AUTOMATIC INTEGRATION OF FOLIAGE INTO 3D CITY MODELS
Real-Time Tree Extraction and Rendering Based on Seasonal Large Scale Aerial Pictures

Timo Ropinski, Frank Steinicke, Jennis Meyer-Spradow and Klaus Hinrichs
Institut für Informatik, Westfälische Wilhelms-Universität Münster, Germany

Keywords: 3D city visualization, treetop extraction, massive texture rendering, aerial photography.

Abstract: In this paper we introduce visualization techniques for massive multimodal texture datasets. These techniques work on registered texture datasets and have been developed with the goal to improve rendering of these datasets when used in virtual landscape or city environments. We briefly discuss how to render these multimodal datasets at interactive frame rates, and we present an interactive treetop extraction technique, which allows to segment treetops and visualize them as 3D objects to increase realism.

1 INTRODUCTION

With the widespread use of geographic information systems and the increasing availability of image acquisition systems the amount of generated geo-spatial data has increased dramatically within the last years. Systems like Google Earth have demonstrated that the interactive exploration of these datasets finds many application areas in the domains of tourist information, way finding, climatology and many more. Two aspects of applications supporting the exploration of geo-spatial datasets are interactivity and visual representation. Usually there is a trade-off between these two aspects, since in general visually appealing representations require more complex computations which results in reduced frame rates and thus interferes with the interactive experience of the user. In this paper we will propose interactive visualization techniques for large scale raster data which generate appealing visual representations at interactive frame rates.

Since nowadays for many locations more than one aerial image is available, multiple information layers can be taken into account for the presentation, possibly showing a location during different seasons and/or weather conditions. Since the information required for a location may be contained in different aerial images, novel visualization techniques are needed to interactively analyze and fuse this multimodal content. We have developed techniques which support the interactive exploration of multiple registered large scale aerial images, each having a size of possibly several gigabytes. In particular we will show how to combine the information stored in two registered aerial images in order to achieve a more appealing visual representation in geo-virtual environments. Our techniques use a summer and a winter aerial image to extract treetops during runtime. These extracted 2D treetops are extended to 3D and visualized as objects of the geo-virtual environment.

All concepts proposed within this paper have been tested using two registered aerial image datasets (see Figure 1). The datasets cover a $21km \times 24km$ area around the City of Münster at a resolution of $10cm \times 10cm$. They have been acquired in August 2001 and in January 2005; in the following we will simply refer to them as summer texture resp. as winter texture. Although we deal with two large scale texture datasets, having sizes of $4.33GB$ resp. $7.80GB$, all techniques proposed in this paper can be applied during runtime at interactive frame rates. Thus it is possible to integrate them into client side applications streaming data from map servers without requiring interface adaptions with the data provider.

Ropinski T., Steinicke F., Meyer-Spradow J. and Hinrichs K. (2007).
AUTOMATIC INTEGRATION OF FOLIAGE INTO 3D CITY MODELS - Real-Time Tree Extraction and Rendering Based on Seasonal Large Scale Aerial Pictures.
In *Proceedings of the Second International Conference on Computer Graphics Theory and Applications - GM/R*, pages 299-304
DOI: 10.5220/0002081402990304

(a) summer texture (b) winter texture

Figure 1: Parts of the summer and the winter texture.

2 RELATED WORK

Several techniques have been developed with the goal to achieve interactive frame rates when rendering large scale texture datasets (Hua et al., 2004; Brodersen, 2005). These techniques mainly differ in the supported feature set as well as in implementation and runtime complexity. To achieve interactive frame rates when rendering multimodal texture datasets, we have decided to implement the clipmapping technique originally proposed by Tanner et al. (Tanner et al., 1998). To exploit clipmapping for our visualization techniques we have extended the implementation to maintain more than one clipmap hierarchy, i.e., in our case one for each modality. For a more realistic representation aerial textures need to be projected onto terrain models. Therefore we have integrated the geometry clipmap proposed by Asirvatham and Hoppe (Asirvatham and Hoppe, 2005) into our terrain rendering engine.

Nowadays remote sensing technologies allow an exact classification of certain usage areas (Castel et al., 2001; Lee et al., 2004; Schlerf and Atzberger, 2006). Although, it is possible to extract forest coverage as well as treetops, most of the techniques either require additional registered LIDAR (= light detection and ranging) datasets or are not applicable in real-time.

Many algorithms for the interactive rendering of trees and foliage have been presented in the last years. Since this paper does neither address natural modeling of trees nor specialized LoD representation we simply refer to (Lluch et al., 2003; Deussen et al., 2004) for an overview of some approaches.

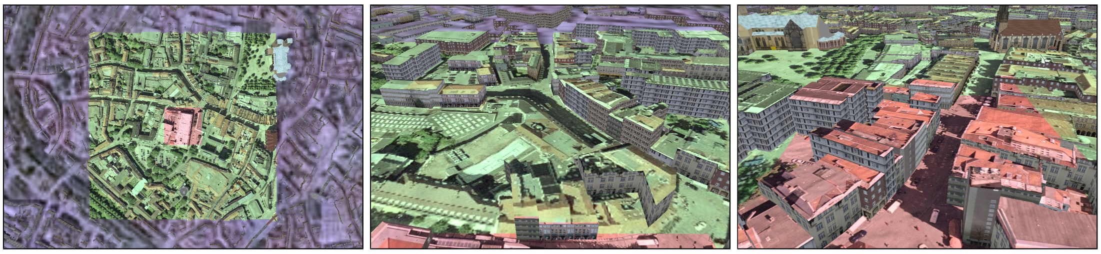
3 LOD DETERMINATION

An important aspect of LoD rendering techniques is the determination of the appropriate LoD for a given point or region in space. Since the LoD is view-dependent the view frustum defined by the camera's position and orientation as well as further properties such as the associated clipping planes have to be considered when determining the correct LoD. Since Tanner et al. (Tanner et al., 1998) have not covered LoD determination in their description of clipmapping, we will describe our approaches to find the correct LoD for a given point or region in space. We distinguish between two different cases of camera control in order to provide an *optimal* arrangement of clipmapping textures: (a) in the 2D case, the view direction of the camera is orthogonal to the ground plane, whereas (b) in the 3D case, the camera is arbitrary. The goal is to arrange the clipmapping textures in such a way that most of the ground plane within the view frustum is covered with textures of the highest quality. For simplicity, we reduce the region for which a LoD has to be determined to a single point in three-dimensional space, which is located on the ground plane. We apply a function $height(x,y)$ that returns the height at the position (x,y). In the following *cmc* denotes this optimal clipmap center position in 3D world coordinates.

3.1 3 DoF Top-View Camera

For the simple case in which the camera is constrained to three degrees of freedom (DoF), i.e., the viewer explores the virtual environment from a top-view perspective where the view direction is focussed along the negative z-axis towards the ground plane, the camera position $p = (p_x, p_y, p_z)$ is the only camera attribute that has to be considered. We define the function $lod(p) = min(\{l|(x^l_{min} \leq p_x \leq x^l_{max}) \wedge (y^l_{min} \leq p_y \leq y^l_{max})\})$ that returns the appropriate LoD for p, i.e., the level of the clipmap hierarchy, such that $lod(p) \in [0, l_{max}]$ where 0 resp. l_{max} is the index of the clipmap level representing the highest resp. lowest resolution; and the clipmap at level k covers the area reaching from (x^k_{min}, y^k_{min}) to (x^k_{max}, y^k_{max}).

With these prerequisites the determination of the correct LoD can be reduced to the calculation of the optimal clipmap center *cmc* for a given camera configuration. We assume that the base plane of the height field onto which the aerial photograph has to be projected is always parallel to the xy-plane having an averaged surface normal $\vec{n} = (0,0,-1)$. Hence, assuming that the viewer focuses on the center of the viewport, we assign the clipmap center $cmc_l = (p_x, p_y, height(p_x, p_y))$ for all levels l of the clipmap hierarchy and we can determine the correct LoD for each camera position p. Thus all clipmap textures are arranged concentrically. The result is shown in Fig-



(a) camera centered 3 DoF view (b) camera centered 6 DoF view (c) our technique 6 DoF view

Figure 2: Visualization of different LoD determinations. The LoDs are color coded: highest/red, middle/green, lowest/blue.

ure 2(a), where red, green and blue are used to color the first three levels of the clipmap hierarchy.

3.2 6 DoF Camera

Calculation of the optimal clipmap center in the 6 DoF case is more complicated. While in the aforementioned case we have assumed the optimal clipmap center to be $cmc_l = (p_x, p_y, height(p_x, p_y))$, i.e., the orthogonal projection of the camera position onto the ground plane, such clipmap center is not sufficient for the general 6 DoF case.

As it can be seen in Figure 2(b) a camera setup with a large angle between the view direction and $\vec{n} = (0,0,-1)$, results in large areas of the highest LoD that cannot be seen by the user because they lie outside the view frustum. Since areas in front of the user are perceived best, low resolution in these areas results in unsatisfying visualizations. Thus the strategy to display the highest LoD in the area around the viewport center as done in the simple top-view case is not sufficient for the general case.

For a sophisticated visualization when using a 6 DoF camera we have to extend the clipmapping technique to allow different clipmap centers for the levels of the hierarchy. We redefine the set of clipmap centers cmc_l, where l denotes again the increasing index of a level in the clipmap hierarchy, without the constraint of concentric clipmap levels, and we perform a clipmap center calculation with the goal to maximize the areas in image space showing the highest LoD.

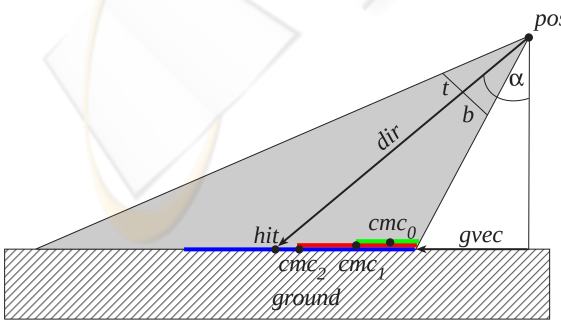


Figure 3: Schematic side view of the view frustum showing the configuration of clipmaps on the ground plane.

Figure 3 illustrates the procedure when arranging the different clipmap textures. As illustrated we assume that the angle α between the camera's view direction $\vec{d}$ and $\vec{n}$ is between 0 and 90 degrees. Since in this case the area which is projected directly above the bottom plane of the view frustum can be perceived best by the user, we have to assure that the highest level of the clipmap hierarchy is presented in this area. Therefore we have to determine the edge formed by the intersection of the ground plane and the bottom plane of the view frustum which is determined by b in the near plane, where t denotes the distance to the top of the field of view in the near plane (see Figure 3). This edge can be calculated by using the angle α as well as the ground vector $g\vec{vec}$ from $(p_x, p_y, height(p_x, p_y))$ to the intersection point hit of the vector $\vec{dir}$ with the height field. The length of $g\vec{vec}$ is given by the distance from $(p_x, p_y, height(p_x, p_y))$ to the corresponding edge of the view frustum using b (see Figure 3). The center of this edge is used to position the clipmap with highest quality at cmc_0. Moreover, we add an offset defined by the half of the clipmap texture's size to shift the texture along the ground vector $g\vec{vec}$ in order to ensure that the user can always see the entire clipmap. The next clipmap centers are shifted analogously to cmc_1, cmc_2 etc. as illustrated in Figure 3. As depicted the clipmap textures do not overlap concentrically, but the front edges, i.e., closest to the viewer, of the clipmap textures coincide. Hence, with decreasing distance between the viewer and the virtual environment displayed on the ground plane, the quality increases up to the highest resolution shown directly in front of the user.

As explained in Section 3.1, when the user focuses on a region orthogonally under her position we use the concentric arrangement of the clipmap textures

(a) trees projected on the ground (b) 3D model rendered on top the projected trees

Figure 4: Two images showing how trees are integrated in nowadays 3D city models.

around the hit point $hit = (p_x, p_y, height(p_x, p_y))$. Therefore, when the angle α gets smaller than a certain threshold we switch to the approach described in Section 3.1. We experienced best results for an angle of about 15 degrees (see Figure 2(c)).

4 AUTOMATIC FOLIAGE INTEGRATION

When combining 2D aerial pictures with 3D city models a common problem arising is that real objects contained in the aerial picture are visualized as projections on the ground as long as no 3D model is provided. In the cases where a 3D model is visualized at the appropriate position, this may not be an issue. Since the projection shown on the aerial picture is either not visible because it is occluded by the model, or the aerial picture is projected onto the model providing it with a realistic coloring. The second alternative is commonly used in 3D city models in order to texture buildings. However, in the areas were no 3D geometry is present the projections on the aerial pictures look unrealistic. Especially when using aerial pictures containing many trees, this issue becomes obvious and results in a very unnatural appearance (see Figure 4(a)). The commonly used strategy to deal with this problem is to place a 3D object, in this case a tree, at the appropriate position of the aerial picture (see Figure 4(b)). Unfortunately when using 3D tree models, in general the trunk does not cover the whole area of the aerial picture which is covered by the tree as seen from above. With the technique presented in this section it is possible to visualize trees, which do not have to be positioned manually on the aerial picture. To achieve this we perform an automated tree segmentation on the fly and visualize the trees in an additional rendering pass.

4.1 Tree Segmentation

In order to visualize the trees contained within a 3D city model without requiring manual user interactions, a tree segmentation is needed. With the segmentation presented in this subsection we were able to extract the parts of the textures which correspond to deciduous trees. The idea is to simply compare the content of two registered textures showing the same area of interest during different seasons, i.e., summer and winter. As it can be seen in Figure 1, the summer texture is visually more appealing while the corresponding winter texture does not contain the treetops, since the leaves are gone in winter. Thus we can assume, that a texel t_s in the summer texture represents parts of a treetop, if the following condition is satisfied:

$$max_{channel}(t_s) = g \wedge max_{channel}(t_w) \neq g.$$

Here t_s denotes the texel in the summer texture, t_w the corresponding texel in the winter texture and $max_{channel}$ returns the color channel with the maximum intensity. Although this looks quite easy in theory, in practice several problems arise. The main problems are:

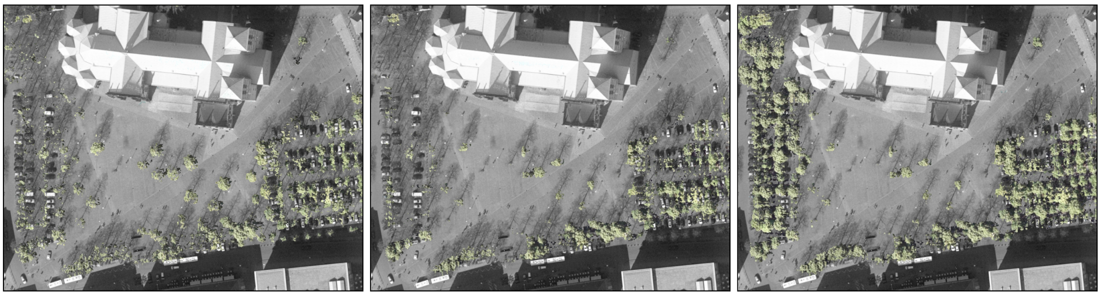
- Because lighting is present in both aerial pictures, treetops are shaded and do not contain a green hue across their whole extent.
- Since the registered aerial pictures have been acquired during different seasons and possibly during different times of the day, the lighting conditions may differ drastically.

Due to the first issue no sufficient segmentation can be obtained when using a simple pixel-based comparison to identify treetop areas (see Figure 5(a)). Thus we have extended the tree segmentation not to process on a texel t isolated from the others, but consider also the $k \times k$ neighborhood $N_k(t)$, k odd, of the current texel t. Thus a texel is identified as a treetop, if the following assertion evaluates to true:

$$|\{t' \in N_k(t_s) | max_{channel}(t') = g\}| > \frac{k^2}{2}$$
$$\wedge |\{t' \in N_k(t_w) | max_{channel}(t') \neq g\}| > \frac{k^2}{2}$$

Obviously the size k of the filter mask is dependent on the resolution of the aerial picture as well as the lighting conditions. For our aerial pictures having a resolution of $10cm \times 10cm$ we achieved a good tradeoff between segmentation results and frame rate, when using a filter covering 3×3 texels (see Figure 5(b)).

To address the second problem we have calculated a histogram equalization which can be accessed when



(a) pixel-based segmentation (b) averaged neighborhood segmentation (c) averaged neighborhood segmentation with color transformation

Figure 5: Application of different tree segmentations strategies: simple pixel-based segmentation, averaged neighborhood segmentation, and averaged neighborhood segmentation with color transformation (from left to right).

fetching the textures. This can be done in real-time and thus we can ensure that the original image data does not have to be modified as it would be necessary when applying the more time consuming lighting estimation techniques performed in a preprocessing step. Figure 5(c) shows the segmentation results we could achieve when combining the neighborhood average operator with this color transformation.

We achieved good segmentation results by using the described method. However, there are also drawbacks. One problem obviously arises when a non-treetop texel is green in the summer texture and of different color in the winter texture. This may be the case for green cars and other green objects present in the summer texture but not in the winter texture or for green objects colored differently in the winter texture due to the lighting conditions. This problem can be solved to a certain extent by increasing the size k of the neighborhood used during the color averaging process. However, increasing the neighborhood results in less performance. Another problem comes up, when coniferous trees are present. Since these trees are green all year round, we are not able to segment them. Thus in cases were a more robust segmentation is required an offline method has to be used.

4.2 Visualizing Trees

Using the segmentation technique described above, we can distinguish aerial image texels, wether they belong to tree tops or not. In the visualization process we use the winter texture, which contains no treetops, for the ground, and we send some extra geometry down the rendering pipeline in order to render the treetops extracted from the summer texture. For rendering of the ground with the winter texture we use a GPU-based implementation of the geometry clipmaps (Asirvatham and Hoppe, 2005). In the following we will describe the approach for rendering the trees.

To get a realistic appearance for the general 6 DoF case, we use a layered geometry consisting of several stacked geometry clipmaps. Each geometry clipmap is altered to a certain height, while the average altitude of all these clipmaps represents the average tree height in the geo-virtual environment.

For processing each clipmap of the clipmap stack, we can either compute the tree extraction in an initial rendering pass, or we can vary the segmentation process for each geometry clipmap. While the first approach allows higher frame rates the second approach results in more convincing visual representations. A reasonable approach for achieving realistic visualizations is to alter the size k of the neighborhood area based on the current height during the segmentation process (see Subsection 4.1). Increasing k with increasing distance from the ground results in more naturally shaped treetops. Furthermore to enhance realism we inserted a slight texture coordinate shift when rendering each geometry clipmap.

A major drawback of the described technique for rendering trees is the fact, that we only render the foliage and not the trunks of the trees. While this is irrelevant for birds-view perspectives (see Figure 6(a) and Figure 6(b)), a different perspective would make the lack of trunks obvious. Thus in cases where arbitrary perspectives are required, we combine our visualization technique with the rendering of tree trunks (see Figure 6(c)). However, positioning these trunks is a critical aspect possibly involving user interactions, although for areas, where cadastral data is available, the position can be extracted automatically.

(a) Trees visualized from top view (b) Trees visualized from top view (c) Additionally integrated trunks

Figure 6: Visualizations of the extracted trees, without and with additional trunks inserted.

5 CONCLUSION

In this paper we have presented an efficient implementation which supports the usage of multimodal clipmaps on standard graphics hardware. Based on this clipmapping implementation we have presented a new approach for estimating the clipmap center to determine the LoD for the general 3D case. Due to the efficiency of the proposed implementation we were able to develop interactive visualization techniques which improve realism as well as exploration of interactive geo-virtual environments. We have presented an interactive treetop segmentation technique, which extracts treetops from aerial images and visualize them as 3D elements.

In the future it should be investigated if more robust offline segmentation algorithms may improve the results and how some of the segmented data can be stored with the aerial pictures.

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers for their valuable comments. Furthermore the City of Münster for providing the texture data as well as the cadastral data as well as the students contributing to the 3D city visualization project.

REFERENCES

Asirvatham, A. and Hoppe, H. (2005). Terrain rendering using gpu-based geometry clipmaps. In *GPU Gems 2*. Addison-Wesley.

Brodersen, A. (2005). Real-time visualization of large textured terrains. In *GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 439–442, New York, NY, USA. ACM Press.

Castel, T., Beaudoin, A., Floury, N., Toan, T. L., Caraglio, Y., and Barczi, J. (2001). Deriving forest canopy parameters for backscatter models using the amap architectural plant model. *IEEE Transactions on Geoscience and Remote Sensing*, 39(3):571–583.

Deussen, O., Ebert, D. S., Fedkiw, R., Musgrave, F. K., Prusinkiewicz, P., Roble, D., Stam, J., and Tessendorf, J. (2004). The elements of nature: interactive and realistic techniques. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes*, page 32, New York, NY, USA. ACM Press.

Hua, W., Zhang, H., Lu, Y., Bao, H., and Peng, Q. (2004). Huge texture mapping for real-time visualization of large-scale terrain. In *VRST '04: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 154–157, New York, NY, USA. ACM Press.

Lee, K.-S., Cohen, W. B., Kennedy, R. E., Maiersperger, T. K., and Gower, S. T. (2004). Hyperspectral versus multispectral data for estimating leaf area index in four different biomes. *Remote Sensing of Environment*, 91(3-4):508–520.

Lluch, J., Camahort, E., and Vivó, R. (2003). Procedural multiresolution for plant and tree rendering. In *AFRIGRAPH '03: Proceedings of the 2nd international conference on Computer graphics, virtual Reality, visualisation and interaction in Africa*, pages 31–38, New York, NY, USA. ACM Press.

Schlerf, M. and Atzberger, C. (2006). Inversion of a forest reflectance model to estimate structural canopy variables from hyperspectral remote sensing data. *Remote Sensing of Environment*, 100(3):281–294.

Tanner, C. C., Migdal, C. J., and Jones, M. T. (1998). The clipmap: a virtual mipmap. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 151–158, New York, NY, USA. ACM Press.