Real Time Landscape Modelling and Visualization

Rui Liu, Darius Burschka, Gerd Hirzinger
Institute of Robotics and Mechatronics,
German Aerospace Center (DLR)
Muenchner Str. 20, 82234 Wessling, Germany
Rui.Liu@dlr.de

***Abstract*—With the rapid development of information and communication technology and widely spreading of internet, digital landscape becomes a high topic recently. This paper presents a landscape modeling- and visualizationssystem. The first part of this paper focuses on the real time 3D-model-lingsprocess out of large point-clouds. Diverse experiments have been done on the reconstruction of various famous regions of Bavaria with tourist features. And the second part concentrates on the interactive online visualizationssystem for massive meshes. To enable an efficient interactive online visualization of these large meshes, we convert them firstly with a multi-resolution hierarchy, and then display them progressively. As a pre-processing for the inter-active rendering, we generate a hierarchical tree of bounding spheres from triangle meshes and write it to disk. It will be used for view frustum culling, backface culling and level-of-detail control. We use a recursive algorithm for display. And the traverse-depth in the tree is decided by the projected size of the current node on the screen. By an interactive rendering, once the user stops moving the mouse, the scene will be redrawn with successively smaller thresholds until a size of one pixel is reached.**

Keywords-environmental modelling; data mining, achieving and retrieval

I. INTRODUCTION

To generate digital Landscape, man uses airplane, airship, even satellite to gain the colored aerial views outdoors. Of course, the obtained data amounts are very large. Therefore, how to handle these data, to simplify them, to transform them to 3D models fast and automatically for further applications, e.g. CAD modeling or interactive online visualization etc., becomes a challenging task.

The first part of this paper presents a fully automatic 3D-modeling process of the huge 2.5D landscape point-clouds generated with a High Resolution Stereo Camera (HRSC) [1] assembled on a flying airplane (See Fig. 1). This camera has been developed by the DLR Institute of Planetary Research for the exploration of the Mars surface. The airborne version HRSC-AX is currently used for capturing earth's landscape and cities from flight altitudes between 1500m to 5000 m [2]. The input data of our modeling process are resulted from the stereo-matching [3].

To handling these large datasets we designed a real-time automatic modeling process, which is performed in six steps: tilling with overlap, mesh generation, mesh simplifying, mesh cut, mesh merger and texture mapping.

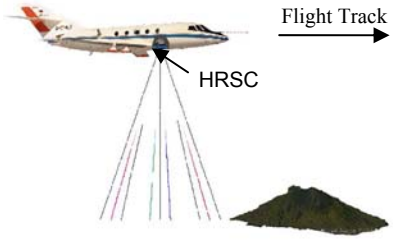


Figure 1. HRSC on Airplan.

And the second part of this paper presents an interactive online visualisationssystem for massive colored meshes. Firstly, we generate a hierarchical tree of bounding spheres from triangle meshes and write it to disk. And then it will be read progressively. During the display, view frustum culling, backface culling and level-of-detail control will be done.

This paper is organized as follows. In section 2, previous work is briefly summarized. Then, section 3 describes the workflow of automatic modeling and section 4 illustrates the details for visualization, e.g. the utilized data structure, rendering algorithms, etc. And the experimental results are shown in the two sections. Finally, the related acknowled-gement is announced in the end section.

II. RELATED WORK

For the modeling and visualization of landscape data, LOD (Level of Detail) [4] is used as a traditional way. Recently, more and more efforts have been taken to find and reconstruct the man-made structures and natural structures automatically. For example, buildings, highways, trees etc. [5, 6, 7, 8, 9, 10].

The visualizationstechniques for very large dataset can be classified into two categories: the first approach focuses on the optimizing the placement of the individual edges and vertices [11]. Though it can result in a gut rendering quality, it requires more runtime and memory. So it is not suitable for very large dataset. The typical work in this field is the combination of traditional mesh simplification and progressive display. And the second approach treats the individual points as relatively unimportant and consumes less effort per primitive. It results in a high rendering speed and less memory requirements, but leads to a relative lower quality. Typical systems are Yemez and Schmitt's rendering system based on octree particles [12], Rusinkiewicz's splat system based on a balanced hierarchical

tree [13]. The improvement of the point-based rendering quality has been taken by EWA splatting [14] and object-space point blending [15]. Recently, some researches combined both of them in LOD-based rendering approach [16, 17].

III. MODELING

The input of our system is the 2.5 D colored point-clouds resulted from the photogrammetric preprocessing. Because one of such aerial views is normally several gigabytes, it overrides the capabilities of fast all 3D modeling- and visualizationstools. So we run an automatic modeling process on these data to reduce the data amount firstly. Here we take a "divide-and-conquer" strategy: piecewise reading and modeling, then making the border of the adjacent parts seamless. So our approach of fully automatic 3D-modelling consists of the following six steps.

A. Tiling with overlap

In this stage, the whole area will be divided in tiles with a proper size. To avoid great distort out of merging of meshes, we set an overlap between neighbored tiles.

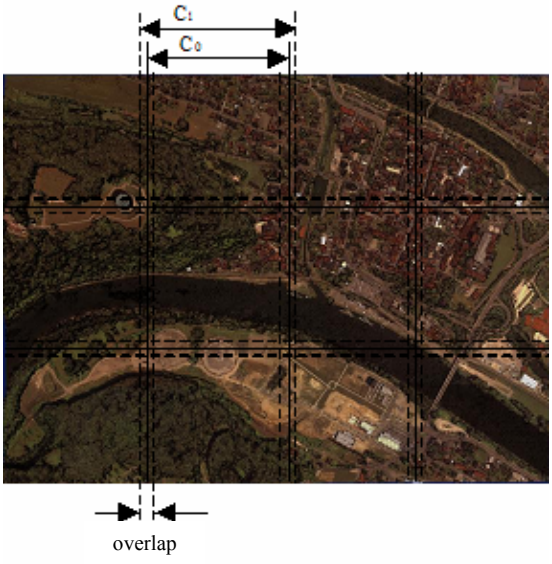


Figure 2. Tiling with overlap.

For a fast reading of input data in arbitrary areas with any resolution, the input data were saved as double-Tiff-format: one Tiff-file contains the height-information of every pixel, and the other holds the RGB colored information.

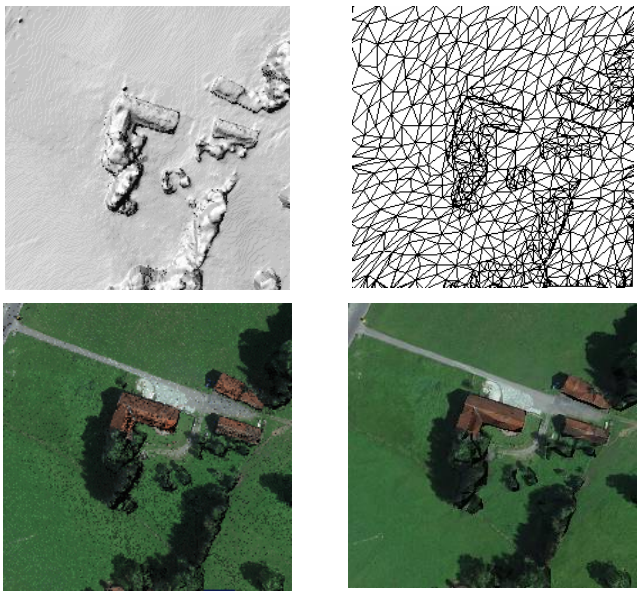
B. Mesh generation

As the heights of pixels are saved as matrix-format in tiff-file, by connecting the shorter diagonal of every rectangle grid, we get a regular grid mesh.

C. Mesh simplifying

A modified Quadric Error Metrics algorithm [18] was applied in this stage. Geometric characters become more outstanding, for example, edge becomes sharper. And the data amount can be reduced over 99% without distinguish distort.

Fig. 3 shows a little area with several houses. The original grid mesh is generated from HRSC 2.5D data with resolution of 20 cm. It has 359,552 triangles. After simplification, only 0.5% triangles (1,795) are remained. However, the geometric characters become more outstanding, and the appearance of the textured 3D model is improved considerably.



a) before b) after

Figure 3. Mesh Simplifying.

D. Mesh cut

In this step, the bounding box of each tile will be generated according to the tile size and position. Exactly along the middle axis of the overlapping area, every mesh will be cut neatly. That means, the projection of all resulted boundary edges from one cut action should lie in one straight line.

E. Mesh merge

After neatly cutting of the simplified meshes, we apply the following "stitch-actions" to merge the adjacent meshes seam-less.

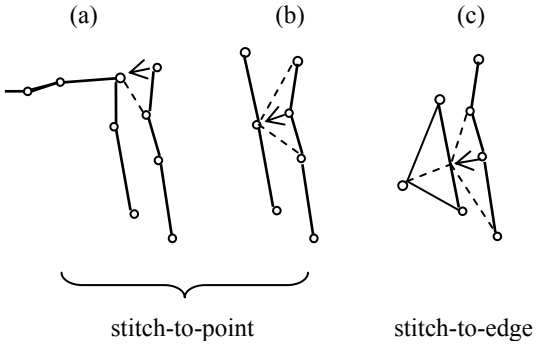


Figure 4. Merger of adjacent meshes.

F. *Texture mapping*

The RGB information of each tile was saved as compressed image format. And the mapping from texture to 3D mesh is achieved by projection the 3D meshes to the texture coordinate system.

The above six steps are integrated in one process, which can be completed in 20 ~ 30 minutes for one area with 283 million points (See Tab. 1). The resulted 3D landscape is saved as a group of simplified meshes with corresponding texture.

TABLE I. PROCESS TIME OF DIVERSE REGIONS

Region	**Area Size** n x m (pixels^2)	**Resolution** (m)	**Output** (Triangles)	**Runtime** (min)
Kehlheim	24480 x 11570	0.20	3,623,330	26,5
Andechs	6000 x 6000	0.15	463,333	3,6
Wiekirche	6000 x 6000	0.20	465,557	3,7

IV. VISUALIZATION

To enable an efficient interactive online visualization of these large meshes, we convert them firstly with a multi-resolution hierarchy, write them to disk and display them progressively.

A. *Datastructure*

We use a balanced hierarchical tree of bounding spheres as our basis structure to hold and display the mesh (See Fig. 5). There three types of tree-nodes: a root, many inner nodes und leaves. Each leaf contains a pointer to his underling 3D-object (this can be a point, a triangle or a group of triangles, etc.) generated from the original mesh. It also contains the center and radius of the bounding sphere of the 3D-object, a normal, width of the normal cone and optionally a color. A none-leaf node (inner node or root) contains the number of his child-nodes, the center and radius of the bounding sphere enclosing his child-nodes, a normal, width of the normal cone and optionally a color. A none-leaf node has maximal four children. Its properties, such as normal and color, are set as the average properties of his children.

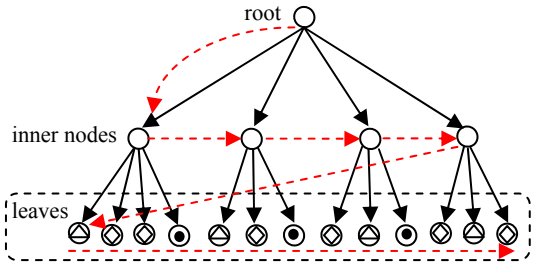


Figure 5. A balanced hierarchical tree of bounding spheres.

And the save order of the nodes in disk is along the rot dotted arrows.

B. *Preprocessing*

In this process, we build the balanced tree from triangles meshes. Firstly, we init the leaves by computing the bounding sphere of the chosen 3D-object from mesh. And then, we build up the tree by using a recursive algorithm from bottom to top.

To get a balanced tree, we split the set of nodes along the longest axis of its bounding box at the node with the middle range in the node-set by every recursive call, until the size of the node-set is smaller than five, then we construct a parent node out auf this node-set.

After the tree is built, we write it out in a compressed format auf the disk along the order denoted by the rot arrows in Fig. 5. For the size of every node is pre-defined according to the compression method, we can directly access the sub-trees of one node.

C. *Rendering*

We use the following algorithms for an efficient read and display of the hierarchical tree.

```
int Draw(tree-node* n )
{
    if (!n or n not visible)
        Return 0;
    if (projection size of n on screen under a threshold)
    {
        Draw a splat;
        Return 1;
    }
    if (n is leaf node)
    {
        Draw the 3D-object linked to n;
        Return 1;
    }
     Return Draw(n->child[0])+Draw(n->child[1])+
            Draw(n->child[2])+Draw(n->child[3]);
}
```

Figure 6. Pseudo-code of the rendering algorithm..

The above code shows the principle of level-of-detail control. And to decide the visibility of a node, we use view frustum culling and backface culling.

1) *View frustum culling*

View frustum culling is performed by testing the bounding sphere of the node against the view frustum. If the sphere lies outside the frustum, the node is not visible (See Fig. 7).

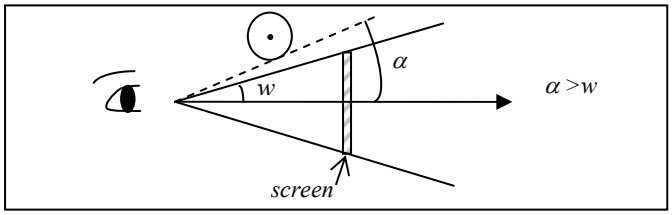


Figure 7. View frustum culling.

2) Backface culling

Backface culling means, if the normal cone of the node is completely facing away from the viewer, the node is not visible (See Fig. 8).

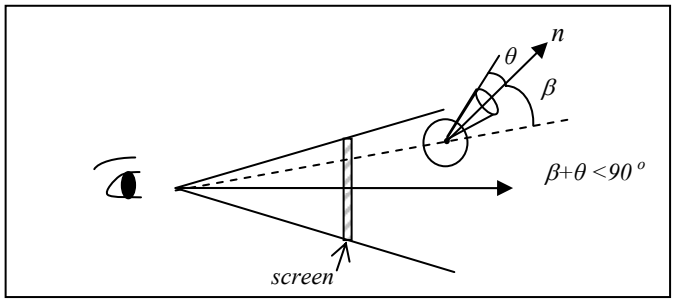


Figure 8. Backface culling

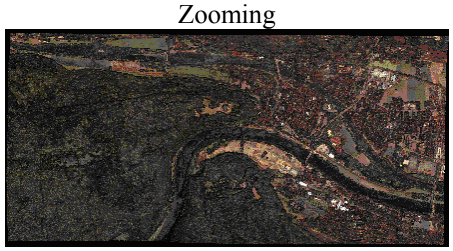
An example of our visualization is shown below (See Fig. 9). The number of the drawn objects and rendering time are listed.

Start of loading

375,597

0.039 s

Zooming



1,209,808

0.276 s

Zooming and rotating

1,582,751 0.368 s

Figure 9. Visualisation of Kehlheim: 11.33 km^2, 283 million points as input for our modelling- and visualisationssystem

ACKNOWLEDGMENT

Thanks to Mr. Johann Heindl and Dr. Heiko Hirschmüler, they give us useful advices and sufficient input data to test our modeling process. Further thanks to Mr. Wadim Tolstoi for his excellent support for our work.

REFERENCES

[1] F. Wewel, F. Scholten, and K. Gwinner, "High resolution stereo camera (HRSC)-multispectral 3D-data acquisition and photogrammetric data processing," Canadian J. Remote Sensing, 26:466–474, 2000.

[2] G. Hirzinger et al., "Photo-Realistic 3D-Modelling - From Robotics Perception Towards Cultural Heritage," International Workshop on Recording, Modelling and Visualization of Cultural Heritage, Ascona, Switzerland, 2005.

[3] H. Hirschmüller, F. Scholten, G. Hirzinger, "Stereo Vision Based Reconstruction of Huge Urban Areas from an Airborne Pushbroom Camera (HRSC)", in Lecture Notes in Computer Science: Pattern Recognition, Proceedings of the 27th DAGM Symposium, Volume 3663, pp. 58-66, Vienna, Austria, 2005.

[4] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, and R. Huebner, "Level of Detail for 3D Graphics," Morgen Kaufmann Publisher, 2003.

[5] S. Lefevre, J. Weber and D. Sheeren, "Automatic Building Extraction in VHR Images using Advanced Morphological Operators," in Proc. of Urban Remote Sensing Joint Event, Paris, 2007.

[6] C. Iovan, D. Boldo and M. Cord, "Automatic Extraction of Urban Vegetation Structures form HR Imagery and Digital Elevation Model," in Proc. of Urban Remote Sensing Joint Event, Paris, 2007.

[7] G. Sithole, G. Vosselman, "Automatic structure detection in a point-cloud of urban landscape", Proceedings of the 2nd GRSS/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas (URBAN2003), pp. 67-71, Berlin, Germany, 2003.

[8] M. Niederöst. "Detection and reconstruction of buildings for a 3-D landscape model of Switzerland," Proceedings of UM3/2000 Workshop, Tokyo 2000.

[9] I. Laptev, H. Mayer, T. Lindeberg, W. Eckstein, C. Steger and A. Baumgartner. "Automatic extraction of roads from aerial images based on scale space and snakes," Machine Vision and Applications (2000) 12: 23–31

[10] M. Gerke, B.M. Straub and A. Koch. "Automatic Detection of Buildings and Trees from Aerial Imagery Using Different Levels of Abstraction," Publikationen der Deutschen Gesellschaft für Photogrammetrie und Fernerkundung, Band 10, Eckhardt Seyfert (Hrsg.), pp.273-280, 2001.

[11] H. Huppe, "Smooth. View-Dependant. Level-of-Detail Control and its Application to. Terrain Rendering, " IEEE Visualization '98

[12] Y. Yemez and F. Schmitt, " Progressive Multilevel Meshes from Octree Particles, " in Proc. of 3D Digital Imaging and Modeling, 1999.

[13] S. Rusinkiewicz and M. Levoy, "Qsplat: A multiresolution point rendering system for large meshes, " In Proc. of SIGGRAPH, pp. 343–352, ACM SIGGRAPH, 2000.

[14] L. Ren, H. Pfister, and M. Zwicker, "Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering, " Computer Graphics Forum, 2002, 21(3): 461～470.

[15] R. Pajarola, M. Sainz, and P. Guidotti, "Object-space blending and splatting of points," Technical Report UCI-ICS-03-01, The School of Information and Computer Science, University of California Irvine, 2003.

[16] T. K. Dey and J. Hudson, "PMR: Point to mesh rendeering, a feature-based approach," in Proc. of IEEE Visualization, pp.155–162, 2002.

[17] L. Coconu and H.-C. Hege, "Hardware-oriented point-based rendering of complex scenes," in Proc. of Eurographics Workshop on Rendering, pp. 43–52, 2002.

[18] M. Garland, P.S. Heckbert. "Surface Simplification using Quadric Error Metrics," Proc. of the ACM SIGGRAPH' 97, pp. 209-216, Los Angeles, California, 1997