FAÇADE TEXTURING FOR RENDERING 3D CITY MODELS

Martin Kada, Darko Klinec, Norbert Haala
Institute for Photogrammetry
University of Stuttgart
Geschwister-Scholl-Str. 24
Stuttgart, Germany
firstname.lastname@ifp.uni-stuttgart.de

ABSTRACT

Realistic visualizations of 3D city models require the availability of detailed geometric descriptions of the visible buildings. Especially, if virtual viewpoints from a pedestrian perspective have to be generated, texture images have to be additionally available. For this purpose geo-referenced terrestrial images have to be captured and mapped against the corresponding elements of the building model. By these means the large scale structure and the material of the building faces can be visually represented. In our approach this texture mapping is realized using a programmable graphics processing unit. In contrast to the application of standard viewers, this technique allows to model even complex geometric effects like self-occlusion or lens distortion. This allows for a very fast and flexible on-the-fly generation of façade texture using real world imagery. Our approach was implemented within a project aiming on the real-time visualization of urban landscapes, which will be discussed in the final part of the paper.

INTRODUCTION

Also as a result of the wide availability of commercial software products for the acquisition of 3D city models, the visualization of urban landscapes has been of major interest in the past years. Especially real-time applications are very popular nowadays, particularly if the data sets are streamed over the internet. Key markets for this kind of visualization are city planning and marketing, traffic management, three-dimensional car navigation, location-based services, entertainment and education. Although abstract rendering techniques have been proposed especially for planning purposes (Döllner and Walther, 2003), a photo-realistic visualization is in most cases preferred by the consumers. In addition to a correct geometric description, the structure and material of the building façades is particularly essential for a good visual impression. This type of information allows a user to orientate himself in the model especially in complex residential areas.

For an area covering reconstruction of buildings, the acquisition is generally based on measurement from aerial stereo imagery or airborne laser scanning

data. A good overview of 3D building reconstruction software is e.g. given in (Baltsavias, Grün and van Gool, 2001). The models that result from aerial data, however, consist of only few polygons that feature no detail for the building façades. (Früh and Zakhor, 2003) present a method that merges ground based and airborne laser scans and images. The additional terrestrial data naturally leads to more detailed models but also considerably increases the size of the datasets. A more efficient technique to model the façades is to extract textures from terrestrial images and place them on the coarse, polygonal models. If this is done in a manual fashion, however, the texturing of a single building is a tedious task and can easily take up to several hours. For a large number of building façades, such an approach is consequently not applicable.
In this article we present a new approach that automatically extracts façade textures from terrestrial photographs and maps them to geo-referenced 3D building models. In following sections the automatic provision of façade texture based on geo-referenced images is described. By using the functionality of 3D graphics hardware, the process of texture extraction and placement can be realized very efficiently. These approaches were realized within a project aiming on the real-time visualization of urban landscapes, which will be discussed in the final part of the paper.

MODEL BASED DETERMINATION OF EXTERIOR ORIENTATION FOR TERRESTRIAL IMAGES

Frequently, texture mapping is realised by a GUI, which allows a human operator to select corresponding primitives between the available building model and the respective terrestrial images. If a large number of buildings have to be processed, this manual process is too time consuming. In order to automate this process a co-registration of image and model data is required. In detail the 6 parameters of the exterior orientation have to be determined. Based on these parameters of central-perspective transformation we are able to link object and image space and to provide the required texture information.

Methods for Collection of Orientation Parameters

In general, the exterior orientation can be determined either by a direct or an indirect approach. Within the direct method, the parameters of the exterior orientation are measured at the time of exposure using appropriate sensors. While the position is usually determined by GPS measurement, the orientation angles can be collected by an inertial measurement unit. Such system configurations are e.g. used for direct geo-referencing of aerial camera systems (Cramer, 2003). One problem of this direct method is the sensitivity with respect to the quality of the measured orientation parameters and the calibration of the camera. Small errors of the exterior orientation can result in large discrepancies between the co-registered image and the building model. In order to extract texture information at sufficient accuracy, high quality and thus expensive sensor components would be necessary. As we want to use low cost hardware like navigation grade GPS or

digital compass, the additional use of the indirect approach will be more suitable for orientation determination.
In order to calculate the exterior orientation by the indirect method, 3D control point information is required. Still, for a task like texture mapping this information can be extracted easily from the 3D building models, which are already available. For the indirect approach corresponding features like points or lines have to be provided. Airborne applications frequently are based on the application of signalised points. In contrast, for terrestrial applications these one-dimensional primitives are frequently occluded by objects like trees, cars or pedestrians. Additional problems can result during exact identification and measurement. For these reasons, linear features are usually more suitable to provide the required image to model correspondences (Figure 1). Additionally, linear features can be provided very accurate by standard image processing software.

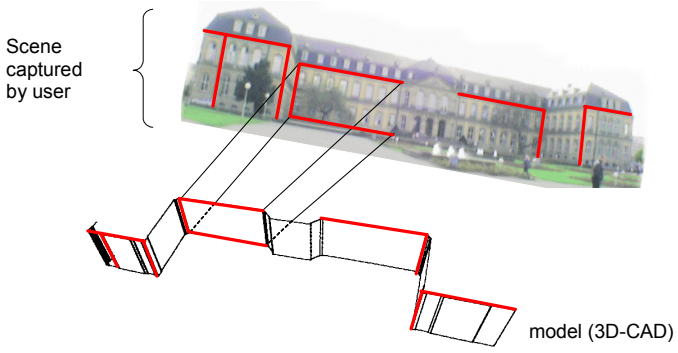

Figure 1. Assignment of linear features between image and 3D model.

Computation of Exterior Orientation Parameters

When a set of corresponding linear features is available, a modified spatial resection (M-SR) is used to compute the parameters of the exterior orientation (Klinec, 2004). Since the non linear M-SR equations are solved by a least squares approach, coarse approximation values are required. These initial values can either be directly measured by low-cost sensors, or can be provided by applying a direct linear transformation (DLT). Since the DLT equations are a linear formulation of the standard colinearity equations, a direct solution of the problem is feasible. One disadvantage of the DLT is that it is over-parameterised if the interior orientation of the camera and the parameters for lens distortion are available. For this reason, the result of the DLT has to be refined by the M-SR method for exact calculation of the exterior orientation of the terrestrial images. Our modified spatial resection algorithm was implemented in C++, additionally the Intel® Math Kernel Library was used for efficient matrix operations.

HARDWARE-BASED FAÇADE TEXTURE EXTRACTION

Once the exterior orientation and the camera parameters are available, the 3D object coordinates of the geo-referenced building model can be linked to the corresponding image coordinates using a world to image transformation (see Figure 2). Based on these correspondences, the presentation of the textured model by a standard VRML viewer is feasible. Nevertheless, the quality of visual realism is limited since these viewers only allow for simple transformations during texture mapping. As an example, complex geometric image transformations in order to model perspective rectification or lens distortion are not available. One option to solve this problem is to eliminate these effects before texture mapping by the generation of 'ideal' images, which are then used as an input for the standard viewer.

In contrast, in our approach these distortions are eliminated on-the-fly by programmable graphics hardware. By these means, problems resulting from self-occlusions can additionally be solved and multiple images can be integrated during texture mapping. Another benefit of our approach is that the façade texture is directly extracted from the original images; no intermediate images have to be generated and stored. Additionally, within the whole process image pixels are interpolated only once, which results in façade textures of higher quality.

The approach described in this article is based on technologies that can be found in today's commodity 3D graphics hardware. Graphics processing units (GPU) that are integrated in modern graphics cards are optimized for the transformation of vertices and the processing of pixel data. As they have evolved from a fixed function to a programmable pipeline design, they can now be utilized for various fields of applications. The programs that are executed on the hardware are called shaders. They can be implemented using high level programming languages like HLSL (developed by Microsoft) (Gray, 2003) or C for graphics (developed by NVIDIA) (Fernando and Kilgard, 2003). In our approach shaders are used to realize specialized projective texture lookups, depth buffer algorithms and an on-the-fly removal of lens distortions for calibrated cameras.

Texture Extraction and Placement

Our approach uses the graphics rendering pipeline of the graphics card to generate quadrilateral texture images. In general, the function of the pipeline is to render a visualization of a scene from a given viewpoint based on three-dimensional objects, textures and light sources.

Because the texture images, which are mapped against the façades during visualization, have to be represented by quadrilaterals, the polygons of the building are substituted by their bounding rectangles during the extraction process. For these bounding rectangles 3D world coordinates are available. This information is used to calculate the corresponding image pixels, which provide the required façade texture.

In more detail, the first step is to set up the graphics rendering pipeline to draw the entire target pixel buffer of the final façade texture. For this purpose, the

transformation matrices are initialized with the identity, so that drawing a unit square will render all pixels in the target buffer as wanted. As no color information is provided yet, a photograph must be assigned to the pipeline as an input texture from where to take the color information from. As mentioned above, the polygon's projected bounding box defines the pixels to be extracted from the input texture. So in addition to the vertices, the texture coordinates of the four vertices of the unit square are specified as the four-element (homogenous) world space coordinates of the bounding box. Setting the texture transformation matrix with the aforementioned transformation from world to image space concludes the initialization. During rendering, the rasterizer of the GPU linearly interpolates the four-dimensional texture coordinates across the quadrilateral. A perspective texture lookup in the pixel shader results in the perspectively correct façade texture (see Figure 2 and Figure 3).

Figure 2. Projected 3D building model overlaid on the input photograph (left) and the extracted facade textures (right).

Figure 3. Resulting 3D building model with the extracted textures placed on the façade polygons.

After the extraction, the textures need to be placed on the corresponding polygons (see Figure 3). In order to find the two-dimensional texture coordinates for the polygon vertices, a function identical to glTexGen (Shreiner, 2003) of OpenGL is used.

Image Fusion

A common problem is that parts of the building façades are not visible in the photograph due to self-occlusions. If this effect is not modeled correctly erroneous pixels are extracted from the respective texture image (see Figure 4). To avoid such artifacts, invalid pixels that belong to other polygons must be identified and marked. The color information from various positions is then combined to generate the final façade texture.

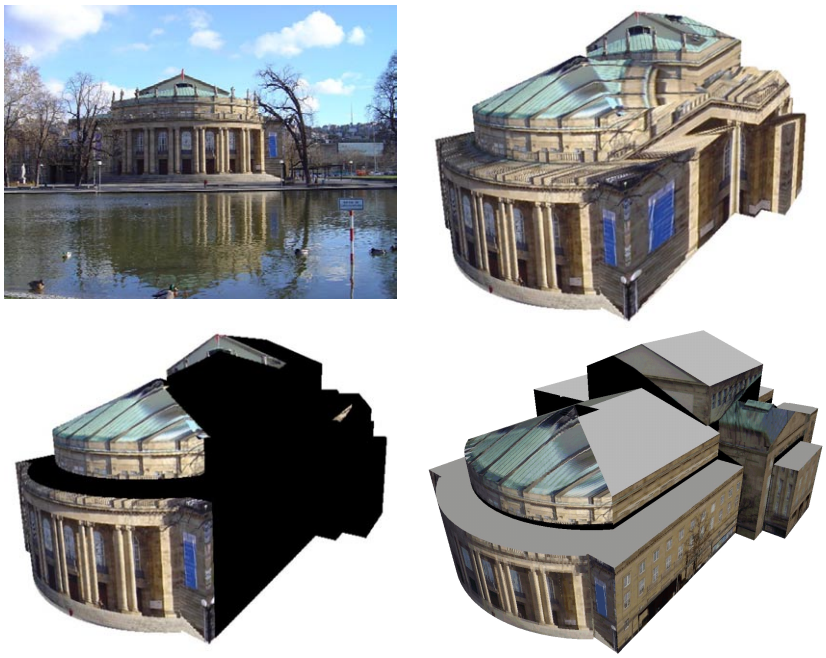

Figure 4. Original input image (top left). Automatically textured building model with occlusion culling disabled (top right) and enabled. Black pixels mark invalid texture pixels (bottom left). Texture mapping from multiple images with occlusion free pixels selected (bottom right).

By using the depth buffer algorithm, the closest polygon for each pixel in the photograph can be determined. We use a pixel shader to calculate the depth value and render it directly into a 32 bit floating-point depth texture. During texture extraction, this value is then read out in the extracting pixel shader using the same texture coordinates as for the color lookup. After the perspective divide is applied to the texture coordinates, the z-component holds the depth value for the current polygon pixel. A comparison of these two depth values then determines if the pixel in the color value belongs to the polygon. Figure 4 shows some re-

sults where occluded pixel values have been masked out. When processing more then one photograph, the final color can be merged by using the closest, non-occluded pixel in all images. Even though the approach is brute force, it is still very efficient with hardware support.

Removal of Lens Distortions

In order to link the 3D object coordinates of the building model to the terrestrial images for texture mapping, the parameters of camera calibration have to be available in addition to the exterior orientation. Especially for standard consumer type cameras, which are frequently used to collect the façade images, lens distortion has to be considered to guarantee a correct world to image transformation. Additionally, uncalibrated lens distortion can result in straight object lines which are depicted as curved lines in the texture image. In our experiments the lens distortion of the camera is described by the parameter set introduced by (Brown, 1971). These parameters are estimated based on the bundle adjustment program Australis (Fraser, 1997).
Usually lens distortion provides the transformation of pixels from the distorted to the idealized image. Because our texture extraction process requires a transformation from the idealized to the distorted image, an iterative method has to be used. This approach can be implemented based on the graphics API Direct3D 9.0 which defines dynamic flow control in Pixel Shader 3.0 (Microsoft, 2003). By these means, the lens distortion in the images can be corrected on-the-fly in the pixel shader.

Results of the Texture Extraction

The texture extraction algorithm was implemented in C++ with the graphics API Direct3D 9.0, which includes the high level shading language HLSL. The performance analysis has been conducted on a standard PC with an Intel 4 3.0 GHz Processor, 1GB of DDR-RAM and a graphics card that is based on the ATI 9800 GPU with 256 MB of graphics memory. With all features enabled, the extraction time for a model of approx. 146 polygons and 16 input images is still below one second.

REAL-TIME VISUALIZATION OF URBAN LANDSCAPES

With the techniques described in the previous sections it is possible to efficiently texture map 3D building models. The generation of urban landscapes with a large number of texture mapped building models is thus possible, but also results in new requirements for their real-time visualization. Within the project GISMO a model of the city of Stuttgart, Germany, was generated and integrated into a visualization application, which was also implemented. Since the project mainly aimed on high performance gains for flyovers, techniques like occlusion culling (Wonka and Schmalstieg, 1999), which are more efficient for walkthroughs were not integrated. Instead, the building data was pre-processed in order to optimize

the graphics rendering pipeline and a continuous level-of-detail approach was used for visualization of the underlying terrain data.

Data Acquisition

The 3D model of Stuttgart contains a digital terrain model of the city and the surrounding area of the size 50 * 50 km (see Figure 5). While the virtual view-point remains in the city area, the visualization stretches as far as the virtual horizon. The resolution of the DTM is 10 meter for the inner city region and 30 meter for the surrounding area. The corresponding aerial and satellite images have a ground pixel resolution of 0.8 and 5 meter, respectively. To speed up the rendering time, the ground textures needed to be sub-sampled to a resolution of approximately 1 meter for the city center while decreasing gradually towards the perimeter.

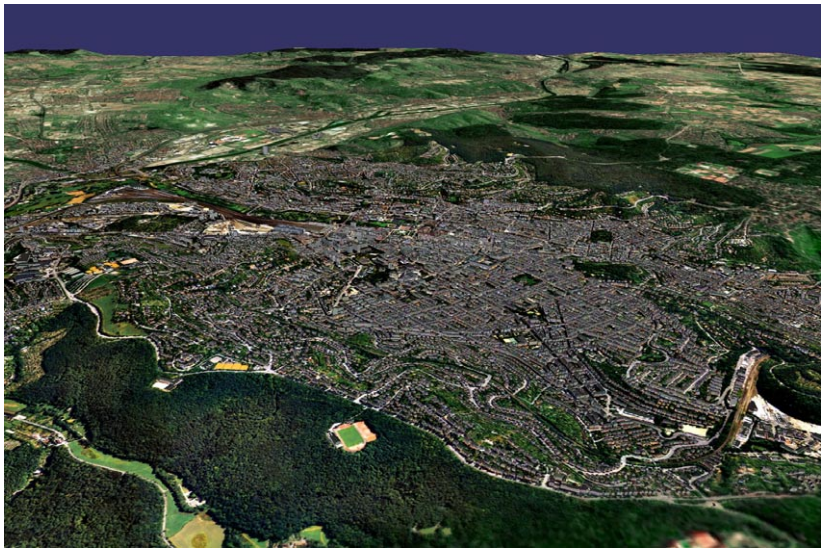

Figure 5. Overview look of the urban landscape model of Stuttgart.

The included building models are provided by the City Surveying Office of Stuttgart. They were photogrammetrically reconstructed in a semi-automatic process. For data capturing, the building ground plans from the public Automated Real Estate Map (ALK) and the 3D shapes measured from aerial images were used (Wolf, 1999). The resulting wireframe model contains the geometry of 36,000 buildings covering an area of 25 km², meaning that almost every building of the city and its suburbs is included. The overall complexity of all the building models amounts to 1.5 million triangles. In addition to the majority of relatively simple building models, some prominent buildings like the historical New Palace

of Stuttgart are represented by 3,000 (and more) triangles. To improve the visual appearance, we captured the façade textures of 1,000 buildings that are located in the main pedestrian area. Approximately 8,000 ground based close-up photographs of the building façades were taken using a standard digital camera. The textures were extracted from the images, perspectively corrected, rectified and manually mapped on the corresponding planar façade segments. We managed to process the aforementioned 1,000 buildings in roughly 30 man-months. Because of the large size of the original texture dataset, we had to down-sample the textures to a resolution of approximately 15 centimeters per pixel. Buildings with no real captured façade textures were finally colored randomly with different colors for the façade and the roof.

Visualization

For rendering digital terrain models, a continuous level-of-detail approach is generally chosen (see e.g. (Lindstrom et al., 1996)). We have used the terrain rendering library libMini which is licensed under an open source license. The approach that is realized in the library recursively generates triangle fans from a view-dependent quad-tree structured triangulation (Roettger et al., 1998). It is very easy to integrate the library in other software packages as the API is simple to use. To suppress popping artifacts that can be otherwise experienced because of changes in geometry, a technique called geomorphing is applied which slowly moves newly introduced vertices from a position on the terrain to its final position (Zach, 2002).
The 3D building models are pre-processed for visualization in order to avoid unnecessary state changes. All buildings are pre-transformed to lie in the same coordinate system and buildings without textures are grouped together to form larger data units. Primitives are rendered as indexed vertices in a triangle list. As the position of the global light source does usually not change, we omitted the normal vector for each vertex and pre-lit all vertices. The memory requirement of the vector data could be reduced this way by 42%.

Results of Real-Time Visualization

The performance analysis of the visualization application has been conducted on a standard PC equipped with a 3.0 GHz Intel Pentium 4 processor, 1 GB RAM and an ATI X800 Pro compliant graphics card. In previous work we had used the impostor technique to accelerate the rendering of the building models (Kada et al., 2003). As performance has considerably increased with the latest hardware generations, we felt that the speed-up of the impostor approach does not justify its disadvantages (especially the occasional frame rate drops) anymore. Instead we preprocessed the data so that most of the models and textures were guaranteed to remain on the graphics memory. The time extensive paging of data in and out of dedicated graphics memory was consequently minimized. Running at a screen resolution of 1280*1024 the application almost reaches real-time performance, meaning that approximately 15 to 20 frames per second are rendered.

Figure 6. 3D city model rendered in the real-time visualization environment.

CONCLUSION

Meanwhile, the availability of modern graphic cards allows high-end visualization using reasonably priced standard hardware. Also due to this fact, real-time visualization of virtual 3D city models is now integrated in a growing number of applications. In addition to the great computational power of these systems, programmable graphics hardware also allows the direct implementation of complex algorithms. Within the paper the texture mapping of building façades using directly geo-referenced terrestrial images was implemented in programmable graphics hardware. By these means 'photogrammetric' tasks like the transformation and mapping of world to image coordinates were directly integrated in the graphics rendering pipeline to allow for a time efficient solution. This demonstrates the potential of integrating techniques from computer graphic and photogrammetry for time critical applications like the link of virtual 3D models and real imagery for real-time visualization.

ACKNOWLEDGEMENTS

The research described in this paper is founded by "Deutsche Forschungsgemeinschaft" (DFG – German Research Foundation). The research takes place within the Center of Excellence No. 627 "NEXUS – SPATIAL WORLD MODELS

FOR MOBILE CONTEXT-AWARE APPLICATIONS" at the University of Stuttgart. The geometry of the building models is provided by Stadtmessungsamt Stuttgart (City Surveying Office of Stuttgart).

REFERENCES

Baltsavias, E., Grün, A. and van Gool, L. (2001). Automatic Extraction of Mad-Made Objects from Aerial and Space Images (III). Swets & Zeitlinger B.V., Lisse, The Netherlands.

Brown, D.C., 1971. Close-Range Camera Calibration. *Photogrammetric Engineering*, 37 (8), pp. 855-866.

Cramer, M. (2003) Integrated GPS/inertial and digital aerial triangulation - recent test results. In: Fritsch (Ed.) *Photogrammetric Week '03*, Wichmann Verlag, Heidelberg, pp. 161-172.

Döllner, J. and Walther, M. (2003). Real-Time Expressive Rendering of City Models. In: *Proceedings IEEE 2003 Information Visualization*. London. pp. 245-250.

Fernando, R. and Kilgard, M. (2003). *The Cg Tutorial*. Addison-Wesley.

Fraser, C.S. (1997). Digital Camera Self-Calibration. In: *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 52, pp. 149-159.

Früh, C. and Zakhor, A. (2003). Constructing 3D City Models by Merging Aerial and Ground Views. *IEEE Computer Graphics and Applications*, Vol. 23 No. 6, pp. 52-61.

Gray, K. (2003). *The Microsoft DirectX 9 Programmable Graphics Pipeline*. Microsoft Press.

Kada, M., Roettger, S., Weiss, K., Erlt, T. and Fritsch, D. (2003). Real-Time Visualisation of Urban Landscapes Using Open-Source Software. In: *Proceedings of the ACRS 2003 ISRS*, 24th Asian Conference on Remote Sensing& 2003 International Symposium on Remote Sensing, Busan, Korea.

Klinec, D. (2004). A model based approach for orientation in urban environments. In*: IAPRS*, Vol. 35, Part B, pp. 903-908.

Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L.F., Faust, N. and Turner, G. (1996). Real-Time, Continuous Level of Detail Rendering of Height Fields. In: *Proceedings of SIGGRAPH '96*, pp. 109-118.

Microsoft, 2003. DirectX Documentation for C++. Microsoft DirectX 9.0 SDK. http://msdn.microsoft.com/library/default.asp?url=/downloads/list/directx.asp

Roettger, S., Heidrich, W. Slusallek, Ph and Seidel, H.-P. (1998). Real-Time Generation of Continuous Levels of Detail for Height Fields. In: *Proceedings WSCG '98*, pp. 315-322.

Roettger, S. (2004). libMini Terrain Rendering Library. http://wwwvis.informatik.uni-stuttgart.de/~roettger.

Shreiner, D., Woo, M. and Neider, J. (2003). *OpenGL Programming Guide* (Version 1.4). Addison-Wesley.

Wolf, M. (1999). Photogrammetric Data Capture and Calculation for 3D City Models. In: *Photogrammetric Week '99*, pp. 305-312.

Wonka, P. and Schmalstieg, D. (1999). Occluder Shadows for Fast Walk-throughs of Urban Environments. In: *Proceeding Eurographics '99*, pp. 51-60.

Zach, C. (2002). Integration of Geomorphing into Level of Detail Management for Realtime Rendering. Technical Report, VRVis Research Center, University of Graz.