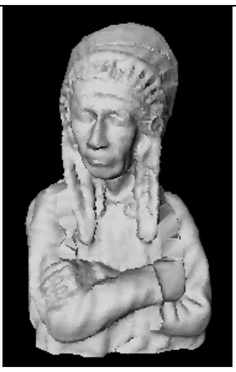
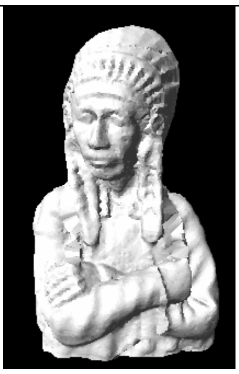
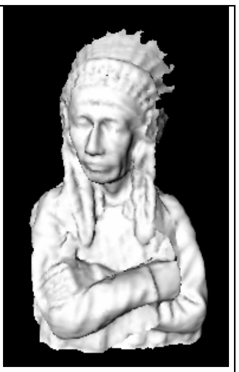
				
Original Mesh	Original Pointcloud	Tight Cocone	Power Crust	Our Method

Figure 1. Comparison between different methods for water-tight mesh generation

ABSTRACT:

This paper presents a new approach to generate water-tight surface mesh. In comparison with the two famous water-tight mesh generation algorithms: Tight Cocone (Dey and Goswami, 2003) and Power Crust (Amenta, Choi and Kolluri, 2001), our method shows a significant quality on the resulted models due to the efficient pre- and post- processing. We choose our previously published online mesh generation method, for it reduces the very large point-clouds very fast and effective, which is suitable for our applications: reconstruction of historical buildings and online digitalisation of small scale objects. To optimisation the resulted mesh, we apply a fast and automatic pre- and post-processing to fill the holes effectively. It consists of six steps: mesh segmentation, filtering of noise, hole detection, filling of holes, triangles subdivision and surface smoothing. The whole process is integrated in one and is performed fully automatically . Validation of our method is based on the digitalisation of small objects with a hand-guided multi-sensory device and 3D-reconstruction of cultural heritage.

1. INTRODUCTION

Automatic generation of a water-tight surface mesh from 3D point clouds is always a beautiful dream for 3D reconstruction. We have tested all of the available water-tight surface meshing software, a few visible holes are always existing. We can not say, that our resulted mesh is “strict” water-tight. Like the other people, we are on the way to water-tight mesh.

Generally, an original generated mesh has holes due to lack of information or because of the special process-requirement, e.g. online-visualization parallel to scanning process, etc. The hand-guided multi-sensor device of our institute performs a online surface reconstruction from the 3D point-cloud. And the resulted mesh has noises and holes. It needs an optimisation. Then, we apply an automatic process, which combines filtering of noise, detection and filling of holes, subdivision and smoothing in one program. And we used the same software for the reconstruction of cultural heritage. It shows a good quality on the resulted mesh. Since our method for mesh generation (Bodenmüller and Hirzinger, 2004) is already published. This paper focuses on the methods for mesh optimisation.

The main contributions of this paper are:

– Firstly, it describes how to get a ***qualified*** water-tight surface mesh through some effective pre- and post-processes;
– Secondly, it introduces an algorithms to fill ***big and complex holes*** in 3D space. There are normally over 1000 vertices at the boundary of one of such big holes.

We have applied our method both on the 3D-reconstruction of historical buildings and on the modelling of objects at the small-scale, which were captured with our hand-guided laser-scanner. The processing is very fast and the resulted meshes show a distinguished quality.

Our paper is organized as follows. In section 2, we summarize and the existing researches in this area and give a brief comparison of them. Then, section 3 describes the main method of our approach, and the related experimental results are shown too. At the end, the possible improvements in the future are addressed in the conclusion section.

2. RELATED WORK

Surface reconstruction from scattered data was firstly addressed by Uselton (Uselton, 1983) at the beginning of 1980's. As one of the early researchers, Boissonnat presented a sculpting of the Delaunay triangulation for reconstruction (Boissonnat, 1984). Edelsbrunner and Mücke introduced a three-dimensional alpha shapes algorithm for a refined sculpting strategy (Edelsbrunner and Mücke, 1994). Huppe et al. approximated a signed distance function induced by the input points and constructed the output surface as a polygon approximation of the zero-set of this function (Huppe et al., 1992). Curless and Levoy represented the signed distance function on voxel grids for surface reconstruction from multiple range scans (Curless and Levoy, 1996). Bernardini et al. designed a ball-pivoting algorithm, which avoids the computation of Voronoi diagram (Bernardini et al.). Gopi, Krishnan and Silva proposed to project the sample points with their neighbours on a plane and then lift the local two dimensional Delaunay triangulations to reconstruct the surface (Gopi, Krishnan and Silva, 2002).

The two important algorithms to generate water-tight surface mesh are power crust and tight cocone. The power crust algorithm (Amenta, Choi and Kolluri, 2001) produces directly a watertight mesh without a post-processing. The main approach is to approximate the medial axis transform (MAT) of a 3D object firstly, and then use an inverse transform of MAT to produce the watertight boundary of a 3D polyhedral solid. Another algorithm to generate water-tight mesh is tight cocone (Dey and Goswami, 2003), which was designed on the basis of the above crust algorithm. It reduces the runtime complexity of it, but needs a post-process to fill the holes. Schall O. and Samozino M. have given a comprehensive analysis between them and shown the related experiment-results (Schall O. and Samozino M., 2005).

3. MAIN METHOD

Since our mesh generation algorithm is already published, we concentrate on the mesh optimisation in this paper. In fact, an arbitrary triangle mesh can be taken as input for our algorithm, and the output is a "water-tight" surface mesh.

Because there are always noises in the original mesh, e.g. small patches, none-manifold edges etc. To achieve an effective hole-filling, we firstly applied an pre-process to filter them out. And after the hole-filling, we subdivide the filled triangles to average size of the original mesh and smooth the filled regions for a better mesh quality. The whole processing can be summarized as:

- Pre-processing: segmentation, filtering and hole-detection;
- Filling of holes;
- Post-processing: subdivision and smoothing

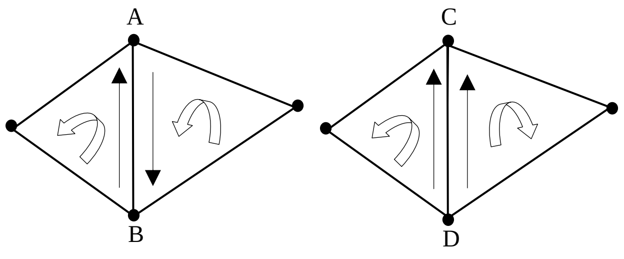
3.1 Segmentation

Since the noises produced during the 3D-points acquisition are always presented as small point-clouds around the main objects, they will be triangulated as small mesh-pieces isolated from the main object or adjacent to it only through one point. To detect these small pieces, we apply an modified BFS (Breath First Search) to identify the connected components. Through searching neighbourhood by edges, we can isolate the part, which is adjacent to the main mesh only by one point.

3.2 Filtering

In this section, small components will be treated as noises and deleted. The threshold can be set as the number of vertices or triangles in one component. However, we think that a better criterion is the sum of the triangles-areas.

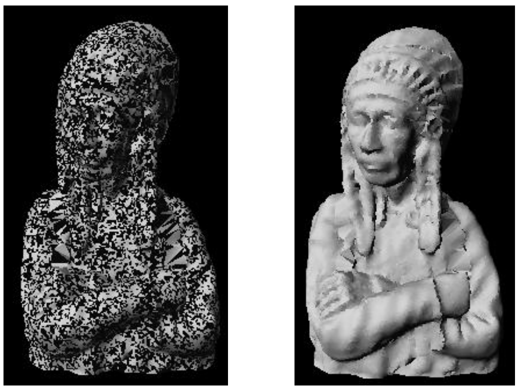
Besides missing of triangles, another reason of the visible holes in triangles mesh is the false triangulation direction. With the option of back-face culling by rendering, these triangles will be displayed as holes. Therefore we should find and delete them. As the common ordering of the triangle in mesh is counter-clockwise, an simple method is to find the edge with the same start- and endpoints in its two neighboured triangles. As illustrated in Figure 2, edge AB is legal and edge CD is illegal.



a) Edge AB is legal   b) Edge CD is illegal

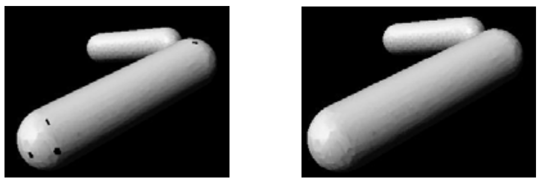
Figure 2. Detection of illegal edges

According to the test of "Tight-Cocone" and "Power Crust", both of them have some false triangulated polygons in the resulted mesh. If we set "two-side lighting off" and "back-face culling on" as the rendering options, there are some visible holes in the generated "water-tight" models (see Figure 3.).



I. two-side lighting off

a) Tight-Cocone generated model: Indian



I. back-face culling on   II. back-face culling off

b) Power Crust generated model: hotdog

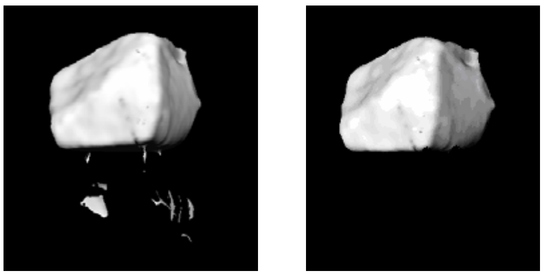
Figure 3. Direction-errors of Tight-Cocone and Power Crust

Fore the more, to maintain the manifold of the surface mesh, edges with more than two adjacent triangles will be treated as illegal and deleted.

Thus, the filtering has the following content:

- Delete illegal edges
- Delete false triangulated triangles
- Delete small components

Figure 4 shows filtering of a 3D triangle-mesh model of a real meteorite fund in Bavaria, Germany. This model was generated with our hand-guided laser-scanner.

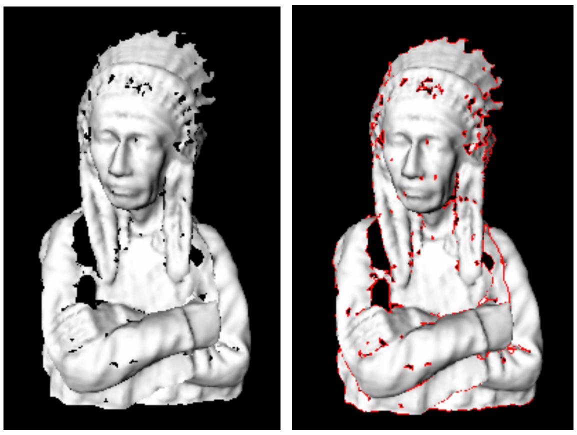


a) Original b) After filtering

Figure 4. Filtering of the 3D model of a real meteorite

3.3 Hole-Detection

After the filtering, all of the edges with only one adjacent triangle will be defined as ***boundary-edges***. Holes will be generated as closed chains out of the boundary edges. Figure 5 shows the detected holes in the 3D triangle-mesh model of a Indian-figure.



a) Original mesh b) Detected holes

Figure 5. Detection of Holes

3.4 Hole-Filling

As the holes in 2D space can be filled by the modification of “ear clipping” or “sweep line” algorithms and the implementation of these algorithms can by fund in many books, e.g. “computational geometry in c” (O'Rourke, 1998), so we need not describe it here. We simply denote the function to fill holes in 2D space as fill_simple_2D_hole(vertex s), with s as one vertex at the boundary of the hole.

The main problem in 3D model of real buildings is the large holes with complex form in 3D space. So we set the focus of this paper as: how to divide a complex and large 3D-hole into a group of simple-formed 2D-holes. It will be done by the following recursive algorithm.

```
fill_complex_3D_hole (vertex s)
{
    1. init plane P = (s, s→pre, s→next);
    2. search in the direction of s→pre until find a
       vertex a not in the plane P;
    3. search in the direction of s→next until find
       a vertex b not in the plane P;
    4. add an edge E from b→pre to a→next;
    5. fill_simple_2D_hole(s);
    6. reverse edge E;
    7. fill_complex_3D_hole(b);
}
```

Figure 6. Algorithm for filling of complex 3D-holes

The data structure of vertex on the hole is shown below:

```
Class vertex
{
    vertex* pre;
    vertex* next;
    float value[3];
}
```

We set two pointers in one vertex: one indicates the pre-vertex of it; and the other points at the next-vertex. The ordering of the pre- and next-vertices is made by the triangulation direction of the adjacent triangles on the boundary of the hole.

As illustrated in Figure 7, by searching in the pre- and next-directions, a simple-2D-holes is generated and the rest of the hole will be filled recursively.

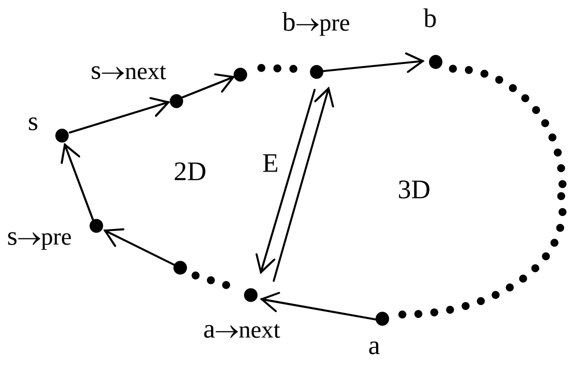
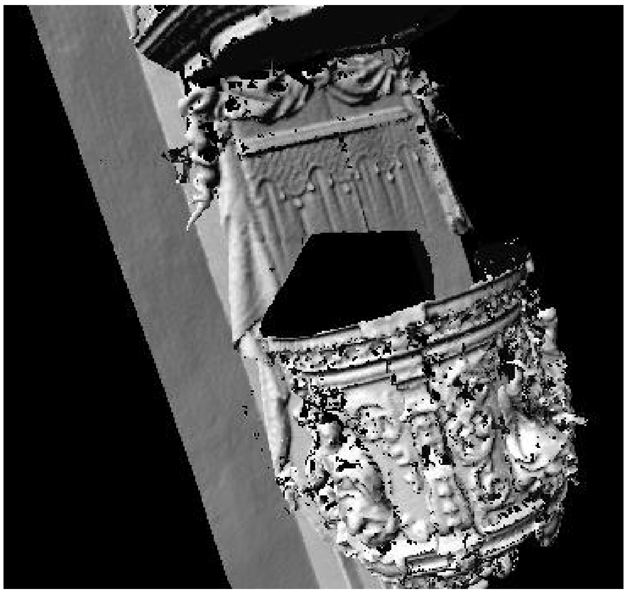


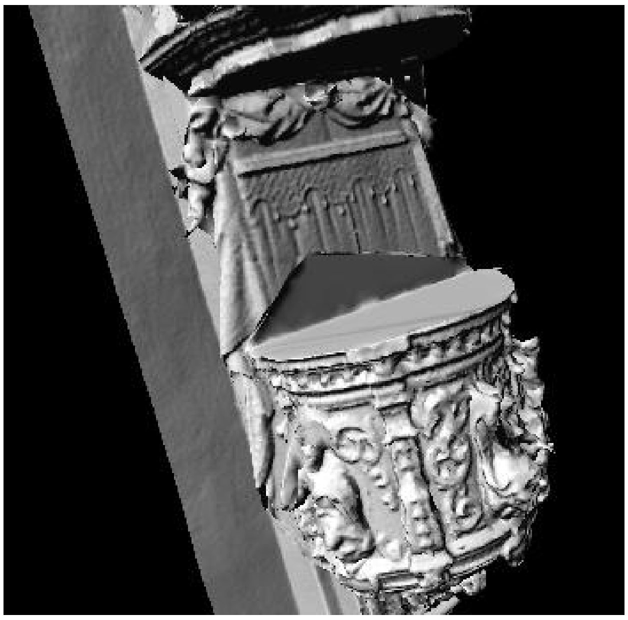
Figure 7. Recursive filling of complex 3D-holes

By the reconstruction of cultural heritage with laser range scanner, many large holes with over 1000 vertices auf boundary appeared beside thousands of small holes. With the above recursive algorithm, all of the holes were filled automatically.

Figure 7 shows the reconstructed model of the pulpit in one Church of Seefeld in Bavaria, Germany. In the middle of the model, a big hole with about 1200 vertices on it and they are lying on the different planes.



a) Mesh with holes
(including a big hole with over 1000 vertices on the boundary)

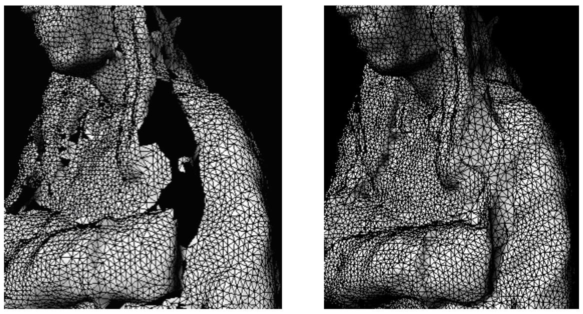


b) All of the holes are filled automatically

Figure 8. 3D model of the pulpit in one Church of Seefeld

3.5 Subdivision

To gain a better smoothing effect, the filled triangles in holes should be divided to the average size of the triangles in the original mesh. Figure 9 shows a part of 3D Indian mesh-model. The holes were filled with subdivided triangles.



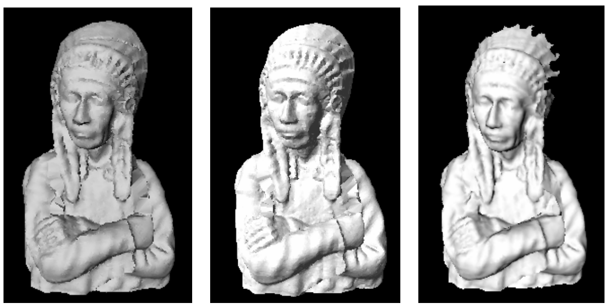
a) mesh with holes b) fill-patch subdivision

Figure 9. Subdivision of the filled triangles

3.6 Smoothing

With a a scale depended fairing algorithm (Desbrun et al. 1999), the filled mesh will be smoothed.

One of the important differences between the models generated by Tight-Cocone / Power Crust and ours is that our model is smoothed, which shows a better visual effect. And we set the extremely large holes as legal holes and let them open (see the outer contour of Indian in c) of Figure 10) .

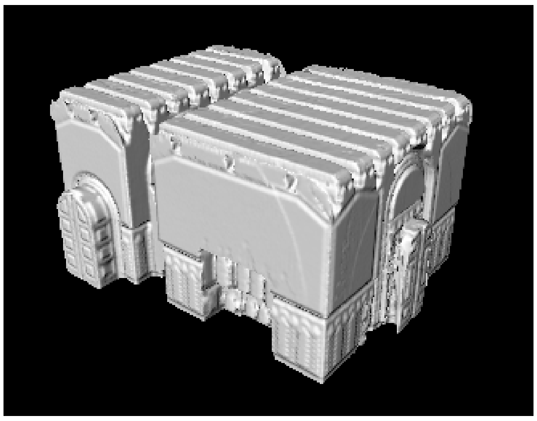


a) tight-cocone b) power crust c) our method

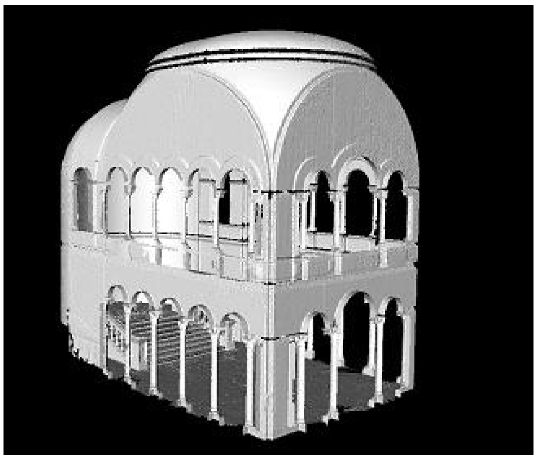
Figure 10. Comparison between the end-results of Tight-Cocone, Power-Crust and our method

4. EXPERIMENTS AND RESULTS

We have use our method both in the reconstruction of various historical buildings in Bavaria, Germany, and in the modelling of small figures captured by the hand-guided laser scanner developed in our institute. In Figure 11, some examples of them are shown.



a) King's working room in the Neuschwanstein Castle



b) Throne Hall in the Neuschwanstein Castle

Figure 11. Automatically reconstructed historic buildings

5. CONCLUSIONS

For the development in the future, two important problems should be solved: the first is automatic filling of very long vacancies (see the Throne Hall model in Figure 11), and the second is to present the whole model in a more simplified and compressed form. This is especially necessary for the reconstruction of cultural heritages, for there are over millions of triangles in one model.

6. ACKNOWLEDGEMENTS

We would like to give our thanks to Tamal Dey and Nina Amenta for making their software available.

And further thanks to Bernhard Strackenbrock, Michael Suppa, Tim Bodenmueller, Johann Heindl and Wadim Tolstoi for the useful discussions and delivery of related data.

7. REFERENCES

Amenta N., Choi S. and Kolluri R., 2001. The power crust. In: *Proceedings of 6th ACM Symposium on Solid Modeling*, pp. 249-260.

Bernardini F., Mittleman, J., Rushmeier, H, Silva C. and Taubin G., 1999. The ball-pivoting algorithm for surface reconstruction. In: *IEEE Transaction on Visualization Compute Graphics*, archive Volume 5, Issue 4, pp. 349-359.

Bodenmüller T. and Hirzinger G., 2004. Online surface reconstruction from unorganized 3D-points for the DLR hand-guided scanner system. In: *Proceedings of 2nd Symposium on 3D Data Processing, Visualization, Transmission*. pp. 285-292.

Boissonnat, J.-D., 1984. Geometric structures for three-dimensional shape representation. In: *ACM Transactions on Graphics*, 3(4), pp. 266-286.

Curless B. and Levoy M., 1996. A volumetric method for building complex models from range images. In: *Proceedings of ACM SIGGRAPH 96*, pp. 303-312.

Desbrun M., Meyer M., Schröder P. and Barr A. H., 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In: *proceedings of ACM SIGGRAPH 99*, pp. 317-324.

Dey T. K. and Goswami S., 2003. Tight cocone: A watertight surface reconstructor. In: *Proceedings of 8th ACM Symposium on Solid Modeling Application*, pp. 127-134.

Edelsbrunner H. and Mücke E. P, 1994. Three-dimensional alpha
shapes. In: *ACM Transaction on Graphics*, Volume 13, Issue 1, pp. 43-72.

Gopi M., Krishnan S. and Silva C. T., 2002. Surface reconstruction based on lower dimensional localized delaunay triangulation. In: *Computer Graphics Forum*, Volume 19, Number 3, pp. 467-478

Hoppe H., DeRose T., Duchamp T., McDonald J. and Stuetzle W., 1992. Surface reconstruction from unorganized points. In: *Proceedings of ACM SIGGRAPH 92*, pp. 71-78

O'Rourke J., 1998. *Computational Geometry in C (Second Edition).* Cambridge University Press.

Schall O. and Samozino M., 2005. Surface from scattered points - a brief survey of recent developments. In: *Proceedings of 1st International Workshop on Semantic Virtual Environments*, pp. 138-147.

Uselton S.P., 1983. Three dimensional surface descriptions from sensed data. In: *Proceedings of the 16th Hawaii International Conference on System Sciences.* volume. 2, pp. 387-385.