Building and Rendering 3D Navigable Digital Cities

Jose Luis PINA, Francisco SERON and Eva CEREZO

The GI_Forum Program Committee accepted this paper as reviewed full paper.

Abstract

This work focuses on automatically building 3D data sets from urban databases and visualizing them in real-time. The main contribution of the paper is to present a new data structure, associated with a corresponding construction method, specially suited to perform real-time digital city navigation. The structure enables an efficient application of the usual view-culling algorithms, making it possible to perform interactive urban walkthroughs and flights without the use of predefined paths. The basis of the structure is the block, which can be easily identified in any urban environment, regardless of the origins and structure of the entry data. In our case, a digital cadastral map is taken as the data source for the 3D city model, but the proposed data structure can be applied to data acquired from several sources: 2D GIS, terrain measurements, etc.

1 Introduction

Urban environments, which are among the largest and most complex 3D scenes, are particularly interesting for several simulation applications: crowds in motion, emergency evacuations, virtual tours, urban planning, military operations, traffic management and its noise impact in the surrounding buildings, etc. To take city data and convert them into a 3D city model in an automated way it is not an easy task; neither to visualize its geometric elements all together and in real time. In this article, it is discussed how to meet these challenges and produce visually navigable city models. The system presented allows the automatic generation of 3D models from urban databases and shows that if urban data are organized by using an appropriated data structure, real time visualization is possible.

All the elements of the city are grouped following a basic unit: the urban block. Setting out from typical quadtree decomposition and an R-tree, a new data structure that we named BqR-Tree or Block-quadtree R-Tree is built. The results obtained prove that it is possible to improve to a great extent the rendering speed using an adequate data structure such as ours, achieving significant improvements even when compared with the data structures which have yielded best results to date. A digital cadastral map is taken as the data source for the 3D city model, but the proposed data structure can be applied to data acquired from several sources: 2D GIS, terrain measurements, etc. Another advantage of our method is that identification of buildings is not required. The basic unit, the urban block, can be automatically identified under any circumstance. Therefore, the BqR-tree it is capable of being applied to urban data with low structure or no structure at all.

The organization of the rest of paper is the following: next section is devoted to discuss previous related work. Section 3 shows the method to build the 3D city model, section 4 presents the new data structure and section 5 shows visualization results and comparisons. In section 6 conclusions are summarized and future work is outlined.

2 Related Work

Most usual techniques to reconstruct 3D cities are classified into:

A. Photogrammetry techniques (ATKINSON 1996): small reflective targets points are placed and a set of images are obtained. It is a technique which delivers results of high accuracy but human intervention is usually required. It is a point-based technique, which does not consider the structure of buildings and thus it is not optimal.

B. Computer vision techniques (FAUGERAS et al. 1998): they are rather similar to photogrammetry but do not require targets placed, there are based on feature points. They establish a correspondence between the same features in different photographs of the same scene. Nevertheless, automatic detection of features and correspondence between them are difficult to obtain in an accurate way.

C. Aerial methods (SUVEG & VOSSELMAN 2000): they can provide locations and height information that ranges from good to highly accurate data. However, the detail of the sides of buildings, is very poor and it is difficult to perform the extraction of buildings in densely populated areas. Aerial images can be fused with 2D GIS data (PASKO & GRUBER 1996), which improves both methods, but really it is a reconstruction method from a 2D map refined by aerial location.

D. Ground-based (ZHAO & SHIBASAKI 2001) systems: they use cameras to collect highly accurate 3D details on building facades and ground-level, but the data provided by these technologies are typically incomplete due to the occlusion of trees or cars.

E. Ground plans (BRUNN & WEIDNER 1997): compared to other automatic procedures these ground plans are very reliable since they contain aggregated information which has been acquired and interpreted by a human being. The digital cadastral map is one of the most usual ground plans, and provides geometric data and aggregated information such as the names of streets. In fact, transformation from a 2D digital cadastral map to a 3D city model, as is our case, is a modern issue of urban data management.

Once the model is available, its interactive visualization requires the use of specific techniques especially devoted to render large scenes. These methods can be classified into four main groups of techniques: billboards, occlusion culling, level of detail, and view culling.

A. Billboards (SILLION & DRETTAKIS 1997) and impostors, are based on the use of images of objects instead of the 3D objects themselves. They are very effective in reducing rendering time. In fact, the task of visualizing models with many polygons is not really solved, but avoided. A minor variation of billboards is the image reuse technique (SHADE et al 1996): the images of the object are stored in real time for the duration of a frame, and then the images are reused providing changes remain below a certain threshold.

B. The occlusion culling technique resorting only to software techniques (COHEN-OR et al. 2002) or taking advantage of the GPU (BITTNER et al. 2004), involves culling away objects that are not visible due to their being hidden. The identification of hidden objects from the camera's point of view is the major challenge involved. A new identification is needed for each frame every time the camera or an object moves. This technique is widely used in urban environments, but it is not suited to urban flights in real time.

C. Level of detail (LOD) is one of the most frequently used techniques: continuous LODs have been tested (DOLLNER & BUCHHOLZ 2005), but they reduce rendering speed; so have hierarchical models (FUNKHOUSER & SÉQUIN 1993) that choose the proper resolution according to the node of the tree being displayed. However, it is very difficult to avoid bothersome drops between frames of different resolutions.

D. View frustum culling (NIRNIMESH & NARAYANAN 2006) is an acceleration technique that culls away all the scenegraph's nodes that lay outside the view frustum, i.e. those objects that are outside the observer's field of view, by using the nodes' bounding volumes. The division of complex geometry, consisting of multiple triangles, into an adequate data structure can greatly improve the ability to cull away triangles that lie outside the view frustum, resulting in less triangles to be sent to the rendering pipeline.

Regarding data structures (FONSECA & JORGE 2003), two main categories can be found among the most widely used indexing structures: those which belong to the K-D-Tree, and those which belong to the R-Tree category. Structures which belong to the first category use space partitioning methods that divide data space along predefined hyper-planes regardless of data distribution. The resulting regions are mutually disjointed and their union completes the entire space. Structures which belong to the second category use data-partitioning methods which divide data space in buckets of MBV (Minimum Bounding Volumes) according to its distribution. This can lead to possible overlapping regions. Quadtree, belongs to the first category.

From the comparative survey of data structures published by (GAEDE & GÜNTHER 1998), we may conclude that those data structures which belong to the K-D-Tree and R-Tree families yield the best performance. The best improvements of the R-Tree family have been analyzed by (KATAYAMA & SHINICHI 1997), and conclude that the VAMSplit R-Tree (WHITE & JAIN 1996) is the data structure with the greatest rendering speed up.

In this paper, is proposed the use of an indexed data structure to be built in preprocessing time for acceleration purposes. Regarding spatial decomposition, instead of using MBV as is usual in an R-Tree, and in most urban applications, it have been chosen to use a quadtree decomposition (AYALA et al. 1985) which is rare but not unique in urban environments (MANOCHA 2008). To test the structure performance, the VAMSplit R-Tree will be used as reference, as it is recognized as the one that yields better results. Details of the proposed structure are given in next sections.

3 Loading the City Data

The source of the city data used to test the proposed data structure is a digital cadastral map. The reason of choosing a cadastral map is its great number of advantages compared to other acquisition methods:

- Its not expensive, almost every town has got nowadays one, and city councils are responsible of its maintenance.
- Data have been collected and incorporated by persons and are, therefore, highly accurate, and usually, up to date. Being our cities living entities that grow up every day, there is not any other data source as updated as cadastral maps.
- Cadastral maps contain information (such as names, uses, ...) associated with the geometric data that can be incorporated to the 3D city model; some of this information is not accessible from any other acquisition source.

To automatically build the 3D model from the cadastral map, the next steps were followed:

- First of all, a selection of the geometry and the useful associated data was performed.
- Second, streets were located; streets are usually identified in cadastral maps, otherwise it is necessary to perform the identification.
- Third, blocks were identified from streets. For every street, all the streets ending or beginning at it were tested in a recursive manner. Beginning at the initial point of a street, the paths formed by streets connected together, which ended at the initial point, were searched. If the final point of a street coincided with the initial point of several streets (street intersection), the selection of the suitable street was performed in a counter-clockwise manner. Every block obtained was stored with its four extreme points to build its bounding-box.
- Fourth, with all the blocks located, the allocation process of the geometry into the blocks was performed. Bounding-boxes speed up this allocation process. Every polygon must belong to a block; one point of each polygon was tested against the bounding-box of the blocks. This is a very fast operation and it allows to remove almost all candidates. The final assignation was performed by means of the ray method. In this method a line is traced from the coordinates' origin to a point of the polygon; if the perimeter is cut an odd number of times the point is inside, otherwise it is outside. There is a little group of uncertainties, when the traced line coincides with a street of the perimeter, or when the cutting point is a vertex of the perimeter. However, the problem is solved by using another point of the polygon not included in the line. It is only necessary to test one point for each polygon, because each polygon must have all its points into one and only one block.

Finally, with all the blocks identified and all the geometry (and all the associated data) belonging to a block, the data structure is ready to be built following the method explained in the next section. As it will be see, the use of the data structure allows fast visualization and navigation across the 3D city model.

4 Defining the BqR-Tree

The proposed BqR-Tree is based on the decomposition of a city into blocks: the block is considered the minimum and indivisible unit of the city as well as the basis of the proposed structure. The term block is used to name the group of urban elements completely surrounded by streets, i. e., the usual meaning of urban block.

The BqR-Tree structure is an improved R-Tree whose method of space partitioning is a quadtree decomposition. Compared to the K-D-Tree decomposition used in the VAMSPlit R-tree, the quadtree decomposition is more adequate to the spatial distribution of the blocks and requires less culling time.

The structure also incorporates a second improvement. The bounding-boxes of the nodes have been replaced by bounding-spheres according to the results of (KATAYAMA & SHINICHI 1997). This produces improvements in node access speed and storage requirements. Nevertheless, the use of bounding-spheres produces too much overlapping, which is the reason for the use of the intersection of bounding-boxes and spheres for the leaves, as it will be explained later on.

In order to build the structure, certain tasks have to be carried out first. In particular, it is necessary:

- To identify all the blocks in the town and calculate its geometric centers and bounding spheres,
- To assign every graphic element of the town to an urban block.

In Figure 1 (left) a part of the city under consideration (Zaragoza, Spain) with the blocks already identified is displayed. Identification of blocks takes place, in this case, setting out from the streets, which are identified from the outset. Block status is assigned to any portion of terrain completely surrounded by streets.

To build the BqR-Tree, the first step is to take the minor rectangle bounding the city. This rectangle is divided into four equal rectangles called quadrants or buckets; all quadrants are recursively subdivided in this manner. The BqR-Tree tree is formed by recursive division of the city into quadrants. A quadrant is considered to be indivisible if it contains one block. No splitting of the urban blocks is implemented, since this would involve a scattering along the tree of the pieces of urban blocks, which would in turn lead to a poorer performance of the data structure. Should a division cross a block, the entire block is assigned to a single quadrant, the one that contains the geometric center of the block. At the end of the process, every final quadrant contains a single block or remains empty. Figure 1 (right) presents a quadtree decomposition performed on the example shown in Figure 1 (left).

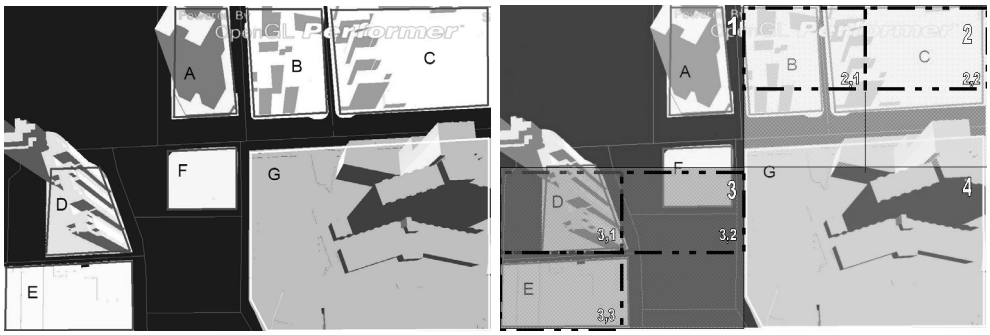

Fig. 1: Left: Identifying the blocks in a city, right: Quadtree decomposition of the Fig. 1 left

The second step is to build the tree. To do this, every quadrant is assigned to a node or sub-node and every block is assigned to a leaf. Figure 2 shows a graphic representation of the BqR-Tree structure corresponding to the example of Figure 1. Leaves are represented by squares and nodes by ellipses. Empty quadrants are not assigned to nodes, thus categorizing the quadtree as an adaptive quadtree. Each leaf of the tree stores the entire geometry of the block as well as the bounding volume for that block. Each node stores a pointer to its descendants and to the bounding-sphere of the node. In the case of the leaves, the bounding-box-sphere of each block (intersection of the sphere of the block and the quadrant box) is calculated and stored. For each node, the bounding sphere is composed of the union of bounding-spheres of its descendants, in a bottom-up manner.

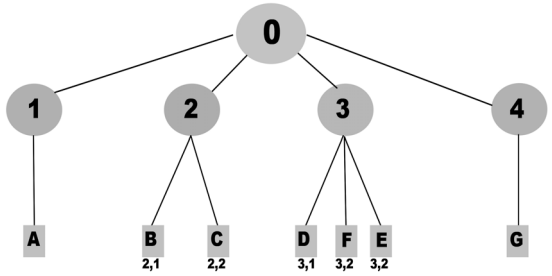

Fig. 2:
Tree representation of the Figure 1

Therefore, the resulting structure is a tree of the nodes' bounding-spheres which ends with the bounding-boxes-spheres and contains the geometry of the blocks as well. A linear codification, corresponding with its quadtree decomposition, based on a numeric key, is used to identify all the elements of the tree, nodes and leaves. Storage of the key of each element of the tree is required due to the non-representation of empty nodes. The quadrants' dimensions are not needed to be stored in the tree, there are only needed for the spatial distribution, also the bounding-volumes of the quadrants are not calculated.

Summarizing, the properties of the BqR-Tree data structure are the following:

- The structure is a tree of the nodes' bounding-spheres (quadrants are disjointed but not the bounding-spheres of the nodes associated with them) and ends with the bounding-boxes-spheres and the geometry of the blocks.
- Although very similar to an SR-Tree and VAMSplit R-Tree, the most important difference is the use of the quadtree decomposition with no splitting blocks, and the use of bounding-boxes-spheres for the leaves.

5 Results

The data used to test our model belongs to the city of Zaragoza (Spain). The original cadastral map was in GIS Microstation format. Data was exported to the FXCU1/FXCR (2004) exchange format, which is widely used in Spain to exchange the graphical information associated to cadastral maps. Several files were available, each one corresponding to a layer: constructions, roads and sidewalks. Every file contained graphical and associated data, for example, the name of the streets and the number of plants of a building.

The core of the graphical data were polygons closed and disordered. In the initial files neither buildings nor blocks were already identified as entities. Therefore, filtering, structuring and 3D elevation has been necessary.

The starting point for the tests was the generated 3D model of the city of Zaragoza, which is comprised of 1,688,023 triangles (see Fig. 3), 145 nodes and 96 leaves. All tests were performed with a computer provided with a DualP4 Xeon Pentium 4 processor at 2.8 GHz and a memory of 2 GB. The graphic card is a GeForce Fx 6800 Ultra with a memory of 256 Mb. The operating system is Windows XP.

OpenGL Performer, has been used to test the proposed BqR-Tree data structure and compare it with the VamSplit R-Tree data structure. Figure 3 (left) shows the BqR-Tree tree implemented by Performer while flying over the city. In order to load the structure, a file with all the geometry and the BqR-Tree structure elements arranged in nodes and leaves is supplied to Performer. Of course, any other scene graph may be used, as only a new dll, which can be generated for the new format, is required. Although Performer is equipped with tools for speeding up rendering, in order to show that the increase in rendering speed is caused only by the data structure, the only acceleration technique implemented is view culling, which is directly derived from the culling of the structures.

Performer, as the other scene graphs, implements its view culling in a top-down manner. The bounding-volume of a node, the root node at beginning, is tested against the view frustrum, if the bounding-volume of the node is partially into the view frustrum, the culling process continues with its descendent, if the node is completely into the view frustrum the node and its descendent are accepted without further search, otherwise the node and its descendent are culled away because they are not visible.

A program with the essential elements allowing free or guided camera movements was implemented. A route containing all the usual camera movement in flights and walkthroughs was designed; this route remains the same for all tested structures. The route (see Fig. 3 right) begins at the outskirts of the city at ground level, and undertakes a walkthrough to the heart of the city. Once there, the camera rises vertically until it is above the rooftops facing the ground. Finally, a diagonal flight is performed from this position to the ground.

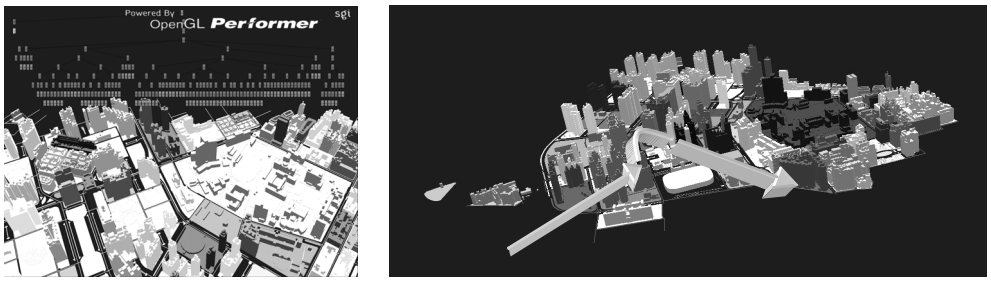

Fig 3: Left: Tree implementation in Performer, right: 3D city model and camera route

5.1 Performance Tests

To determine the real improvement resulting from the use of the BqR-Tree, it was decided to compare it with the data structure which yielded the best results in the tests performed by other authors, the VAMSplit R-Tree structure. An unstructured model was also used as a reference.

The evolution of the time required (in seconds) to render each frame for each of the three previously mentioned data structures can be seen in Figure 4:

- The white line (top line) corresponds the unstructured route.
- The grey line (middle line) corresponds the VamSplit R-Tree structure, and the black line (bottom line) corresponds the BqR-Tree line.

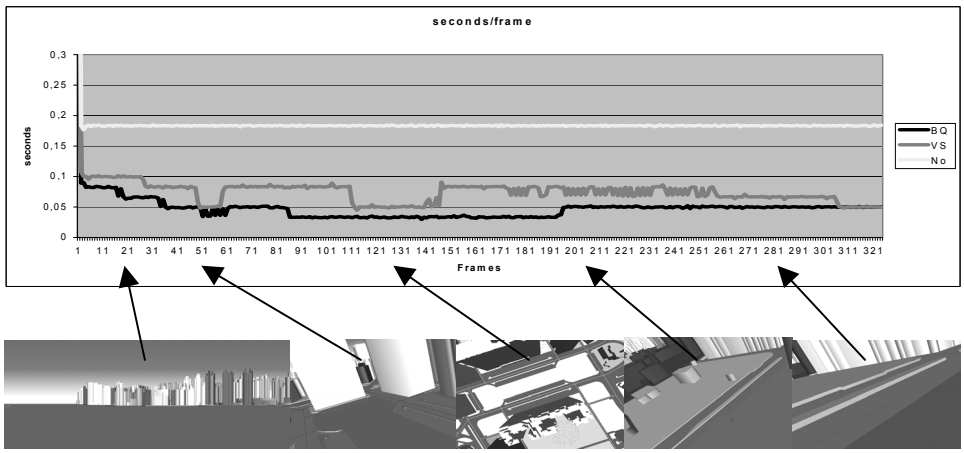

Fig. 4: Evolution of rendering time for each structure. A few images of what the camera sees at key moments have been added.

As explained, the only acceleration technique implemented is view culling, which is directly derived from the culling of the structures. The white line is constant, and therefore the time required for rendering is always the same for each frame. This was predictable, since there can be no view culling if there is no data structure. Regarding the other two lines, the first consideration is that the black line (which represents the BqR-Tree) is the line with the quickest rendering time for each frame, and is markedly inferior than the grey line (which represents the VamSplit R-Tree). In fact, the grey line is not better at any point of the route; at the very most, it is equal to the black line in a small number of frames. The BqR-Tree line is also noticeably more regular than the line which represents the VamSplit R-Tree. The latter shows two great depressions and multiple oscillations. If the results are analyzed the reason for this becomes apparent. At the beginning of the route, the camera sees the city from far away, and therefore with no opportunity for view culling, thus the initial equivalence of the red and blue lines. Nevertheless, the more the camera advances into the city, the more important view culling becomes and the more both lines deviate. The ascent of the camera takes place between frames 48 and 140; this is the interval where both

structures approximate the most. Later on, a diagonal flight towards the ground separates them once again. In the last 20 frames both lines come together again when the camera is facing the floor and is about to land and view culling possibilities are similar for both structures.

The average performance of each structure is shown in Table 1. An improvement of almost 40 % can be observed in relation to the VamSplit R-Tree structure and the results are 75 % better than when no structure is used.

Table 1: Average values for each structure

	Seconds/frame	**Ratio BqR-tree/Structure**
No tree	0.185	0.261
VamSplit R-tree	0.0758	0.638
BqR-Tree	0.0483	1

6 Conclusions and Future Work

We have developed a straightforward method to build a 3D city model from a digital cadastral map. The use of special structures especially suited to urban environments, allows to perform flights and walkthroughs in real time. The main features of the structure, the BqR-Tree are:

- It is defined by considering the city block as the basic and logical unit. The advantage of the block as opposed to the traditional unit, the building, is that it is easily identified regardless of the data source format.
- The usefulness of the structure has been tested with low structured city data, which makes its application appropriate to almost all city data.
- The application of view-culling to the BqR-Tree data structure leads to an important improvement compared to existing data structures when performing city visualizations, rendering times improve by an average of 30 % in comparison to the data structures which have yielded best results to date.

The next step would be to add mobile elements to the city, such as pedestrians or cars, and to handle them appropriately by using the proposed data structure.

Acknowledgements: This work has been partly financed by the Spanish Dirección General de Investigación, contract number Nº TIN 2008-0574 and by the Government of Aragón by way of the WALQA agreement.

References

ATKINSON, K .B. (1996), Close range photogrammetry and machine vision. Whittles.

AYALA, D., BRUNET, P., JUAN, R. & NAVAZO, I. (1985), Object representation by means of no minimal division quadtrees and octrees. ACM Trans. Graph., 4 (1), pp. 41-59.

BITTNER, J., WIMMER, M., PIRINGER, H. & PURGATHOFER, W. (2004), Coherent Hierarchical Culling: Hardware Occlusion Queries Made Useful. Computer Graphics Forum Proceedings of Eurographics 2004, 23 (3), pp. 615-624.

BRUNN, A. & WEIDNER, U. (1997), Extracting buildings from digital surface models. IAPRS.

COHEN-OR, D., CHRYSANTHOU, Y., SILVA, C. & DURAND, F. (2002), A Survey of Visibility for Walkthrough Applications. IEEE Transaction on Visualization and Computer Graphics.

DOLLNER, J. & BUCHHOLZ, H. (2005), Continuous level-of-detail modeling of buildings in 3D city models. Proc. of the 13th annual ACM international workshop on Geographic information systems, Bremen, Germany, pp. 173-181.

FAUGERAS, O. D., ROBERT, L., LAVEAU, S., CSURKA, G., ZELLER, C., GAUCLIN, C. & ZOGHLAMI, I. (1998), 3d reconstruction of urban scenes from image sequences. Computer vision and image understanding, 69 (3), pp. 292-309.

FONSECA, M. & JORGE, J. (2003), Indexing high-dimensional data for content-based retrieval in large databases. Technical report, INESC-ID.

FUNKHOUSER, T. A. & SÉQUIN, C. H. (1993), Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments. Computer Graphics Annual Conference Series, 27, pp. 247-254.

GAEDE, V. & GÜNTHER, O. (1998), Multidimensional access methods. ACM Comput. Surv. 30, 2, pp. 170-231.

KATAYAMA, N. & SHINICHI, S. (1997), The SR-Tree: an index structure for high-dimensional nearest neighbor queries. Proc. of ACM SIGMOD, 1997, pp. 369-380.

MANOCHA, D. (2008), Real-Time Motion Planning For Agent-Based Crowd Simulation. Proc. of 2008 IEEE Virtual Reality Workshop on Virtual Cityscapes.

NIRNIMESH, P. H. & NARAYANAN, P. J. (2006), Culling an Object Hierarchy to a Frustum Hierarchy. Berlin, Springer, 4338/2006, Computer Vision, Graphics and Image Processing, pp. 252-263.

PASKO, M. & GRUBER, M. (1996). Fusion of 2D GIS Data and Aerial Images for 3D Building Reconstruction. International Archives of Photogrammetry and Remote Sensing, XXXI, B3, pp. 257-260.

SHADE, J., LISCHINSKI, D., SALESIN, D. H., DEROSE, T. & ZINDER, J. (1996), Hierarchical image caching for accelerated walkthroughs of complex environments. SIGGRAPH 96: 23rd annual conference on Computer graphics and interactive techniques, pp. 75-82.

SILLION, F. G. & DRETTAKIS, B. B. (1997), Efficient Impostor Manipulation for Real-Time, Visualization of Urban Scenery. Computer Graphics Forum, Proc of EUROGRAPHICS, 16, 3, pp. 207-218.

SUVEG, I. & VOSSELMAN, G. (2000), 3D Reconstruction of Building Models, International Archives of Photogrammetry. Remote Sensing and Spatial Information Sciences, 33, B2, pp. 538-545.

WHITE, D. A. & JAIN, R. (1996), Similarity indexing: Algorithms and Performance. Proc SPIE, 2670. San Diego, USA, pp. 62-73.

ZHAO, H. & SHIBASAKI, R. (2001), Reconstructing Urban 3D Model using Vehicle-borne Laser Range Scanners. 3 Int. Conf. 3-D Digital Imag. and Modeling, pp. 349-356.