Photogrammetric Week '03 Dieter Fritsch (Ed.) Wichmann Verlag, Heidelberg, 2003

3D Building Generalisation and Visualisation

MARTIN KADA, Institute for Photogrammetry (ifp), University of Stuttgart

ABSTRACT

The advances in the automatic acquisition of 3D geospatial data has lead to large-scale urban landscape models that can easily consist nowadays of thousands of building objects. For a photo-realistic visualisation, each building model has a number of highly detailed façade textures associated with. Because of the vast amount of objects and façade images, a brute-force rendering approach is not feasible for such scenes even on high-performance visualisation systems.
In this paper, two methods are presented to improve the rendering performance of 3D city models. The first approach is a generalisation algorithm that automatically generates simplified versions of 3D building models that can then be used in level of detail structures. Compared to already known surface simplification algorithms from the field of computer graphics, the presented solution yields more sophisticated building models by combining least squares adjustment theory with an elaborate set of surface classification und simplification operations. The concept allows the integration of surface regularities into the building models which are important for the visual impression. These regularities are stringently preserved over the course of the generalisation process.
The second approach is an image caching technique that accelerates the rendering process by reusing parts of previous rendered images in the current frame. Using this so called impostors technique, real-time rendering of large-scale urban landscape models is possible nowadays even on low-cost PC hardware equipped with commodity 3D graphics accelerators.

1. INTRODUCTION

The development of tools for the efficient collection of 3D urban landscapes has been a topic of intense research for the past years. In addition to Digital Height Models and 3D data representing streets and urban vegetation, building models are the most important part thereof. Meanwhile, a number of algorithms based on 3D measurement from aerial stereo imagery or airborne laser scanner data are available for automatic and semi-automatic collection of 3D building models. A good overview on the current state-of-the-art of experimental systems and commercial software packages is for example given in (E. Baltsavias, A. Grün and L. van Gool, 2001). Almost all of these systems describe the reconstructed building by general polyhedrons, since a building representation by planar faces and straight edges is feasible for most cases. As an example, Figure 1 shows a 3D model of Stuttgart reconstructed by the approach of (M. Wolf, 1999).
Originally, simulations for the propagation of electromagnetic waves used for the planning of antenna locations were the major application areas for 3D building models. Meanwhile visualization in the context of three-dimensional car navigation, urban planning, virtual tourism information systems, emergency response and entertainment has become the key market for that type of data.
For an interactive visualisation application, the image must be updated between 15 to 25 times per second in order to achieve a smooth animation. Large-scale urban landscape models can easily include thousands of quite complex building objects and a vast amount of façade images. A brute force visualisation approach is not feasible for this kind of data even on high performance graphics systems. Occlusion culling mechanisms as described in (P. Wonka, D. Schmalstieg, 1999) and (P. Wonka, M. Wimmer, F. Silion, 2001) are very efficient for walkthroughs, but do not yield high performance gain for flyovers and are therefore not generally applicable.
The level of detail technique accelerates the rendering of each single image frame by using versions with varying complexity for each building model depending on the distance of the object to the

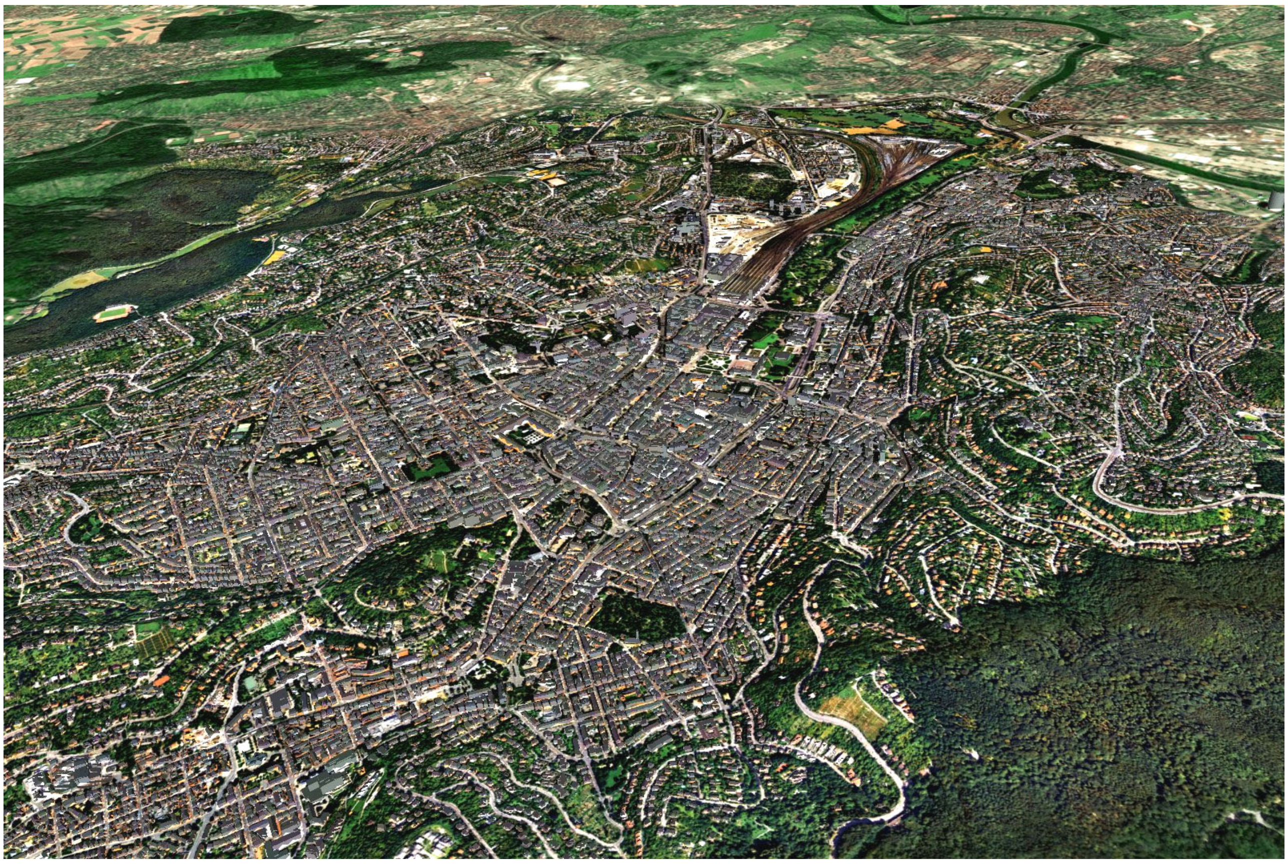

Figure 1: 3D real-time visualisation of the Stuttgart city model showing over 36.000 buildings.

viewer. As the user moves around in the scene, the complexity of the scene is dynamically adjusted, so that highly detailed models are used for close by objects and lower detailed models for objects located further away. The generalisation algorithm described within this article presents an approach to automatically generate less detailed versions of 3D building models that can then be used in the aforementioned level of detail structures.
The impostor technique as described e.g. in (G. Schaufler, 1995) and (F. Sillion, G. Drettakis, B. Bodelet, 1997) takes advantage of the coherence between consecutive images in an animation sequence. The approach is an image caching technique that accelerates the rendering process by reusing parts of previous rendered images in the current frame. The reused parts represent one or more objects that are located further away from the viewer, as their appearance in the final image does not change very much.

2. GENERALISATION ALGORITHM OUTLINE

The presented generalisation algorithm is designed for polyhedral three-dimensional building models. Such models consist of a set of vertices and a set of polygonal faces. Each face may additionally contain a number of interior points, determining the parameters of the associated planar surfaces. These interior points are provided during data collection, e.g. from stereo measurements, or result from vertices that are removed during the geometric simplification of the building model. The algorithm uses these interior points in the least squares adjustment to resolve the new coordinates of vertices after each generalisation step. This approach ensures a minimum deviation

of the generalised building model to every vertex of the original model and thus to its original shape.
Our algorithm is based on the fact that most walls are oriented in parallel to the principal axes of the building, which are again often rectangular. It can therefore be assumed that the faces of a building model are usually coplanar, parallel or rectangular to other faces in the same model. A generalisation must preserve these properties as correctly as possible. For this reason, the presented algorithm considers the aforementioned properties between faces as constraints during the simplification process. As this information is usually not explicitly available for a building model, the first step in the generalisation algorithm is to create the so-called constraint building model, which is basically the polygonal building model enriched by a set of constraints.
Following the generation of the constraint building model, the geometry of the model is iteratively simplified. First, a feature detection algorithm searches for features like extrusions, intrusions, notches, tips etc. and evaluates their significance to the overall appearance of the model. A feature removal step next eliminates the features of least importance, i.e., which only slightly influence the silhouette of the building. The feature removal step not only alters the geometry of the constraint building model, but also the constraints that are affected by it. This is important, as constraints become obsolete through the process of feature removal. Vertices that are removed from the geometry, however, are not just discarded from the model, but the algorithm stores their coordinates as additional interior points as mentioned in the beginning of this section. The last step of the algorithm uses least squares adjustment to find a new position for every vertex in the constraint building model in order to fulfil all its constraints.

3. CONSTRAINT BUILDING MODEL

The initial step of the generalisation algorithm is to find the regularities of the building model that need to be preserved in subsequent stages. These regularities are the coplanarity, parallelism and rectangularity of faces. The simplification step is designed to avoid violating constraints until they become obsolete. Simplification operations that are carried out on the building model also aim on preserving the coordinates of affected vertices and rely on the adjustment stage to determine their final position. Thus, the quality of the final, generalised building model directly depends on the quality of the constraints that are stored within the constraint building model.
For each planar surface, an unambiguous plane can be computed using the coordinates of the vertices and inner points. The correlation of the building faces can then be automatically determined using angles between the normal vectors, the distances between faces and a set of tolerance values.

Coplanarity: Two faces are assumed to be coplanar if the angle between the normal vectors is close to 0° and 180° and the difference of the absolute values of the distances lies under a given threshold.

Parallelism: Two faces are assumed to be parallel if the angle between their normal vectors is close to 0° or 180°.

Rectangularity: Two faces are assumed to be rectangular if the angle between their normal vectors is close to 90°.

It is the author's belief that not every constraint can be found by an automatic approach. Dependent on the quality of the input model, a number of constraints will almost always be missed due to errors introduced in the generation of the model. Those absent constraints might reduce the quality of the final model if missed in high quantities. An application should therefore offer the possibility to identify and insert more constraints into the constraint building model in a semi-automatic fashion to work around those errors and to improve the overall quality of the final building model. A semi-automatic tool also helps surveying the effects of certain constraints on the generalisation process by manually adding or removing those constraints.

4. FEATURE DETECTION AND REMOVAL

In order to simplify the geometry of a building model, it is not sufficient to just remove arbitrary vertices or edges. Even if the introduced geometric error is small, the symmetry of the building model will irretrievably get disturbed. It is thus necessary to take notice to the regularities of the model during its simplification. Our feature detection and removal algorithm for generalisation allows the use of a manifold set of surface simplification operators; each designed to remove one specific class of feature types. In contrast to the rather simple operators used in traditional surface simplification algorithms, our operators remove entire features in one continuous process, while preserving the integrity of the remaining parts of the building model. We distinguish between three classes of features types: the extrusion, the notch and the tip (Figure 2).

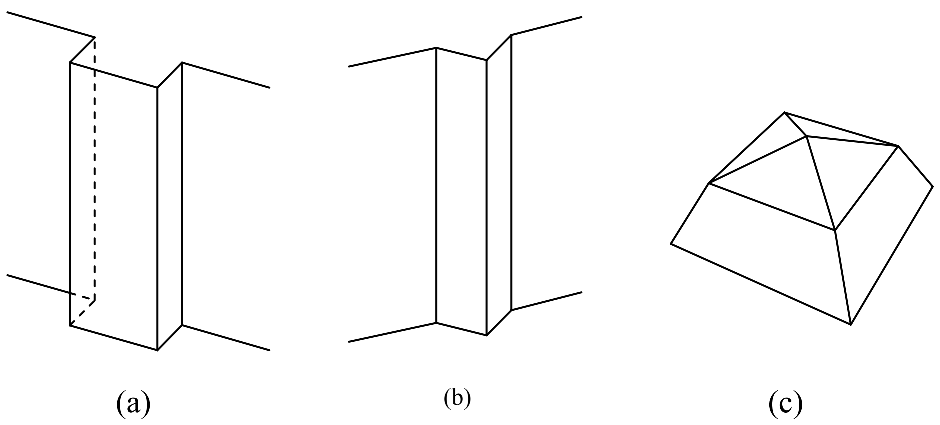

Figure 2: Feature detection distinguishes between faces, edges and vertex-based features: e.g. (a) extrusion, (b) notch and (c) tip.

The presented algorithm detects features by looping through all primitives of the building model and by testing them against the feature types of the particular feature class. Once the algorithm detects a feature, its impact to the appearance of the silhouette of the building model is evaluated. This can be a very complicated process as small features may be important due to their semantic meaning. At this point, we use a simple metric to measure a value, which corresponds to the maximum distance moved by a vertex during the removal of the feature.
After feature detection is completed, the algorithm removes the features of lowest importance to simplify the geometry of the building model. In our example, an extrusion is removed by using a combination of edge collapse (see e.g. M. Garland, P. Heckbert, 1997) and edge foreshortening operations (Figure 3). Then, the algorithm checks the validity of constraints between affected faces and updates them according to their new condition.

5. LEAST SQUARES ADJUSTMENT

By feature removal, parts of the object are detected and completely eliminated from the dataset. All original points, however, should still determine the optimal shape of the reduced model, even

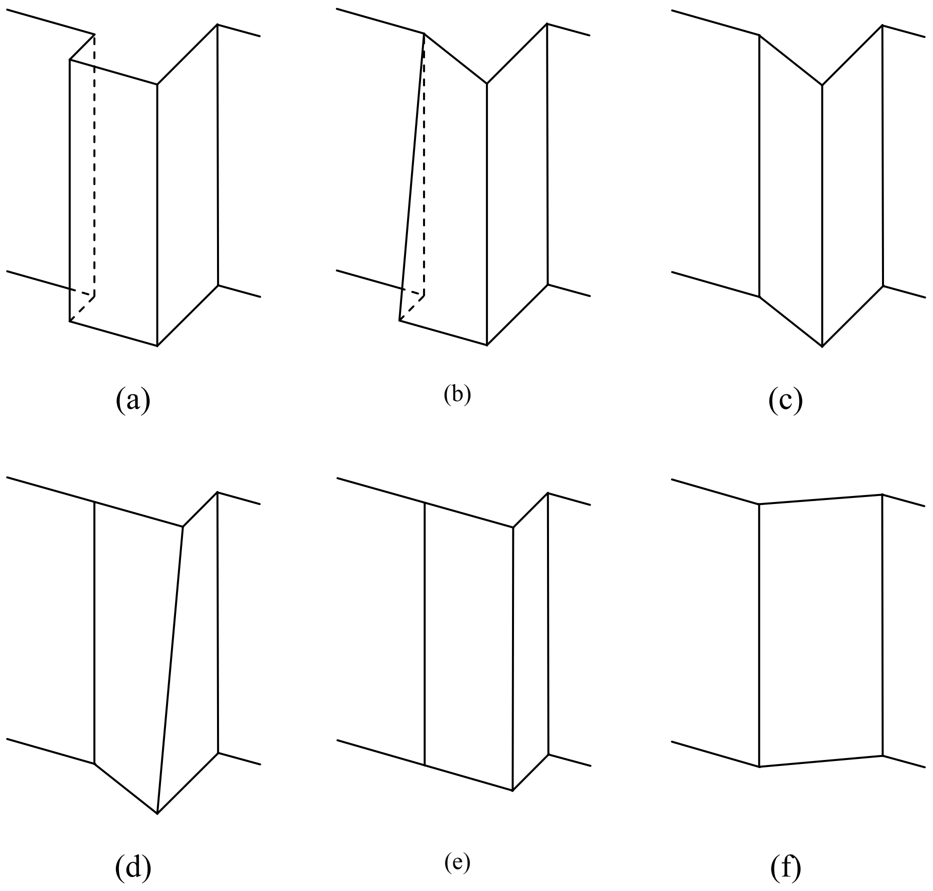

Figure 3: (a-e) Removal of an extrusion: The endpoints of the short edges are collapsed into their base vertices (b+c), whereas the longer edges are foreshortened by the same length (d+e). Collapsing all edges results in the removal of too much geometry (f).

though the number of planar faces is reduced by the preceding step. In order to resolve the final shape of the simplified model, a least squares adjustment is applied using the available constraints between the remaining faces as well as the points of the original model.
The new positions of the remaining vertices of the model are determined by the intersection of three or more faces. In order to provide a complete solution, not only the planar surfaces, but also their points of intersection are integrated into the adjustment. This approach is additionally motivated by the fact, that the topological information about the intersection of the planar surfaces is not yet used. If this information is ignored, four or more planar surfaces are not guaranteed to intersect in one unique point after generalisation. Using a different weight value for each type of constraint, helps to exert influence on the adjustment, e.g. to favour unique intersection points over parallel planar surfaces.

6. IMPOSTORS

Impostors are an image-based rendering technique. Like a billboard, an impostor replaces a complex object by an image that is projected on a transparent quadrilateral. A common example for the use of billboards is the visualisation of trees. Provided that the viewer stays close to the ground level, the image of a tree is a good approximation of the real geometry for all points of view. As the viewer moves around the scene, the quadrilateral is rotated so that the image always faces forward.

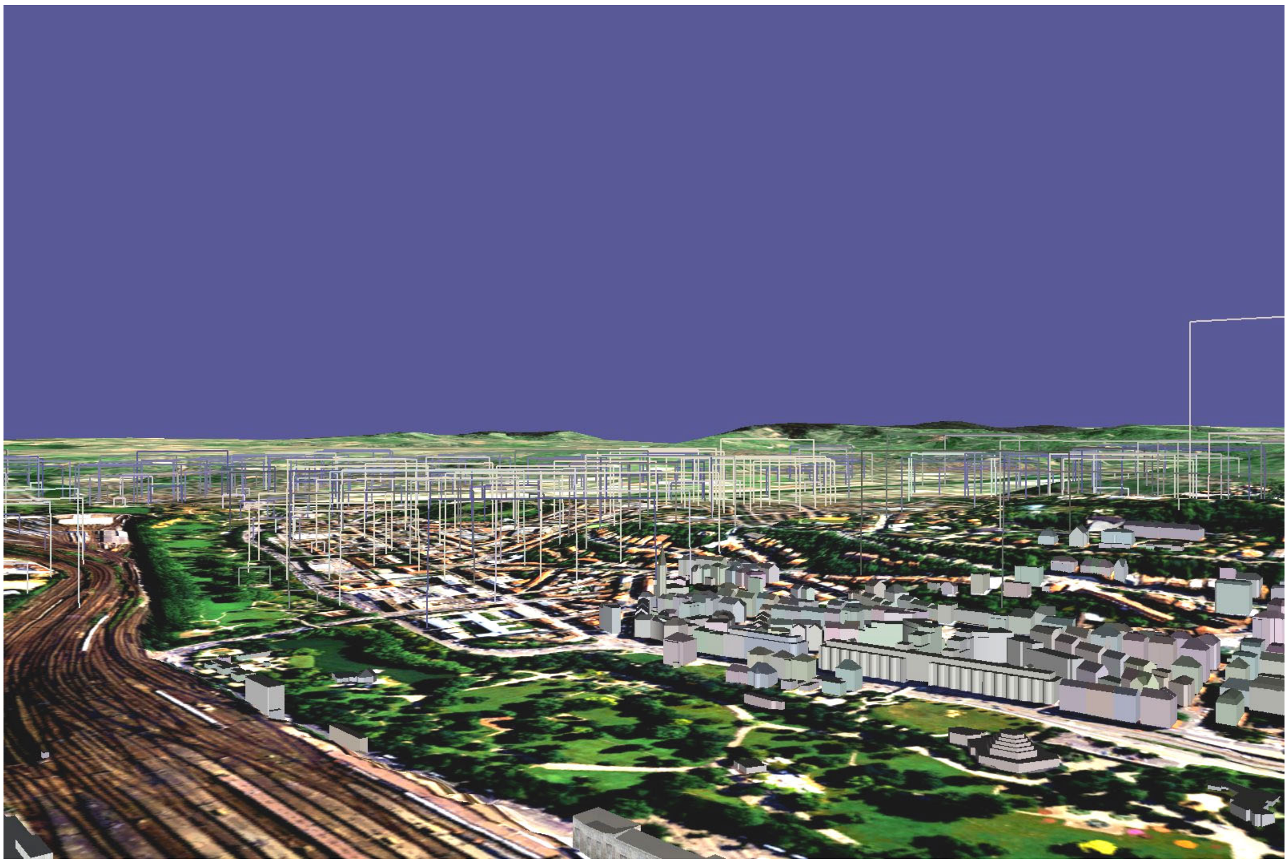

Figure 4: Building objects that are located beyond a distance threshold are rendered as impostors as depicted by the quadrilateral impostor frames.

Because billboard images are created a priori and are therefore static, this technique can only be used for objects that look similar under rotation. In contrast to that, rendering the objects themselves for the current point of view dynamically generates the images of impostors. If consecutive viewpoints are close together, the impostor images of slowly moving objects that are located far from the viewer do not change notably with every frame. From this it follows that impostor images can be reused for several frames and therefore speed up the overall rendering process.

A 3D city model consists of vast amounts of building objects. In an urban scene, these innately static objects extend over a large area so that only a fraction of the objects is actually close to the viewer. It can be assumed that in this context the majority of the building objects is located far enough from the viewer to cause few image updates and can therefore be visualized efficiently by impostors. A user given distance threshold determines if objects are rendered traditionally or with the impostor technique. The use of impostors results in a texture memory overhead, however, because the impostor images additionally occupy valuable texture space on the graphics hardware. To limit the additional memory usage, impostors must consequently not replace single building objects, but rather several buildings that are located close together.

We evaluated the impostor approach using the implementation of the Open Scene Graph (OSG), which is a cross-platform 3D graphics library for real-time visualisation that makes excellent use of PC graphics accelerators (OSG 2003). To arrange the building data in the scene graph, a very simple approach was used: the area is divided into a regular 2D grid and building objects whose centroids are located in the same cells are grouped together. We also did some testing with hierarchically organized impostors, but did not find them to be superior in our context. The additional hierarchy levels of the impostors noticeably reduced the image quality due to multiple filtering and the texture memory increased even further.

7. RESULTS

The generalisation algorithm described in the paper has been implemented and tested on selected building objects of a 3D city dataset. In order to measure the complexity of each model, we used the number of triangles gained by triangulating the planar surfaces. The algorithm showed promising results on both complex and simple models. The complexity of the building models could in many cases be reduced by over 30%, in some cases, where the model exhibited a lot of extrusions, even by 50%. The model of the New Palace of Stuttgart (Figure 5), e.g., comprises of 721 planar surfaces, that make up a total number of 2730 triangles.

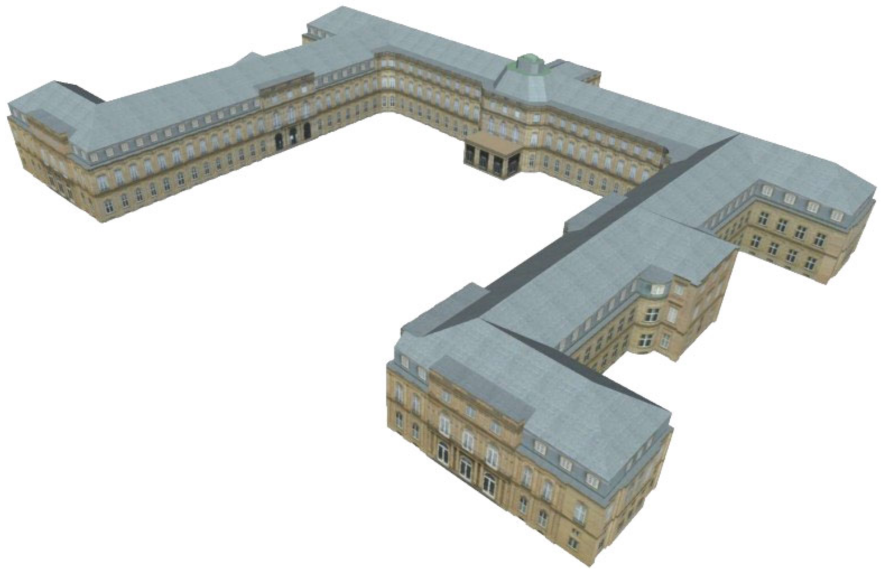
Figure 5: The New Palace of Stuttgart is used to show the results of our generalization algorithm.

Our generalisation approach was able to detect 110 extrusions using three iterations. After removal of the extrusions, the model only comprised of 1837 triangles. The results are demonstrated in Figure 6 to Figure9. Figure 6 shows part of the original model as it was captured from stereo imagery and an existing outline from the public Automated Real Estate Map, respectively. Figure 7 shows the result of the generalisation process. As it is visible, parallelism and rectangularity have been preserved for the remaining faces. Using textured models, as it is depicted in Figure 8 and Figure 9, this amount of detail is sufficient for visualisation in most cases.
For the evaluation of the impostor approach, we used a data set representing the city of Stuttgart and the surrounding area of the size 50x50 km. It includes a 3D city model provided by the City Surveying Office of Stuttgart, a digital terrain model and the corresponding aerial and satellite images. The area that is covered is chosen so that the visualisation stretches as far as the virtual horizon. The wireframe model of the buildings contain the geometry of 36.000 buildings covering an area of 25 km^2 meaning that almost every building of the city and its suburbs is included (also compare Figure 1 and Figure 10). The overall complexity of the model amounts to 1.5 million triangles. To improve the visual appearance, the façade textures of over 500 buildings that are

located in the main pedestrian area were captured. Buildings with no real captured façade textures were finally coloured randomly with different colours for the façade and the roof.

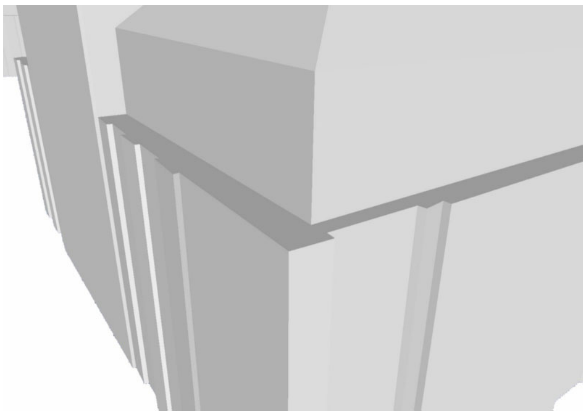
Figure 6: Part of the original building model.

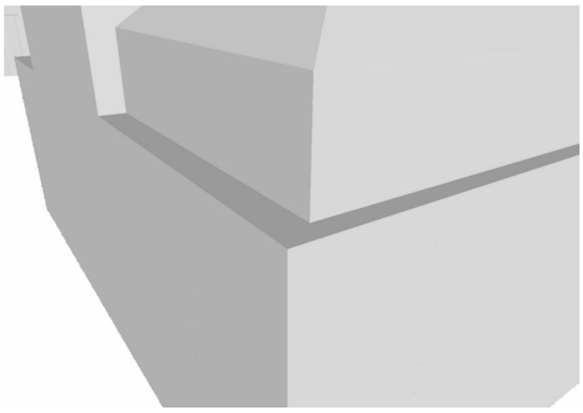
Figure 7: Part of the simplified building model.

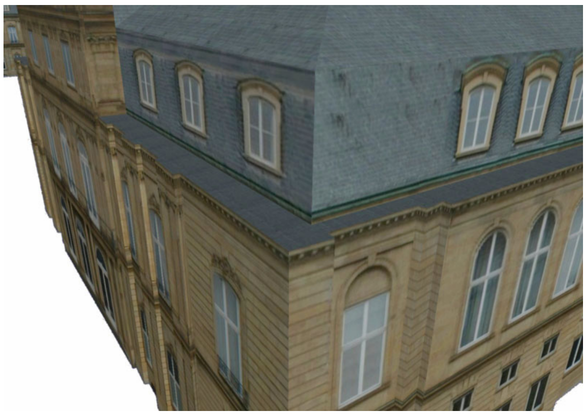
Figure 8: Part of the original building model (textured).

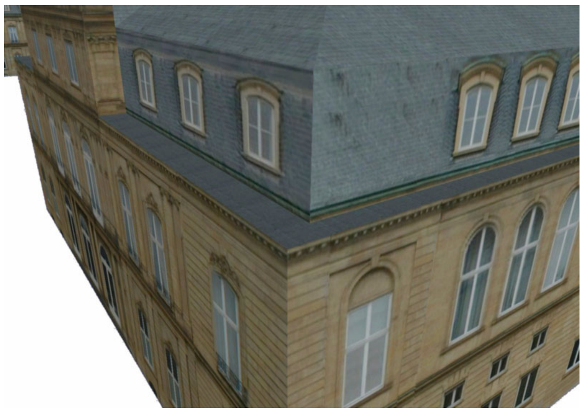
Figure 9: Part of the simplified building model (textured).

The experimental results have been measured on a standard PC equipped with a 2.0 GHz Intel Pentium 4 processor, 512 MB of memory and an NVIDIA GeForce4 Ti4200 graphics accelerator with 128 MB of graphics memory.
Without impostors we achieved a frame rate of 2 images per second for the entire data set including the rendering of the terrain. Enabling impostors, the best results were achieved when the buildings were organized in a regular layout of 60x60 impostors. Enabling impostors with a distance threshold of 1 km boosted the performance to 11 frames per second on the average, but the recomputation load for the impostors was still quite high in some cases, which led to occasional frame drops. Increasing the impostor distance threshold to 1.5 km removed most of the frame drops, but since impostors replaced fewer buildings the frame rate dropped to approximately 9 fps.

8. FUTURE WORK

Both techniques, generalisation and impostors, have proven to significantly improve the rendering performance for large-scale urban landscapes. As the general algorithm design proved to be correct, our future work in this area mainly consists of defining more features types that can be detected and removed. Especially features that are based on edges and vertices have not yet been evaluated. Also both techniques still need to be incorporated into one single visualisation system to evaluate the performance gain by using both techniques. We also plan to include vegetation and dynamic objects (cars, pedestrians) into our landscape model to further increase the realism.

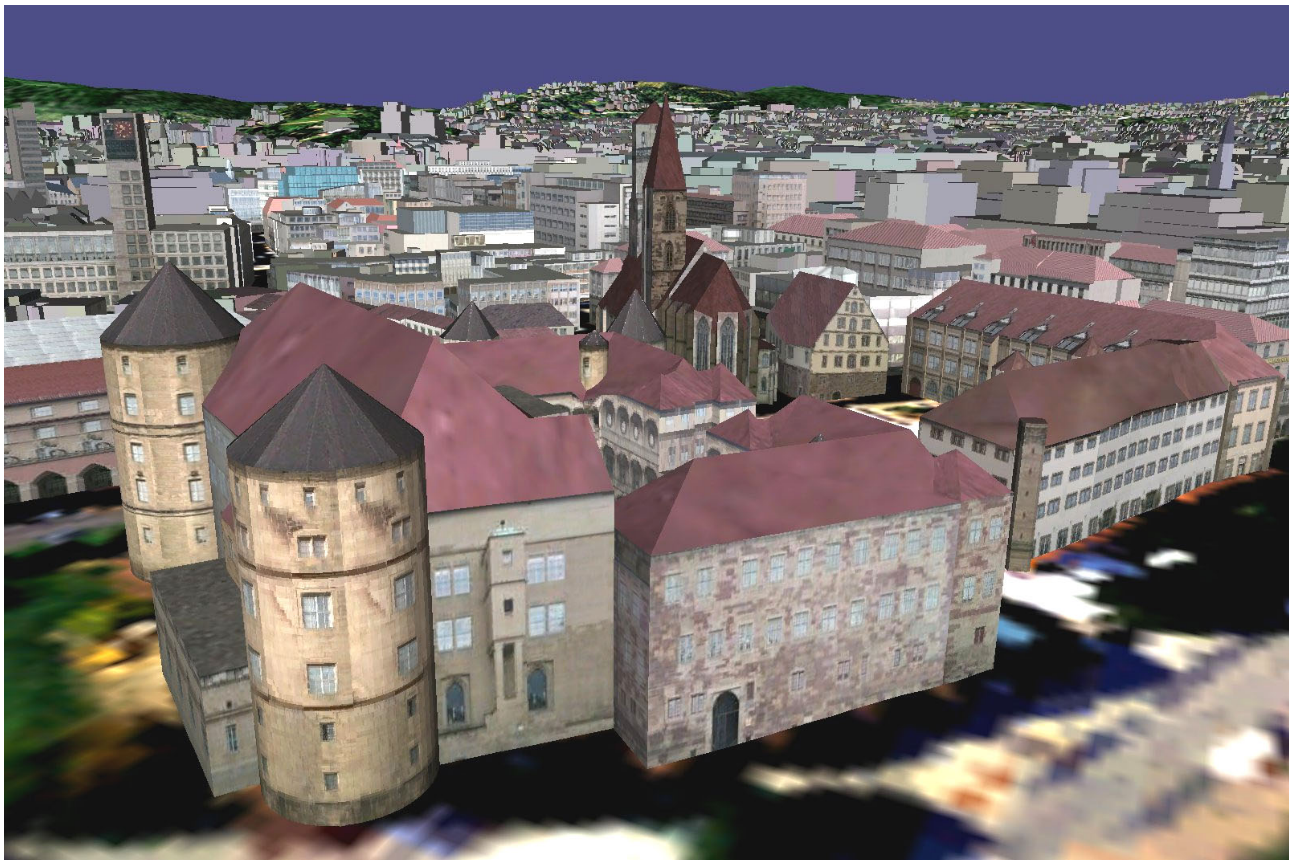

Figure 10: Close-up view of the 3D city model of Stuttgart.

9. REFERENCES

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

Garland, M., Heckbert, P., 1997: Surface Simplification using Quadric Error Metrics. In: Proceedings of ACM SIGGRAPH 97, pp.206-216, 1997.

OSG 2003: OpenSceneGraph. www.openscenegraph.org, 2003.

Schaufler, G., 1995: Dynamically Generated Impostors. In: Proc: GI Workshop on Modeling, Virtual Worlds, and Distributed Graphics ’95, pp. 129-136, 1995.

Silion, F., Drettakis, G., Bodelet, B., 1997: Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery. Computer Graphics Forum, Proc. Eurographics ’97, 16(3): pp. 207-218, 1997.

Wolf, M., 1999: Photogrammetric Data Capture and Calculation for 3D City Models. In: Photogrammetric Week ‘99, Eds. D. Fritsch, R. Spiller, Wichmann Verlag, Heidelberg, pp. 305-312, 1999.

Wonka, P., Schmalstieg, D., 1999: Occluder Shadows for Fast Walkthroughs of Urban Environments. In: Proc. Eurographics ’99, pp. 51-60, 1999.

Wonka, P., Wimmer, M., Silion, F., 2001: Instant Visibility. In: Proceedings Eurographics ’01, pp. 411-421, 2001.