Virtual Habitat: models of the urban outdoors

K. Karner, J. Bauer & A. Klaus
VRVis Research Center, Graz, Austria

F. Leberl & M. Grabner
Institute for Computer Graphics and Vision, Graz University of Technology, Austria

ABSTRACT: City models with building exteriors, some interior spaces, vegetation, addresses and phototexture are of growing interest. Our work is the result of an Austrian national research program within the framework of Virtual Reality and Visualization. We describe a data collection system for buildings, a data modeling and visualization component.

1 INTRODUCTION

We report on a framework for digitally modeling and reconstructing the urban outdoors with high efficiency and at low cost, yet without compromising the detail, accuracy and photo-realism of the result. After many years of demonstrating by various band-aid approaches how such cyber-cities can be created and used, we have now embarked on a system development to do this routinely, robustly, over large urban areas and with a minimum in manual labor.

The geometric accuracy specification of the approach is at $\pm$1cm, the detail is to be resolved at 30cm. Photo-texture is entering into the system at a resolution of at least 2cm. For a medium size city such as Graz with 40,000 buildings and an even larger number of trees, the required data set consists of about 250 terabytes of data.

1.1 *Data creation*

A city scanner is being built and calibrated to efficiently collect terrestrial sensor data with cameras, positioning devices and laser based distance measurement tools, applying them kinematically over large urban areas. The terrestrial data are being processed semi-automatically, but with a minimum of manual labor. The results are being merged with aerial photogrammetric geometry, attribute and texture data as well as with traditional 2D-GIS sources and their attributes to achieve a detailed description of the urban environment.

1.2 *Data management*

The conversion of the input data to a Virtual Habitat requires a second component to organize, manage, store, retrieve and on-line interactively, photo-realistically visualize the data for quality control and various applications. This need in turn results in a new data base structure for the preservation of the relationships among various data sources and data formats, such as GIS data, aerial and terrestrial images and distance data, and descriptive information about objects.

1.3 *Visualization*

The real-time visualization of large virtual habitats is based on the Augmented Reality paradigm and employs a combination of recent rendering techniques. Our report includes a demonstration of the current capability and an outlook to the development over the next 3 years, for which an Austrian national research program has been funded.

2 THE CITY SCANNER

2.1 *Hardware concept*

Our long term approach focuses on a cost effective way to reconstruct complete city models. Thus, reliability and completeness are very important to avoid the necessity of a repeated data acquisition. Further requirements are an absolute accuracy in position of ±20cm, a relative accuracy of ±1cm and a prototype system which is operable by a non-expert.

As it can be seen in Figure 1, the input data in our approach comes from laser scanning devices, optical sensors (area based digital camera), and 3D control information from either aerial surveying, terrestrial surveying (using a total station) or dGPS-INS systems. We want to minimize our research efforts in the area of aerial surveying. Aerial photogrammetry is a well-known discipline and progress towards solutions for automatic DEM creation has been made. Aerial based laser scanning is becoming very popular and some promising results were shown by several companies and research institutes at the 2000 ISPRS Congress. We focus on the integration and use of existing data, which is often already available for large and mid-sized industrial cities.

2.2 *Software concept*

Based on the data flow outlined in Figure 1 various software tools are necessary to reconstruct a virtual model from the input data of the real world city. The input data consist of 3D point clouds from a laser scanning device, terrestrial images captured by an area based digital camera and control information to get a geo-referenced 3D model.

- Camera calibration
  The process of camera calibration determines the internal parameters of a camera, that are: lens distortion, the position of the principal point, and the focal length. The method used in our experiments is described by Zhang (2000) and Heikkilä (2000). It requires a few (at least two) images of a planar target taken from different orientations to calculate the parameters. Once the parameters are known, all new images can be resampled to undistorted images. The calibration process must be repeated from time to time in order to cope with mechanical changes of the camera during operation. Figure 2 shows (a) the original image and (b) the resulting image after removing the distortion.

- Laser scanner to camera calibration
  Our approach is based on the fusion of different imaging sensors (2D and 3D sensors). Thus, we have to calibrate the geometrical errors of these sensors, take into account that the projection centers are at different positions and that the sensor models may vary (area based versus line based sensors). In our calibration procedure we use a photogrammetric calibration target augmented with targets for a laser scanner system.

- Radiometric calibration and feature extraction
  A prerequisite for the segmentation of and the modeling from images is the radiometric calibration and the calculation of features like lines and arcs. The radiometric calibration of the camera is done using an algorithm based on work described by Debevec & Malik (1997).

  The extraction of geometrical features from images is a crucial step in our workflow. In order to reach to desired precision the features must be determined with subpixel accuracy. The features are extracted in two steps: first contour chains are extracted with the method described by Rothwell et al. (1995). In the next step lines and arcs are extracted using the contour chains from the previous step. The 3D locations of features are calculated by a method described by Schmid & Zisserman (2000). Figure 3 shows the results of the contour line extraction and the 3D line modeling.

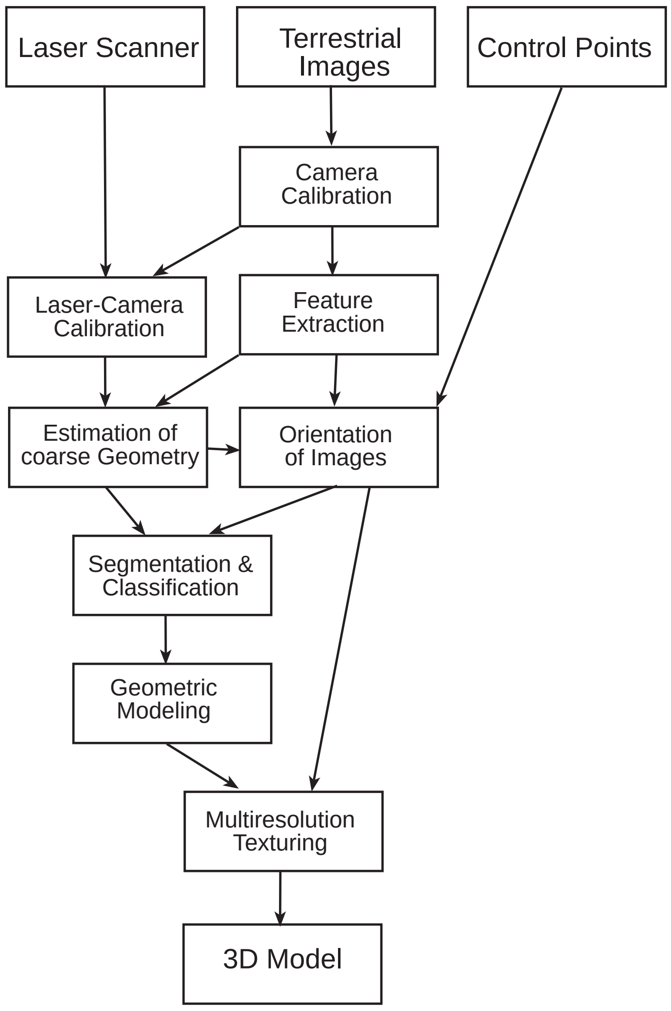
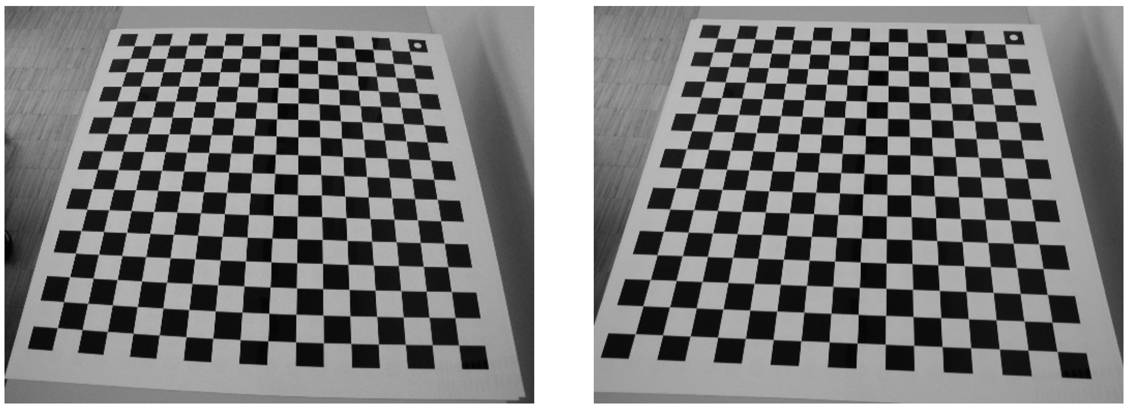


Figure 1. Data flow in the prototype system from the sensors to the 3D model.



(a) original image

(b) undistorted image

Figure 2. Camera calibration. Note the geometric distortion in the left image (a).

(a) View1 (b) View2

Figure 3. Two views of 3D lines which are extracted and modeled automatically from 3 images.

- Estimation of a coarse geometric description
  Using the 3D imaging data and the features from the 2D images, we are able to get a rough estimate of the facades in the local camera coordinate system of our prototype.

- Orientation of images
  While classical algorithms for the estimation of the relative orientation rely on the intersection of rays (minimization of point to epipolar line distances) only, we use the additional information of camera to point distances. Using the rough geometric model the distinction between a change in intensity in the images resulting from geometric or radiometric invariances is possible. This knowledge will result in a more robust relative orientation with known scale. Adding (ground) control points the coordinate system can be aligned to a geo-referenced coordinate system.

- Geometric modeling
  Employing matching methods (point, line, arc, curve) to the data set the geometry can be reconstructed automatically for visualisation tasks. Anyway, for simulation purposes the segmentation of the facades is necessary. Thus, highly automated segmentation with minimal human interaction is needed in this subtask. The segmentation process is supported by precalculated features allowing to model the facades in a CSG-like manner by selecting and fitting primitives to the clouds of 3D points and features.

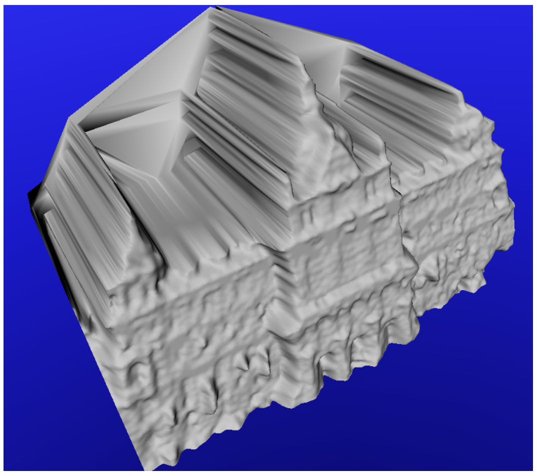
- Multiresolution texturing
  Preliminaries for the texture extraction and correction are the geometric model and the orientation of the corresponding images. Thus, for each 3D point on a facade the corresponding pixels in all images are known. Outlier detection methods help to detect and correct small occlusions (cable lines, pylons, or parking cars).

The above listed subtasks will deliver a photo-textured 3D model of facades which has to be combined with the aerial based data to build the *virtual habitat*. As mentioned in Section 2.1 the main focus in the *aerial data* research path will be on the conversion and integration of existing data combined with topological checks and the texture extraction from aerial images using their orientation parameters. The terrestrial image acquisition planning is based on footprints of buildings from the aerial images.

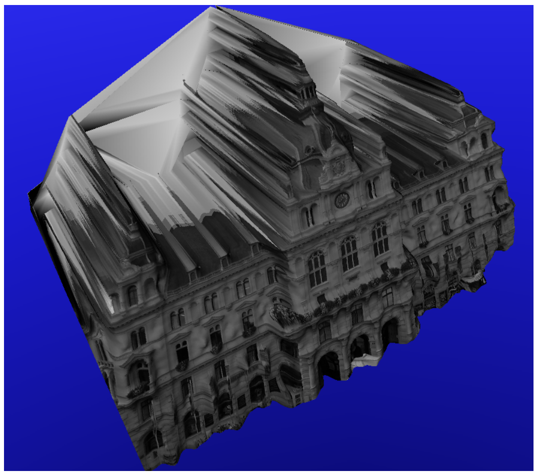
In the next two sections, we focus on subtasks of our software concept.

2.3 *Orientation of image sequences and reconstruction of facades*

The automatic reconstruction of 3D models from image sequences is an active field within the computer vision community (Pollefeys et al. 2000, Urban et al. 2000, Zisserman et al. 2000). The



(a) 3D model of a facade



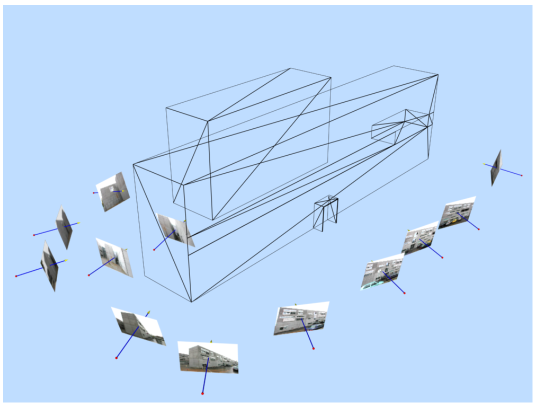
(b) Textured 3D model of the same facade

Figure 4. Automatic orientation of image sequences and reconstruction of facades.

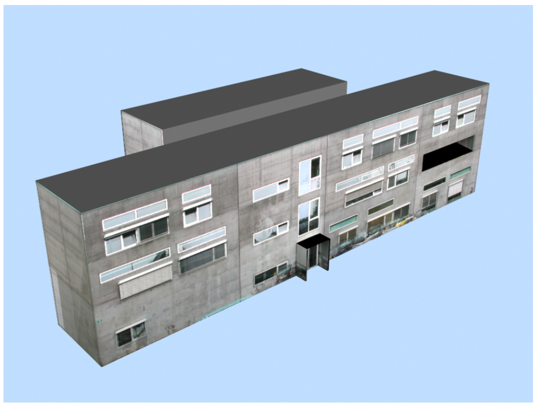
underlying geometric relationships between images are well known. The critical part is to robustly find corresponding points within the image sequences. In our approach we focus on an iterative and hierarchical method based on homographies to find this corresponding points inspired by work published by Redert et al. (1999). Furthermore, we are able to get an estimation for the quality of each triple of points. Figure 4 shows a 3D model obtained from a sequence of images which were oriented and modeled automatically.

2.4 *Multiresolution textures*

In this section, we present a method that automatically calculates texture maps for a given 3D object from a sequence of images. It is used in our image-based modeling approach after registration of the images and geometric modeling are completed. We show that the presented method uses the information from all images by implicitly applying a weighting function. Using this approach the consideration of modeled occlusions as well as the detection and removal of non-modeled occlusions is accomplished. Figure 5 states the problem.



(a) Input data: multiple views and geometric representation of the scene.



(b) Goal: reconstructed textured model of the building.

Figure 5. *Problem description:* input data for our approach versus output data.

The geometric relation between an image of an object, the 3D object itself, and the texture map is described by two transformations. Figure 6 shows these two transformations between texture, model and image space. Texture coordinates are transformed into model coordinates by the inverse texture mapping function M^{-1}.

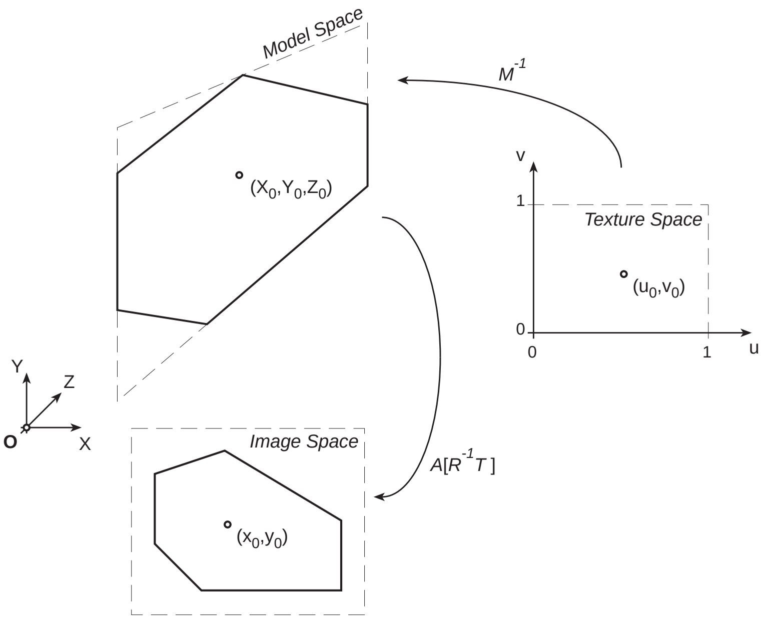


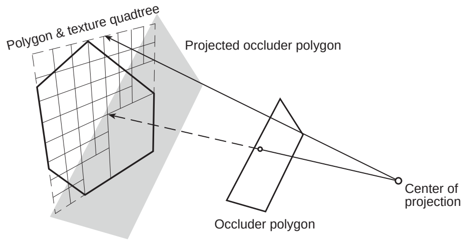
Figure 6. Geometric relations between texture, model, and image space.

The second transformation is the well known projective transformation. Using the computer vision notation, this transformation can be subdivided into an affine A, a perspective and an Euclidean $R^{-1}T$ transformation. The affine transformation is also known as the interior orientation of a camera and is calculated during the camera calibration. For each *image-polygon-texture* relation, a transformation t_{ij} between the texture T_i of a polygon and an image P_j can be found. A texture coordinate $\vec{\nu}$ corresponds then to the image coordinate $\vec{x}$ by the following relation (see Fig. 6):

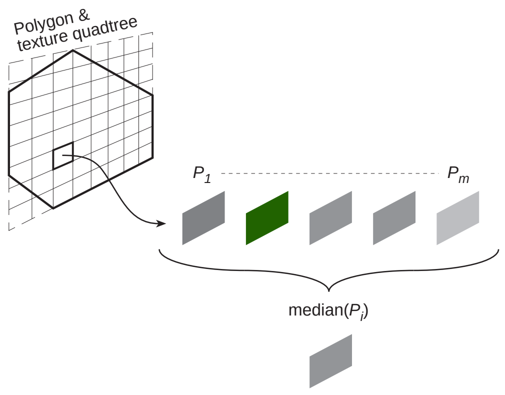
$$\vec{x} = A\,[\,R^{-1}\;T\,]\,M^{-1}\,\vec{\nu} \tag{1}$$

We decided to use a quadtree representation to store our multiresolution textures instead of image pyramids. Quadtrees offer the advantage of different depths for each subtree which is useful considering different resolutions in the texture space due to perspective distortion and occlusions. For every polygon in the scene a texture will be generated. Taking into account the density considerations mentioned above we start building up a quadtree Q for one of those polygons. The root node is the highest quadtree level and represents the whole texture space. A refinement of a quadtree node splits the covered area of this node into four quarters which are referred as children of the parent node. As long as there is information available in the images nodes of the quadtree will be refined. The leaves which are the lowest quadtree nodes represent then the highest resolution of the texture.

We start with a quadtree consisting of one node only. Successively each of the n images $P_1, \ldots, P_n$ are integrated into the quadtree using the transformation $t_i : T \rightarrow P_i$ which relates texture coordinates to image coordinates. The resolution for each texture point which is represented by a node of the quadtree Q is determined applying transformation t_i. The quadtree nodes are expanded according to the texture resolution observed in the image P_i. After gathering all the information from all images this information has to be distributed between the quadtree levels. This enables the extraction of an image texture with a fixed resolution and will suppress noise in



(a) The influence of occluding polygons to the quadtree structure.



(b) Pixel stack for one quadtree node and occlusion detection principle.

Figure 7. Two methods to handle modeled and non-modeled occlusions.

the high resolution portions of the texture. Node insertion is done by recursively subdividing all quadtree nodes with a lower texture resolution than corresponding image resolution.

Until now, scenarios are selected where neither occlusions from other objects nor self-occlusions occur. In other words the entire texture is seen from each viewpoint.

Typically views from real-world scenes expose occlusions which means that parts of or the whole textures under investigation are not visible from that viewpoint. Self-occlusion and occlusion from other modeled objects can be eliminated by integrating visibility considerations. The visibility of a quadtree element (i.e. a node) is calculated by tracing rays from every corner of that node $\vec{P}$ to the center of projection $\vec{C}$.

Our algorithm is therefore adapted to additionally perform a visibility test. The recursion in subdividing the quadtree is carried out unless one of four corner points of the node is visible. Starting with the root node of the quadtree this tree will be refined until the highest resolution within the input image is reached or the corresponding part of the quadtree element is not visible from that view.

Figure 7a shows the influence of an occluding polygon to the structure of the quadtree. Two rays from the corners to the center of projection are shown. One of them hits the occluding polygon and the other one goes through.

Up to now, we assume that every pixel which is collected in the quadtree represents the correct diffuse reflectance coefficient scaled by the intensity of the ambient illumination. Pixels that pass the visibility calculation step will contribute to the corresponding quadtree node according to their calculated size. Differences in pixel color for a particular quadtree node are assumed to relate to differences in resolution. Since quadtree entries are weighted according to the area in image space it is guaranteed that higher resolution information is preserved.

If artifacts from occlusion due to not modeled objects appear, we have to use outlier detection mechanisms. One commonly used method is the employment of a median filter. To be able to apply a median filter we have to store all entries of the quadtree coming from all input images separately in a list first (see Fig. 7b).

It is straightforward that an outlier can be eliminated having at least three entries. Since it is hard to predict the number of entries and the amount of outliers for all quadtree nodes during the image acquisition step, capturing more than three images for a part of a texture is a good idea. One of the possible extensions to our method is to use the number of entries per quadtree node to make a plan for the next view during the image acquisition step. This will require an on-the-fly reconstruction of the geometric model and registration of all views.

3 3D DATA MANAGEMENT AND VISUALIZATION

Due to the large amount of data, it is impracticable to replicate the whole data set on each workstation. Therefore, a dedicated database server is used to provide data to a number of clients. The data structure used for storage of 3D data is crucial for efficient access, transmission, and visualization.

3.1 *3D data management*

A main requirement is the ability to access (typically localized) subsets of the data. This is accomplished by using an hierarchical data structure (such as in Kofler (1998) for GIS applications and El-Sana & Chiang (2000) for arbitrary polygonal data).

We focus on the LOD-R-tree (Kofler 1998), since it is simple and efficient and is suitable for city models for the following reasons:

- Scalability: due to the straightforward mapping of the R-tree-structure onto object-oriented and even relational database designs an existing DBMS (database management system) can be used with little effort. Hence the limiting factor in terms of data size is the amount of available disk space.
- Multiresolution hierarchy: the LOD-R-tree hierarchy is designed to reflect a geometrical subdivision of the scene and can therefore easily be used for spatial queries. This both facilitates identifying a building given a particular location (in full detail) and retrieving all data within some region (at different levels of detail). The former corresponds to typical GIS-style applications, while the latter is mandatory for performance reasons in interactive visualizations of the whole scene.
- Progressive transmission: in case a larger subset is queried from the database over a slow network connection, transmission takes place in a way offering a rough “preview” of the detailed query result.

3.2 *Visualization*

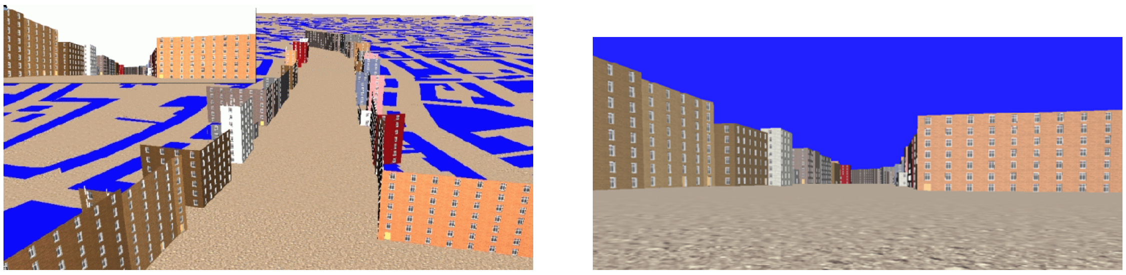
Real-time rendering applications such as the visualization of complete models of our urban outdoors usually exceed the capacity of even high-end computer graphics hardware. Thus, different approaches have to be taken to reduce the complexity of the 3D models. Typical methods are:

- Level of Details (LOD)
- Occlusion culling
- Impostors (image based rendering)

It has early been recognized that an hierarchical scene description of the same object at different geometric resolutions (Clark 1976) has lots of benefits. The LOD-R-tree (see Section 3.1) supports retrieval of different regions of the scene at different levels of detail. As the user navigates through the scene, different regions become visible at different distances. Portions of the scene outside the current view frustum can be omitted from rendering, and regions in the background can be rendered with less detail without visible loss of quality.

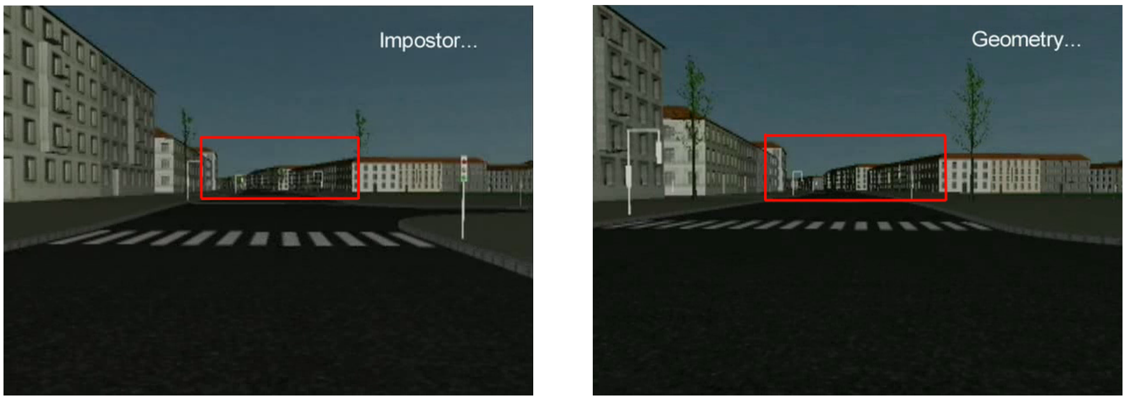
The drawback of LOD-R-trees is the limited number of representations of each object, making transitions between different levels of detail clearly visible. However, each single representation can be highly optimized (e.g., as an OpenGL display list (Woo et al. 1999)) to improve performance.

Occlusion culling is used in a preprocessing step to calculate the visibility for view cells. The advantage is the reduction of geometric complexity by a factor of 100 and more. Figure 8 illustrates the effect of occlusion culling. Figure 8a shows a part of a city model from a bird’s eye of view, with the potential visible polygons. Figure 8b shows the rendered image of the same scene from the viewer’s position. Obviously, this approach works only for restricted positions in height,



(a) Overview of used geometric representation (b) Rendered image from street level

Figure 8. Real-time rendering toolkit: Occlusion culling (contributed by M.Wimmer and P.Wonka, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria).



(a) Impostor (b) Geometry

Figure 9. Real-time rendering toolkit: Impostor (contributed by M.Wimmer and P.Wonka, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria).

useful for instance for car driving simulations. More detailed descriptions can be found in Wonka et al. (2000).
Another useful tool for real-time rendering is the usage of impostors. Parts of a scene which are far away from the viewer may be replaced by a textured polygon. Figure 9 shows two images from the same scene, where parts of the rendered image are represented by a textured polygon (a) respectively by the geometric representation (b). For more information see Wimmer et al. (2001).

4 FUTURE WORK

4.1 *City scanner*

The segmentation of facades into meaningful parts like doors or windows is essential for further applications. Thus, future work will include the implementation of highly automated support this kind of segmentation. Furthermore, the prototype will be designed to guide the user during the data acquisition to avoid missing data.

4.2 *3D data management and visualization*

While the LOD-R-tree concept is well suited for organizing large 3D scenes, it replicates geometry at each level of detail, therefore restricting the number of different levels. A more flexible database hierarchy that also includes CSG-operations only needs to store the differences between LODs, reducing the amount of data being stored and transmitted. To apply refinements to an object in the scene, the client therefore doesn't have to wait for a more detailed version of the whole object to be transmitted. Instead only the modifications are received and executed by the client (progressive LOD).

ACKNOWLEDGEMENT

This work has been done in the VRVis research center, Vienna and Graz, Austria (http://www.vrvis.at), which is partly funded by the Austrian government research program Kplus.

REFERENCES

Clark, J. H. 1976. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM* 19(10): 547–554.

Debevec, P. E. & Malik, J. 1997. Recovering high dynamic range radiance maps from photographs. In T. Whitted (ed.), *Proc. SIGGRAPH 97 Conference, Annual Conference Series, 369–378*. Reading, MA: Addison Wesley.

El-Sana, J. & Chiang, Y.-J. 2000. External memory view-dependent simplification. *Proc. Eurographics, Computer Graphics Forum, 19: 139–150*. Oxford: Blackwell Publishers.

Heikkilä, J. 2000. Geometric camera calibration using circular control points. *PAMI* 22(10): 1066–1077.

Kofler, M. 1998. *R-trees for visualizing and organizing large 3D GIS databases*. Ph.D. Thesis, Institute for Computer Graphics and Vision, Graz University of Technology.

Pollefeys, M., Koch, R., Vergauwen, M., Deknuydt, A. A. & Van Gool, L. 2000. Three-dimensional scene reconstruction from images. In J. H. Nurre & B. R. Corner (eds), *Proc. SPIE Conference on Three-Dimensional Image Capture and Applications II, SPIE Proc. vol. 3640, 215–226*. Bellingham, Washington: SPIE.

Redert, A., Hendriks, E. & Biemond, J. 1999. Correspondence estimation in image pairs. *IEEE Signal Processing Magazine* 16(3): 29–46.

Rothwell, C., Mundy, J., Hoffman, W. & Nguyen, V. 1995. Driving vision by topology. *Proc. IEEE Symposium on Computer Vision (SCV95), 395–400*.

Schmid, C. & Zisserman, A. 2000. The geometry and matching of lines and curves over multiple views. *IJCV* 40(3): 199–233.

Urban, M., Pajdla, T. & Hlavac, V. 2000. Consistent projective reconstruction from multiple views. In A. Leonardis, F. Solina & R. Bajcsy (eds), *Confluence of Computer Vision and Computer Graphics, NATO Science Series, 49–67*. Dordrecht: Kluwer Academic Publishers.

Wimmer, M., Wonka, P. & Sillion, F. 2001. Point-based impostors for real-time visualization. *Proc. Eurographics Workshop on Rendering*. Berlin: Springer (to be published). Available at http://www-imagis.imag.fr/Publications/2001/WWS01b/index.fr.html (accessed 1 June 2001).

Wonka, P., Wimmer, M. & Schmalstieg, D. 2000. Visibility preprocessing with occluder fusion for urban walkthroughs. *Proc. Eurographics Workshop on Rendering, 71–82*. Berlin: Springer.

Woo, M., Neider, J., Davis, T. & Shreiner, D. 1999. *OpenGL Programming Guide, 3rd edition*. Reading, MA: Addison Wesley.

Zhang, Z. 2000. A flexible new technique for camera calibration. *PAMI* 22(11): 1330–1334.

Zisserman, A., Fitzgibbon, A., Baillard, C. & Cross, G. 2000. From images to virtual and augmented reality. In A. Leonardis, F. Solina, F. & R. Bajcsy (eds), *Confluence of Computer Vision and Computer Graphics, NATO Science Series, 1–23*. Dordrecht: Kluwer Academic Publishers.