Modeling and Rendering Architecture from Point Cloud Data

Peter Asplund* Per Carlsson† Markus Wolmerud‡

Figure 1: Film production snapshot

Abstract

Methods for triangulation and mesh simplification of point cloud data are implemented. We present results that demonstrate our methods ability to create polygons of point data with different level of detail. The point cloud was acquired in collaboration with Leica Geosystems and the Östgöta Theater in Norrköping. A short movie with arbitrary view-points and exact representation of the scan site is also presented. The movie was made with Leica Geosystems software *Cyclone* 5.1.1 to obtain an insight into the triangulation and mesh simplification algorithms implemented in existing software.

Keywords: triangulation, mesh simplification, delaunay, 3D scanning, range image, point cloud, large datasets, cultural heritage

1 Introduction

Traditionally laser scanning has been used in construction to measure distances with high accuracy. The scanning is often used to obtain data more exact than blueprints that usually differ from reality. With the decreasing capture time and the improvement in mobility of laser scanners lately they are now used to scan sites for movie special effects or the preservation of cultural heritage [Levoy et al. 2000]. The main advantage of laser scanning is the accurate capturing of complex geometry.

*e-mail: petas874@student.liu.se
†e-mail: perca944@student.liu.se
‡e-mail: marwo513@student.liu.se

2 Theater Reconstruction

The elegant Östgöta Theater in Norrköping was selected as the scanning site. The complex geometry and rounded surfaces were intended to show the advantages of laser scanning. Together with a technician at Leica Geosystems 12 million points were acquired in two scans. The scanner used was a Leica HDS 3000. Using an accuracy of 20 mm at a distance of 25 meters from the scanner the total scanning time was 6 hours. The mounted digital camera was used to get RGB values for each point and markers seen from both scans were used to position the two scans in a mutual coordinate system.

2.1 Cyclone

The scan data was saved as a database file for the *Cyclone* software. Material was rendered from a number of camera shots and put together as a short movie. Figure 1 show some of the camera paths used in the movie. The movie artifacts can be divided into holes and distortion. The distortion effects is due to that 90% of the movie material used was rendered from the unedited point cloud. The holes is due to points not seen from the scanner, low reflectivity dark surfaces like parts of the chairs and holes resulting from the incorporated triangulation. The available hole filling tools in *Cyclone* were not examined since we did not make a hole filling algorithm of our own. The black parts of Figure 2 indicate holes.

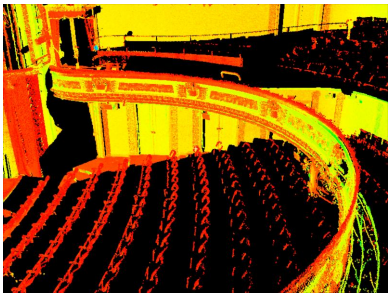

Figure 2: Holes in the point cloud

2.2 Point data selection

Due to the slow running time of our triangulation method we had to pick out regions of the theater point cloud to test our triangulation and mesh simplification algorithms. A standard 3D ASCII file consists of the points and a triangle list linking the points to polygons. To try the algorithms implemented just the points are read into Matlab and then triangle lists for different levels of mesh simplification are generated.

3 Triangulation

To go from a point cloud to a polygonal model triangulation is required. One of the most used (and best looking) is the Delaunay Triangulation [Amenta and Bern 1999], [Attali 1997]. The Delaunay triangulation is closely related geometrically to the Voronoi diagram, see Figure 3. The Voronoi diagram split the plane into a number of polygonal regions called tiles. Each tile has one sample point in its interior called a generating point. All other points inside the polygonal tile are closer to the generating point than to any other. The Delaunay triangulation is created by connecting all generating points which share a common tile edge. This makes the triangle edges perpendicular bisectors of the tile edges.

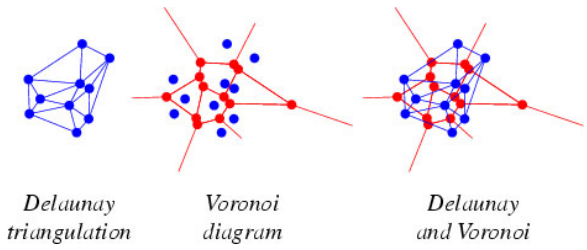

Figure 3: Delaunay/Voronoi relationship

The most desirable feature of the Delaunay triangulation is that It can be shown that the triangles are as equilateral as possible, thin triangles are avoided. There are many problems regarding this method though. The complexity of implementing it is one, and the memory required to triangulate large amounts of data is another. Many attempts have been made to solve these issues [Dey et al. 2001], [Amenta et al. 2000], but we choose not to implement them, due to high complexity. Our triangulation implementation has a running time of $O(n^{1.5})$, and is written in MATLAB, which results in an even worse efficiency. Our goal was not to achieve an efficient algorithm though, but to show how a Delaunay triangulation works, and also that we are able to implement one, by the help of earlier work made by others [Burke 1989]. Our algorithm works like the following:

1. Create a "super triangle" enclosing all the points. This is done the first time only.
2. Insert one of the points into the super triangle.
3. Check to see whether the point is inside the circumcircle of the triangle (a circle with the three vertices of the triangle on its circumference).
4. The triangles with a circumcircle that encompasses the point are deleted.
5. New triangles are formed between the surrounding triangles outer edges and the new point.

We now have twice as many triangles as points. At the end, the super triangle - and all triangles sharing edges with it - is deleted, leaving slightly less than twice as many triangles then number of points. The exact number depends on the distribution of points. This algorithm does not require large amounts of storage, only one array to hold the points, and another to tell which triangles no longer need be considered in the calculations. When comparing our method to the MATLAB function *Delaunay* the result is the same since the Delaunay triangulation for a certain point cloud is unique. Figure 4 show our triangulation of a small selection of the theater point cloud (1500 points).

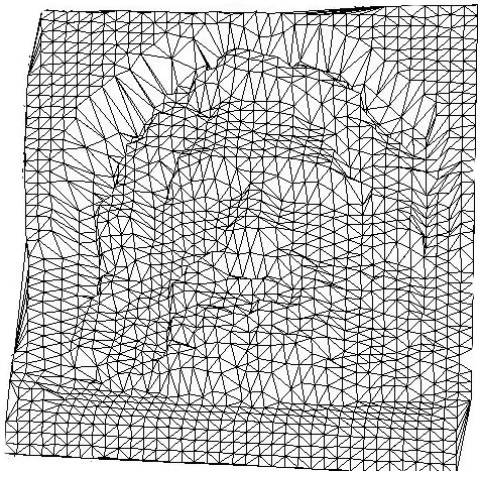

Figure 4: The triangulator in action

4 Mesh Simplification

The triangulation produces nearly two triangles for every inserted point. The theatre point cloud of 12 million points gives us 24 million triangles to deal with. To be able to work with and render the polygons the amount of triangles has to be decimated. Even if the number of triangles is reduced by 90%, conventional 3D software like 3D Studio Max use 600 megabyte of ram just to open the file.

The mesh simplification algorithm [Melax 1998] iterates through the vertex list and calculates which point that will be erased next by estimating a cost function. First the triangles with both the point to be deleted u and the most suitable adjacent point v is selected and removed. The remaining triangles are then rearranged to use v as a vertex instead as u as earlier. The last step is to remove vertex u from the vertex list.

A quick and memory efficient simple mesh simplification selects the edges to be collapsed by random. The drawback of this method is the poor quality of the simplified mesh especially when many points are deleted. No account to the geometry of the mesh is taken in this method. The cost to collapse an edge uv is determined by a random function. Our current implementation uses a random method leaving the geometry dependent approach as future work.

The reference number to u is randomly selected from the total amount of points. One adjacent point to u (v) is selected by iterating through the triangles containing u and randomly selecting one v from all possible ones. A list of the triangles with both u and v in them are created. Those are deleted from the original triangle list by setting the triangles to a certain triangle (here 1 1 1). In the end the triangle list can be sorted and a new smaller triangle list is

created. The remaining triangles that use u as a vertex are set to use v by iterating through the triangle list. Finally vertex u is deleted from the vertex list. Figure 5(a) show the original face, triangulated with our method. The lower image of Figure 5 show the face after a reduction from 1500 to 1000 points using our mesh simplificator.

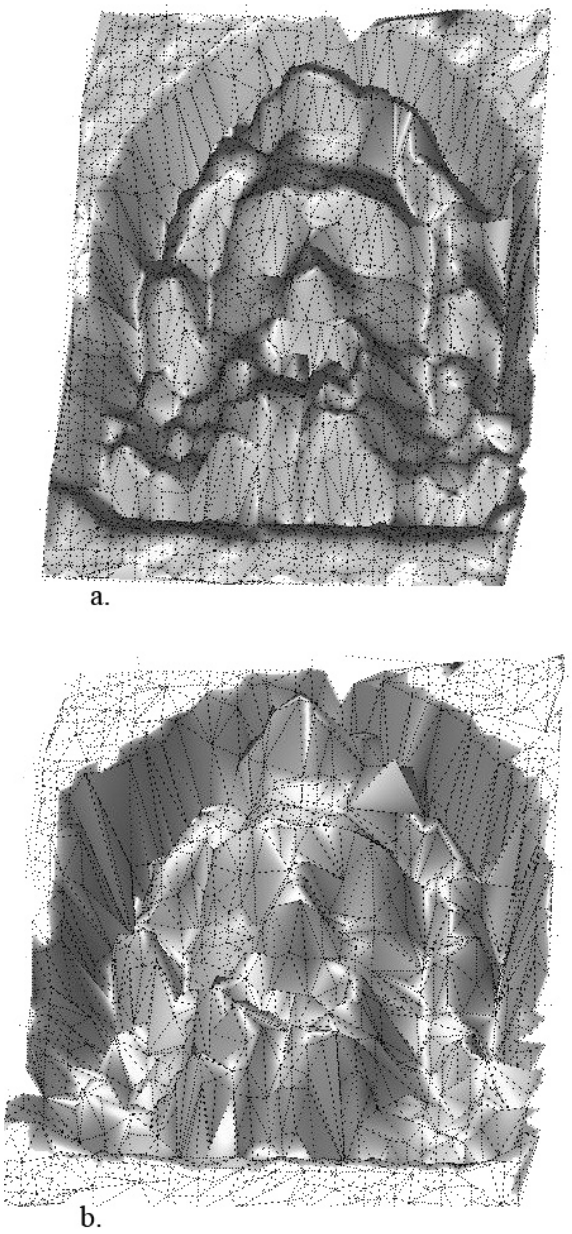

Figure 5: The mesh simplificator in action

5 Conclusion and future work

Our method for triangulation gives fine result when triangulating the 1500 point face. It's even better then the triangulation method incorporated in *Cyclone* for this specific example because it doesn't contain holes, see Figure 6(b). But there are two major limitation of our triangulation. The first is the 2D characteristic of the algorithm. It's referred to as a terrain modeling since it only supports a single height value for each horizontal coordinate. As described in section 3 the interior of any circumcircle for each of the triangle does not contain any other vertices. This makes it impossible to achieve a fine result if the entire point cloud were to be triangulated. If the algorithm would be extended to 3D this translates to that the interior of any circumsphere for each of the tetrahedrons does not contain any other vertices. The second limitation is computation time being $O(n^{1.5})$. This sets the maximum number of vertices to be triangulated in a reasonable time to approximately 100 000. The algorithm computation time would be improved by using a divide and conquer approach.

Our method for mesh simplification is fast and memory efficient. The decimation of the face to any desired number of polygons is done in seconds. The major limitation is the lack in quality since the edges to collapse are randomly chosen. The decimation of the mesh can be done more intelligent by using a cost function that uses information about the objects geometry. One way to do this is to estimate a cost function which depends on edge length and angle of adjacent triangles. The cost is estimated for every edge and the edge with the smallest cost is collapsed. The procedure is repeated until desired result is obtained.

Our movie represents the complex geometry, light and spirit of the theater. A noticeable fact is how well the renderings of the unedited point data came out. 90% of the material used in the movie is unedited point cloud. The movie artifacts in terms of distortion and holes are described in section 2.1. To really improve the result a number of measures have to be taken. First a triangulation method that minimize the number of holes. Second advanced illumination, materials and camera tools. There exists no commercial software today that incorporates this. Scanning software like *Cyclone* is capable of managing the huge datasets using database systems but they only include primitive methods for material, light and cameras. 3D software like *3D Studio Max* have no competence in managing datasets of even moderate size (10 million polygons) since it overloads the RAM. With the available tools in each software, the most appealing picture was rendered, see Figure 6. Its clear that the 3D Studio Max rendering have a quality not possible to achieve in the scanning software. Another issue is that methods for triangulation and mesh simplification need to be state of the art [Dey et al. 2001] to give acceptable result in short execution time. They are to our knowledge yet to be implemented in existing scanning software.

The point cloud of the theater, HDR images, and photos are available on request for future work. A free (registration required) viewer to browse the point cloud is available at http://www.leica-geosystems.com/se/.

6 Acknowledgements

This project was made possible by Leica Geosystems for providing the HDS 3500 scanner and the personal at the Östgöta Theater for providing the scan site. The authors also wish to thank Lennart Krantz for on site maneuveringing the scanner and providing expertise regarding Cyclone and Dr. Mark Ollila for supervising the project.

References

AMENTA, N., AND BERN, M. 1999. Surface reconstruction by

voronoi filtering. In *Discr. Comput. Geom., 22*, ACM Press, 481–504.

AMENTA, N., CHOI, S., DEY, T. K., AND LEEKHA, N. 2000. A simple algorithm for homeomorphic surface reconstruction. In *SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry*, ACM Press, 213–222.

ATTALI, D. 1997. r-regular shape reconstruction from unorganized points. In *SCG '97: Proceedings of the thirteenth annual symposium on Computational geometry*, ACM Press, 248–253.

BURKE, P., 1989. Triangulate - efficient triangulation algorithm suitable for terrain modelling. Privately published.

DEY, T. K., GIESEN, J., AND HUDSON, J. 2001. Delaunay based shape reconstruction from large data. In *PVG '01: Proceedings of the IEEE 2001 symposium on parallel and large-data visualization and graphics*, IEEE Press, 19–27.

LEVOY, M., PULLI, K., CURLESS, B., RUSINKIEWICZ, S., KOLLER, D., PEREIRA, L., GINZTON, M., ANDERSON, S., DAVIS, J., GINSBERG, J., SHADE, J., AND FULK, D. 2000. The digital michelangelo project. In *Proceedings of SIGGRAPH 2000*, ACM Press / ACM SIGGRAPH, New York, K. Akeley, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 131–144.

MELAX, S. 1998. A simple, fast, and effective polygon reduction algorithm. *Game Developer Magazine*.

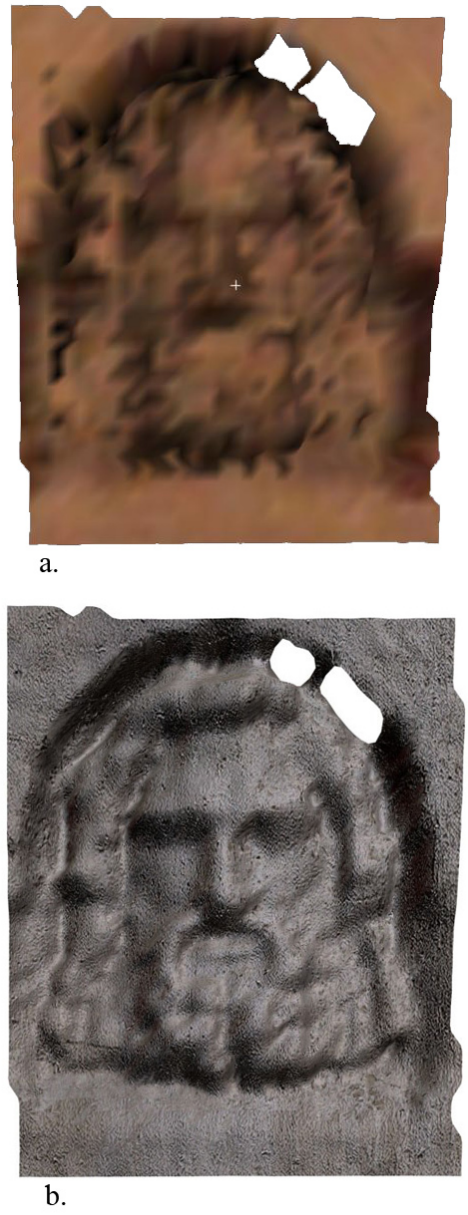

Figure 6: Face rendered in *Cyclone*(a) and *3D Studio Max*(b)