RELEVANCE-DRIVEN ACQUISITION AND
RAPID ON-SITE ANALYSIS OF 3D GEOSPATIAL DATA

D. Eggert, V. Paelke

IKG, Institute for Cartography and Geoinformatics, Leibniz Universität Hannover, Appelstr. 9a, 30167 Hannover, Germany, {eggert, paelke}@ikg.uni-hannover.de

ABSTRACT:

One central problem in geospatial applications using 3D models is the tradeoff between detail and acquisition cost during acquisition, as well as processing speed during use. Commonly used laser-scanning technology can be used to record spatial data in various levels of detail. Much detail, even on a small scale, requires the complete scan to be conducted at high resolution and leads to long acquisition time, as well as a great amount of data and complex processing.

Therefore, we propose a new scheme for the generation of geospatial 3D models that is driven by relevance rather than data. As part of that scheme we present a novel acquisition and analysis workflow, as well as supporting data-models. The workflow includes on-site data evaluation (e.g. quality of the scan) and presentation (e.g. visualization of the quality), which demands fast data processing. Thus, we employ high performance graphics cards (GPGPU) to effectively process and analyze large volumes of LIDAR data. In particular we present a density calculation based on k-nearest-neighbor determination using OpenCL.

The presented GPGPU-accelerated workflow enables a fast data acquisition with highly detailed relevant objects and minimal storage requirements.

1. MOTIVATION

A wide variety of 3D geospatial based applications have been proposed in recent years, mostly in relation to city modeling but also for other domains. Application paradigms like 3D location based services and augmented reality rely on appropriate 3D models of the environment as basic constituents. Despite technical advances, the cost effectiveness of creating and maintaining the required 3D models, as well as their appropriate presentation to users, remain a key issue. This situation is further complicated by the fact that 3D geospatial information is subject to frequent changes and that the current developments in 3D computer games and film animation lead users to expect a high fidelity of the models used in an application.

One central problem in applications using 3D model is the tradeoff between detail and acquisition cost, during conception, as well as processing speed, during use. Much detail, even on a small scale, requires the complete scan to be conducted at high resolution and leads to long acquisition time, great amount of data, and complex processing. Fast scanning in contrast will be shorter in duration but will provide lower resolution and an overall coarse model. Adding more detail and using more realistic graphics may seem the obvious solutions. However, often they are neither cost effective nor viable using existing techniques.

We suggest looking for alternative ways to provide 3D information on a large scale. Recent research has found that in a variety of visual applications that use 3D city models, a high amount of detail is only required for objects that are of high relevance to the user (Cartwright 2005)(Elias, Paelke and Kuhnt 2005).

We propose the generation of large-scale geospatial models that are driven by relevance rather than data. Particularly, developing new progressive acquisition and modeling techniques that provide a more coherent view into the available sources of information. To achieve this, our plan is to use laser-scanning technology and novel user interface techniques, providing instant visual feedback which demands fast data analysis.

The established workflow consists of a data acquisition stage in the field and a following processing and analysis stage in a standard in-door office environment. This makes fast multi-core computers or even entire computer clusters available for the analysis. An on-site environment lacks this computation power; therefore we employ high performance graphic cards to process and analyze the LIDAR data at the recording site. Compared to common CPUs, even mobile GPUs (build into Laptops) have a significant increased computation power. Since these computation capabilities entail higher power consumption, most high performance mobile GPUs come along with an integrated low power GPU. In case the high performance GPU is not

needed, it will be deactivated and the separate low power GPU takes over for longer battery life. Since the high performance GPU is only needed during the analysis, the concept of having a separate low power GPU fits the demands of our on-site analysis.

2. RELATED WORK

Almost all approaches to recognize salient objects and reconstruct their shape are data driven, targeting the extraction of every detail from the data, needed or not, see e.g., (Volsseman and Dijkman 2001), (Rottensteiner and Briese 2002), (Filin 2004), (Filin, Abo-Akel and Doytsher 2007), (Becker and Haala 2007). Recent advances in terrestrial laser scanning has shown that processing the point-clouds can be performed, under adequate representation, both efficiently in terms of processing time and with relatively limited computational resources (Zeibak and Filin 2007), (Gorte 2007), (Barnea and Filin 2007), (Barnea and Filin 2008). These results refer to the registration of the point clouds (Barnea and Filin 2007), (Barnea and Filin 2008), the extraction of primitives and objects (Gorte 2007), (Barnea, Filin and Alchanaties 2007), and to the association of scan pairs (Zeibak and Filin 2007).

The software currently used in the 3D reconstruction process and for data acquisition is designed for operation in standard in-door office environments, e.g., (InnovMetric 2008), (Cyclone 2008). Regarding on-site interaction, the user interface concepts of mixed and augmented reality (Milgram, et al. 1994), (Azuma 1997), (Azuma, Baillot, et al. 2001) that integrate the real environment into the user interface have shown high potential to support complex spatial interaction tasks. As an example, the Studierstube system (Schmalstieg, et al. 2002) demonstrates a number of promising spatial interaction concepts. Hedley (Hedley, et al. 2002) and others have demonstrated collaborative 3D geovisualization applications based on augmented reality techniques. While the technical challenges of mobile outdoor are great, there have been a number of demonstrators, e.g., the outdoor modeling application by (Piekarski and Thomas 2001). Another AR input/output device is the GeoScope (Paelke and Brenner 2007) that aims to avoid some of the central problems by providing high-precision video overlay in outdoor use-cases where high mobility is not required and seems well suited for acquisition applications.

A point cloud's density is an important indicator of its quality. In order to determine this density the k-nearest-neighbors (kNN) can be used. Most approaches for determining the kNN of a point in a point set rely on reducing the complexity of the required neighbor searches. They generally try to reduce the number of distances to calculate by arranging the data in spatial data structures, e.g. a kd-tree structure (Arya, et al. 1998) or by using Morton order or Z-order of points as in (Connor and Kumar 2008). Another recent proposal with promising results uses a brute-force search implemented using the C for CUDA API (Garcia, Debreuve and Barlaud 2008).

3. CONCEPT

The objective of our work is the effective creation of 3D geospatial models based on integrating global data (airborne laser or alternative sources) if available with local detail from terrestrial laser scans through progressive acquisition and modeling. Such modeling will be according to need, relevance, and controlled on-site with an augmented reality user interface.

The central idea is to control and limit the amount of detail in all processing stages to that actually required while providing feedback and on-site interaction capabilities. This will allow reducing acquisition time, modeling time, as well as storage and computation requirements in the actual use of the resulting models. Additionally, it will allow to focus on the relevant features needing further detailing. Such focus is almost impossible to achieve using uniform scans. For this we propose a demand-driven workflow, as shown in Figure 1, into which the acquisition, analysis, integration and presentation activities are embedded.
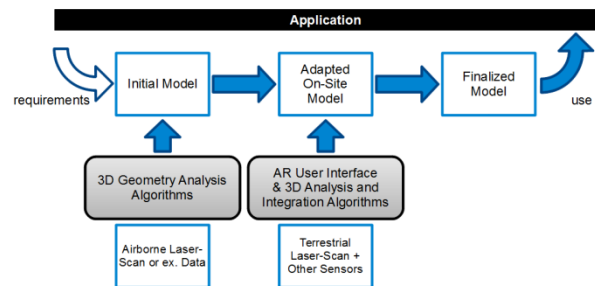

Figure 1: acquisition and analysis workflow

Based on the application and requirements, an initial model is acquired via airborne laser scanning where and if possible to provide a cost effective base model. Alternatively, existing 2D or 3D models can be used if available.

The central activity is on-site modeling, in which terrestrial laser scanning is employed. A user interface based on the paradigm of augmented reality (AR) in which the view of the real environment is augmented with information on the current model, its resolution and quality allows to control the acquisition and modeling process through intuitive decisions and selection of relevant features worth or need detailing. A mobile version of the previous mentioned GeoScope constitutes a suitable AR setup.

The AR user interface is closely coupled to 3D geometry analysis and integration algorithms that match and integrate data from different scans and data sources and provide measures of object distinctiveness, complexity and scan quality. In contrast to common off-line processing techniques, this approach requires 3D geometry analysis and integration schemes that operate at interactive speeds. Thus, we employ high performance graphics cards to speed up the analysis algorithms.

The resulting model can be further extended and refined either on-site or back in the office using established modeling tools or dynamically generated structure elements (e.g. pre-packed facade features from a library) to match the application requirements. It can then be applied in the intended application (e.g. visualization or precise positioning).

4. GEOMETRIC ANALYSIS ALGORITHMS

In order to create a 3D model from the LIDAR data, various geometry analysis algorithms must be applied. Beside several other algorithms the determination of the k nearest neighbors (kNN) of each point in the recorded point cloud is commonly needed. While algorithms like triangle-mesh reconstruction use kNN for surface reconstruction, our approach employs kNN to determine a density value for each point. The density in each point gives information about the quality of the recorded point

set. The density d_P of point P is the inverted sum of the distances between P and the k nearest neighbors P_i of P.

$$d_P = \frac{k}{\sum_{i=0}^{k}|PP_i|}$$

The association of colors to the minimum and maximum density value (e.g. max density = green, min density = red) allows to visualize the calculated density values, as shown in Figure 2.
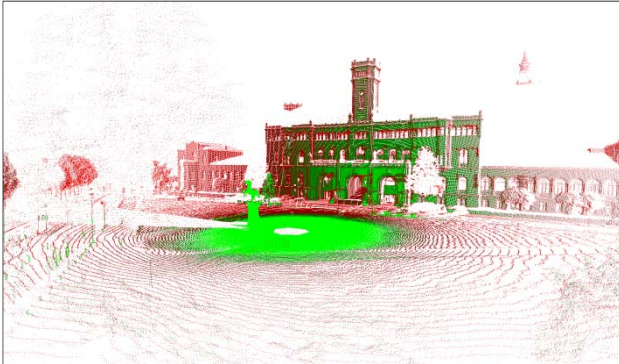

Figure 2: density visualization

Doing the kNN calculations on graphics processing units (GPU) raises two questions:

- Which kNN algorithm to implement
- Which programming language to use

The technique of using a GPU to perform calculations is often referred to as general purpose computation on GPUs (GPGPU). GPGPU is basically a kind of stream processing that exploits the parallel data processing capabilities of modern GPUs. Therefore the used algorithm needs to be highly parallelizable. Furthermore the processed data, or more specific the corresponding results, must be independent. Regarding the programming language to use leaves basically two options. Manufactures of graphics hardware like Nvidia and ATI provide their own proprietary GPU computing languages, like Nvidias "C for CUDA" or ATIs "Stream". Their major drawback lies in the particular hardware support, since only the manufacturers own platforms are supported. On the other hand there are hardware independent languages like OpenCL and Microsoft's DirectCompute. While both work with most graphics hardware, DirectCompute is still limited to operating systems supporting DirectX 11. In contrast the OpenCL language framework, which is managed by the Khronos Group, can be used in a cross-platform manner, regarding the used hardware (not only different GPUs, but even on CPUs and other processing units) as well as the used operating system, analogue to OpenGL. Therefore, we decided to use OpenCL to implement our algorithms in order to keep a maximum of flexibility.

4.1 Methods

Since all programming languages utilizing GPUs are lacking of an object-oriented modeling paradigm, the implemented kNN algorithms needed to be simple to keep the realization costs low. Considering the given GPGPU constraints, parallelizable processing and independent data respectively results, we evaluated two kNN algorithms.

First, we implemented the most simple brute-force kNN algorithm (BF kNN) following (Garcia, Debreuve and Barlaud

2008). BF kNN is by nature highly-parallelizable which makes it suitable for GPU computations. The second evaluated kNN algorithm is based on partitioning the point cloud in a pre-processing step. During the actual search only points of neighbored partitions are considered. This algorithm will be referred to as partitioned kNN (P kNN) search.

4.1.1 Brute-Force kNN Search

The brute-force kNN search calculates the distances between all points to determine the k nearest neighbors. This results in a quadratic runtime $O(n^2)$ with n being the number of points in the point cloud.

Since every distance between arbitrary points can be calculated independently, all distances could be calculated within a single step in parallel, assuming the corresponding number of computation units is present. This characteristic makes the BF kNN search well suitable for GPU computations.

4.1.2 Partitioned kNN Search

In contrast to BF kNN the partitioned kNN search needs a pre-processing step, which divides the space into partitions of equal space (and/or other constrains like equal number of contained points). In our case we just divided the space into equal sized partitions. This enables a linear runtime complexity $O(n)$ for the pre-processing step. The partition indices i_x for each point are calculated as follows:

$$i_x = \lfloor p_x * s \rfloor$$

where x is the corresponding dimension (in case of 3D: $x \in \{1, 2, 3\}$) and s is the number of partitions the particular dimension is divided in.

After creating the partitions the k nearest neighbors are determined by calculating the distances between all points in the same, as well as in the 26 neighbored partitions (partitions at the border of course have less). Assuming each of the three dimensions is divided into eight partitions ($s = 8$) the whole space is divided into $S = 8^3 = 512$ partitions. The kNN search situation for a single partition (solid red cube) for the described case is shown in Figure 3.
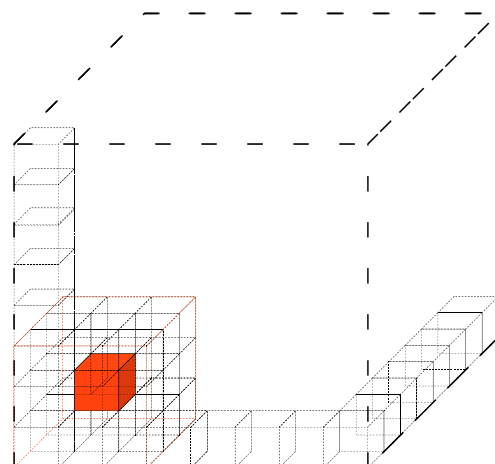

Figure 3: partitioned kNN search space

This also results in a quadratic runtime, however, with a considerably reduced number of distance calculations. Assuming the points are uniformly distributed and the space is divided into S partitions, there are $\frac{27}{S}n$ distances to calculate for each point of a partition, rather than n in case of the brute-force method.

However, the points in point clouds acquired using laser scanning technology are not uniformly distributed. The points accumulate at walls and nearby the scanning device, as seen in Figure 2. Since all partitions are processed in parallel, the resulting duration is the processing duration of the densest partition. Given that the number of points within partitions varies significantly, a more sophisticated processing scheme has high potential to improve the performance of the P kNN for such data sets.

5. TEST CASES AND ENVIRONMENT

We implemented the two kNN methods, BF kNN and P kNN, in OpenCL to run them on a GPU. In order to get comparable CPU-based results we implemented them in C++ as well.

The used test environment consists of an Intel Core 2 Duo E8400 with 3.0 GHz, 8 GB of dual channel DDR2 RAM, as well as an Nvidia Geforce GTX 285 with 240 stream processors.

While the C++ implementation is single-threaded the OpenCL implementation creates multiple calculation threads. In case of the BF kNN there are as much threads started as number of points in the point cloud. In case of the P kNN algorithm the thread count corresponds to the number of partitions.

The test scenarios included the calculation of $k = 10$ nearest neighbors for point clouds consisting of up to 60'000 points. Point clouds containing more points caused the OpenCL implementation to crash, so we assume the OpenCL/CUDA scheduler is unable to handle more threads. In case of P kNN the space was divided into 512 equal sized partitions. Overall, this test case was processed using BF kNN and P kNN running on the mentioned CPU, as well as GPU. In each case we ran three repetitions with different point sets.

6. RESULTS

As stated before we tested both algorithms with GPU-based, as well as with CPU-based implementations. While the x-axis shows the number of points in the point cloud, the y-axis shows the needed calculation time in milliseconds. The BF kNN method results are indicated with a red plus (+), in contrast the P kNN method results are indicated using a green x (x).

Figure 4 shows the CPU-based results. The computation time needed by the BF kNN method shows the expected quadratic complexity (i.e. processing 20'000 points needed approx. 10 seconds, while processing 40'000 points needed approx. 40 seconds). By contrast the P kNN is considerable faster, needing less than five seconds calculating the 10 nearest neighbors of 60'000 points, while the brute-force search needed about 90 seconds.
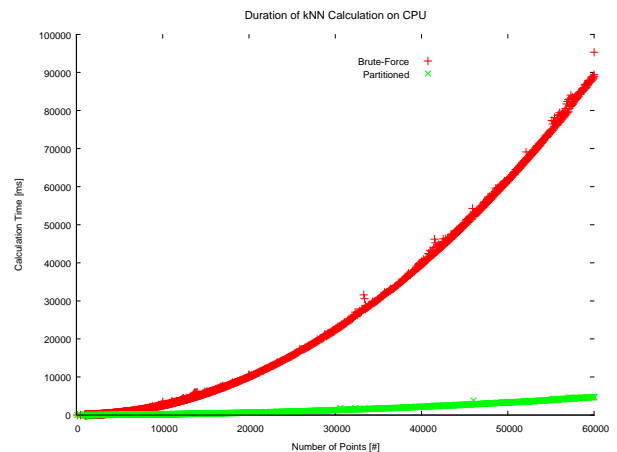

Figure 4: duration of kNN calculation using CPU-based implementation

The results of the GPU-based implementation are shown in Figure 5. Just as the CPU-based methods GPU-based alternatives have a quadric time complexity as well. Nonetheless both are significantly faster due to parallel execution (factor 45 in case of BF kNN and factor 3 in case of P kNN). In contrast with the CPU versions the different methods doesn't show a real difference in time consumption, while the P kNN method is slightly faster.
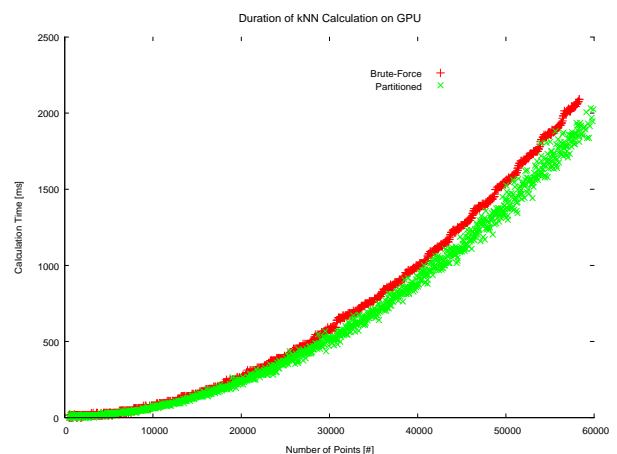

Figure 5: duration of kNN calculation using GPU-based implementation

The minimal difference is somehow unexpected, because the method using partitions calculates just a fraction of the distances the brute-force one does. The reason for this, might caused by the way the points, respectively the memory is accessed. While each GPU thread running the brute-force method always fetches the same point, memory address respectively, the partitioned search doesn't. The former is called coalesced memory access, which appears to be more efficient than the latter un-coalesced one. This will be topic of further research. Furthermore a growing variance of the P kNN method result is evident. This might be caused by the different distribution of the points in the point cloud.

Once the kNN for each point has been calculated, the density in the corresponding point can be determined. After applying adequate colors to the resulting density space, as mentioned in section 4, a live visualization of the density quality of the recorded point set is possible as shown in Figure 2. This enables

an assessment by the operator. Based on the results the operator can take appropriate subsequent actions.

7. CONCLUSION AND OUTLOOK

In this paper, we proposed a new scheme for generation of geospatial 3D models that is driven by relevance. The presented workflow, which includes on-site data evaluation and presentation, demands fast data processing. We are facing these demands by employing GPGPU to effectively process and analyze large volumes of LIDAR data.

In particular we presented a density calculation based on k-nearest-neighbor determination using OpenCL. The evaluated implementations using OpenCL accelerated the kNN search by up to a factor of 45 compared to the brute-force algorithm CPU-implementation. The P kNN algorithm acceleration reached a factor of up to 3. In summary the GPGPU analysis suites well the demands of the on-site environment and enables a much faster data analysis. Furthermore the GPGPU analysis concept isn't just limited to field environment as it can be used on standard PC hardware as well.

As discussed the P kNN OpenCL-based implementation leaves room for improvements, for instance in accessing the same points/memory in each GPU computation thread (coalesced memory access). Furthermore the algorithm needs to be adapted to efficiently process non-uniformly distributed point sets produced by using laser scanner. Both goals could be reached by changing the way the partitions are processed, e.g. processing all partitions in a serial manner and processing the points within a single partition in parallel. As mentioned this will be discussed in further research.

In addition to the improvement of the current implementation our further research focuses on an appropriate AR setup, as well as on suitable integration algorithms.

8. ACKNOWLEDGEMENT

9. REFERENCES

Arya, S., D. M. Mount, N. S. Netanyahu, R. Silverman, und A. Y. Wu. „An Optimal Algorithm for Approximate Nearest Neighbor Searching." *Journal of the ACM, 45*, 1998: 891-923.

Azuma, R. "A Survey of Augmented Reality." *Teleoperators and Virtual Environments, Vol. 6, No. 4.* 1997.

Azuma, R., Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. "Recent Advances in Augmented Reality." *IEEE Computer Graphics and Applications, Vol. 21, No. 6.* 2001.

Barnea, S., and S. Filin. "Keypoint Based Autonomous Registration of Terrestrial Laser Point Clouds." *ISPRS journal of Photogrammetry and Remote Sensing*, 2008.

—. "Registration of terrestrial laser scans via image based features." *International Archives of Photogrammetry and Remote Sensing. 36(3/W52).* 2007. 26-31.

Barnea, S., S. Filin, and V. Alchanaties. "A supervised approach for object extraction from terrestrial laser point clouds demonstrated on trees." *International Archives of Photogrammetry and Remote Sensing. 36(3/W49A).* 2007. 135-140.

Becker, S., and N. Haala. "Combined Feature Extraction for Façade Reconstruction." *International Archives of Photogrammetry and Remote Sensing. 36(3/W52).* 2007. 44-49.

Cartwright, W. "Towards an understanding of the importance of landmarks to assist perceptions of a space in web-delivered 3D worlds." *3rd Symposium on LBS & TeleCartography.* Vienna, Austria, 2005.

Connor, M., and P. Kumar. "Parallel Construction of k-Nearest Neighbor Graphs for Point Clouds." *Eurographics Symposium on Point-Based Graphics.* Los Angeles, CA, USA, 2008.

Cyclone. *Leica Cyclone.* 2008. http://www.leica-geosystems.com/corporate/de/ndef/lgs_6515.htm (accessed 8 1, 2008).

Elias, B., V. Paelke, and S. Kuhnt. "Concepts for the Cartographic Visualization of Landmarks." *Proc. 3rd Symposium on LBS & TeleCartography.* Vienna, Austria, 2005.

Filin, S. "Surface classification from airborne laser scanning data." *Computers & Geoscience 30(9-10)*, 2004: 1033-1041.

Filin, S., Avni, and A. Y. Baruch. "Quantification of Environmental Change in Receding Lake Environments." *Proceedings of FIG working week 2007 and GSDI-8.* Hong-Kong, 2007. 1-6.

Filin, S., N. Abo-Akel, and Y. Doytsher. "Detection and reconstruction of free form surfaces from airborne laser scanning data." *International Archives of Photogrammetry and Remote Sensing. 36(3/W52).* 2007. 119-124.

Garcia, V., E. Debreuve, and M. Barlaud. "Fast k nearest neighbor search using GPU." *CVPR Workshop on Computer Vision on GPU.* Anchorage, Alaska, USA, 2008.

Gorte, B. "Planar Feature Extraction in Terrestrial Laser Scans Using Gradient Based Range Image Segmentation." *International Archives of Photogrammetry and Remote Sensing. 36(3/W52).* 2007. 173-182.

Hedley, N., M. Billinghurst, L. Postner, R. May, and H. Kato. "Explorations in the use of Augmented Reality for Geographic Visualization." *Teleoperators and Virtual Environments, Vol. 11, No. 2.* 2002. 119-133.

InnovMetric. *InnovMetric Polyworks.* 2008. http://www.innovmetric.com/Manufacturing/home.aspx (accessed 1 8, 2008).

Milgram, P., H. Takemura, Utsumi A., and F. Kishino. "Augmented Reality: A class of displays on the reality-virtuality continuum." *SPIE Vol. 2351-34, Telemanipulator and Telepresence Technologies.* 1994.

Paelke, V., and C. Brenner. "Development of a Mixed Reality Device for Interactive On-Site Geo-visualization." *Proc. Simulation und Visualisierung.* Magdeburg, 2007.

Piekarski, W., and B. H. Thomas. "Tinmith-Metro: New Outdoor Techniques for Creating City Models with an Augmented Reality Wearable Computer." *5th Int'l Symposium on Wearable Computers.* Zurich, Switzerland, 2001. 31-38.

Rottensteiner, F., and C. Briese. "A new method for building extraction in urban areas from high-resolution LiDAR data." *International Archives of Photogrammetry and Remote Sensing 34(3A).* 2002. 295-301.

Schmalstieg, D., et al. "The Studierstube Augmented Reality Project." *Teleoperators and Virtual Environments, Vol. 11, No. 1.* 2002.

Volsseman, G., and S. Dijkman. "3D building model reconstruction from point clouds and ground plans." In *International Archives of Photogrammetry and Remote Sensing. 34(3W4)*, 37-43. 2001.

Zeibak, R., and S. Filin. "Change detection via terrestrial laser scanning." *International Archives of Photogrammetry and Remote Sensing. 36(3/W52).* 2007. 430-435.