A FRAMEWORK FOR VOXEL-BASED GLOBAL SCALE MODELING OF URBAN ENVIRONMENTS

Joachim Gehrung[a], Marcus Hebel[a], Michael Arens[a], Uwe Stilla[b]

[a] Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB,
76275 Ettlingen, Germany - (joachim.gehrung, marcus.hebel, michael.arens)@iosb.fraunhofer.de
[b] Photogrammetry and Remote Sensing, Technische Universität München, 80333 Munich, Germany - stilla@tum.de

KEY WORDS: Large Scale Modeling, Urban Environments, Probabilistic Occupancy Maps, Volume Rendering

ABSTRACT:

The generation of 3D city models is a very active field of research. Modeling environments as point clouds may be fast, but has disadvantages. These are easily solvable by using volumetric representations, especially when considering selective data acquisition, change detection and fast changing environments. Therefore, this paper proposes a framework for the volumetric modeling and visualization of large scale urban environments. Beside an architecture and the right mix of algorithms for the task, two compression strategies for volumetric models as well as a data quality based approach for the import of range measurements are proposed. The capabilities of the framework are shown on a mobile laser scanning dataset of the Technical University of Munich. Furthermore the loss of the compression techniques is evaluated and their memory consumption is compared to that of raw point clouds. The presented results show that generation, storage and real-time rendering of even large urban models are feasible, even with off-the-shelf hardware.

1. INTRODUCTION

Large scale urban models can be applied for tasks such as city planning, emergency management and other things such as traffic flow or flooding simulations. Most works in the field focus on the generation of 3D city models. These are, however, not considering the inherent long-term challenge of keeping it up to date. Using point clouds as an intermediate representation may have the advantage of easy and fast processing. For a fast changing urban environment, volume based representations such as voxels are more suitable, taking not only their fixed upper memory boundary into consideration but also the ability to model free and unexplored space. Qualities which are important for critical tasks such as efficient storage, change detection and structured data acquisition. Not to forget that the inherent multi-scale representation allows the efficient execution of algorithms - especially rendering.

The generation of city models based on a volumetric representation requires a sound foundation. For this reason, this paper proposes a framework for the volumetric large scale modeling of urban environments. The main contributions of this paper are:

- A framework for modeling and visualizing volume based environments on a global scale, based on a thoroughly selected mix of technologies and algorithms.
- A data quality based algorithm to import range measurements (e.g. point clouds, depth images, ...).
- Both a lossless and lossy compression strategy for storing the volumetric representation to disk.

The paper is organized as follows: In Section 2, a brief overview of the related work is given. Section 3 presents an overview of the proposed framework, while Section 4 and 5 provide further details about the data quality based import algorithm and the proposed compression strategies. The results provided by the framework are shown in Section 6. An evaluation of the results is given in Section 7.

2. RELATED WORK

2.1 Large Scale Modeling

An comprehensive overview of the field is given in (Leberl et al., 2012). Most of the discussed approaches for city model generation utilize either photogrammetric techniques, active sensors or a combination of both. Some approaches like the one proposed by Poullis et al. have fully automated the extracting of polygonal 3D models (Poullis, 2013).

However, volumetric techniques are hardly used in the field. Mulder demonstrated the benefits of a voxel-based model for the repair of geometrically invalid 3D city building models (Mulder, 2015). Garcia-Dorado et al. utilized volumetric grids as an intermediate representation to extract building models from aerial images using a surface graph cut algorithm (Garcia-Dorado et al., 2013). To the knowledge of the authors, to date there is no work utilizing the full capacity of volumetic modeling techniques as a starting point for large scale urban modeling.

2.2 Probabilistic Modeling Techniques

Point clouds could be considered as a first step towards environment modeling, as done by most 3D SLAM systems (Cole and Newman, 2006) (Nüchter et al., 2007). The absence of an upper bound for memory requirements however is a huge disadvantage. Also point clouds lack expression regarding free and unexplored space, further on the measurement uncertainty is not explicitly modeled.

A more sophisticated approach is probabilistic occupancy grid mapping (Moravec and Elfes, 1985) (Roth-Tabak and Jain, 1989). Although there is an upper memory bound, global scale mapping is still not feasible. Furthermore, a lack of multiple resolutions prevents the efficient execution of spatial algorithms. There are other approaches for refining grid mapping such as 2.5D elevation maps (Herbert et al., 1989) and surface maps (Triebel et al., 2006). Due to either major restrictions regarding the expressiveness or other disadvantages they are not further considered in this paper.

As large scale modeling requires a spatial compression, octree based occupancy maps are a worthwhile approach. Early works stored binary occupancy information in the octree cells (Meagher, 1982). Later probabilistic information has been included into the model (Fournier et al., 2007) (Payeur et al., 1997). As with grid maps, there is an upper bound in memory consumption. Due to the compression and multi-resolution properties of octrees, global modeling and efficient algorithms execution are feasible. The probably most advanced octree-based volumetric representation is OctoMap (Hornung et al., 2013). Its key contribution is a nearly lossless model compression strategy based on probability clamping, allowing fast adaptability and pruning of child nodes.

However, as this approach has been developed for autonomous robotics, it is not entirely suited for large scale modeling. Merging maps is not trivial, unless the map origin and resolution are equal and the affected nodes have no child nodes on their own. Jessup et al. (Jessup et al., 2014) solved this by aligning maps, using transformations determined from commonly mapped portions of the environment. Based on these, trilinear interpolation is used for map resampling and merging. Considering the use of a single global reference frame instead of multiple ones, this approach could be further utilized for large scale modeling.

Furthermore, an adaption of the persistence strategy of OctoMap is required. Since loading a complete global model at once is not feasible, ways to subdivide into partial models are necessary. Further optimization can be done by discarding all parts of the model that do not carry useful information, namely the octrees upper branch nodes. The memory requirements for storing a global model on disk can further be reduced by considering advanced compression techniques.

2.3 Volume Rendering

An exhaustive overview over the topic of volume rendering is given in (Meißner et al., 2000). Techniques are distinguished in Indirect and Direct Volume Rendering. The former one requires the extraction of a polygonal iso-surface (e.g. using Marching Cubes (Lorensen and Cline, 1987)), which is rendered with polygon rendering hardware.

The latter category of techniques renders the volume directly, without the intermediate step of surface extraction. Volume ray casting based approaches like Gigavoxels (Crassin, 2011) produce impressive results, however at the expense of GPU based hardware acceleration. 3D Texture-Mapping is also based on a GPU rendering pipeline. This technique involves loading the volume in question into the texture memory, utilizing the texturing process for volume rendering.

Unlike hardware accelerated techniques both Shear-Warping and Splatting can be implemented in software only. The former involves shearing and distorting the voxel grid according to a chosen perspective which allows traversing both image and grid using a scanline-based algorithm (Cabral et al., 1994). The latter one uses a multi-scale-representation of the volume which is rendered with adaptive depth. A popular variation of this algorithm is QSplat. It allows rendering at a fixed frame rate at the expensive of image quality, even in case of complex geometry or low computational resources (Rusinkiewicz and Levoy, 2000).

3. FRAMEWORK

3.1 Capabilities and Architecture

The proposed framework is capable of handling a volumetric environment representation on a global scale. The inherent upper memory boundary of the chosen modeling approach as well as the proposed compression techniques permit efficient storage even of frequently changing areas. Fast processing is ensured by the capability to load only the required parts of the representation. Real-time visualization is permitted due to a Splatting based rendering approach developed for the framework's volumetric representation.

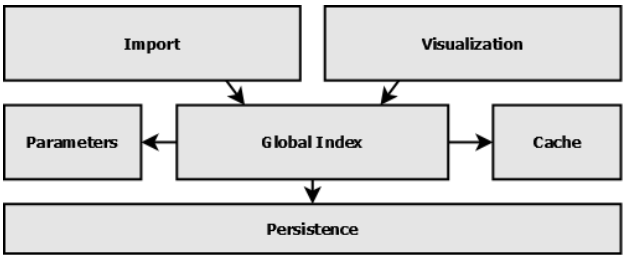


Figure 1. Framework main components overview. The global index handles access to resources required by import and visualization and is in control of parameters, caching and persistence.

The framework is organized as shown in Figure 1. The Global Index coordinates the storage strategy consisting of in-memory caching and long-term storage on disk. It is responsible for selecting the required parts of the representation based on addresses, spatial coordinates and rays. The algorithms for import and visualization access the representation through the Global Index, allowing both import of new measurements and visualization at the same time.

3.2 Volumetric Modeling

The framework is based on the assumption of a global Cartesian reference frame. Therefore it is expected that the agent responsible for data collection is able to geolocate itself respectively its measurements in a sufficiently exact manner. To simplify matters the widely spread ECEF coordinate system was chosen. A cubical bounding box was defined in a way that it completely encloses the earth. Within the bounding box space was tesselated into a fine uniform grid - referred to as *global index* - which grid cells are labelled as *tiles* (see Figure 2).

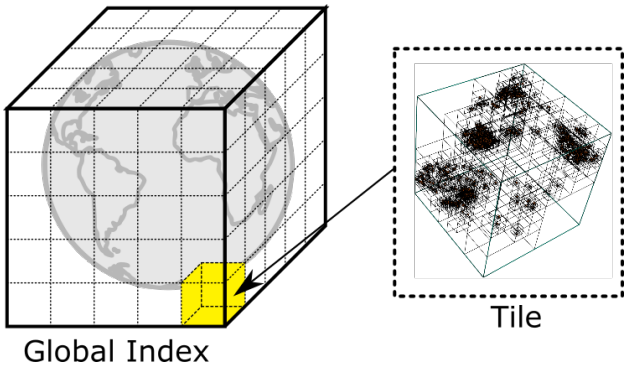


Figure 2. The global index created by tesselating a cubic bounding box around earth into a fine grid. Grid cells are referred to as tiles.

A tile is uniquely addressed by Morton coding (Morton, 1966), whereby addresses can easily be derived from a given point in ECEF coordinates. The space encapsulated within a tile is modelled by a probabilistic occupancy octree based on the theoretical foundation laid by OctoMap (Hornung et al., 2013). This volumetric representation permits the deduction of information about free, solid and unexplored space. An octree node is referred to

as *cell* and can be uniquely addressed by its depth inside the tree and its Morton code. The maximum octree depth determines the maximum spatial resolution as well as the upper memory boundary. Figure 3 shows an exemplary demonstration created from a real world dataset.

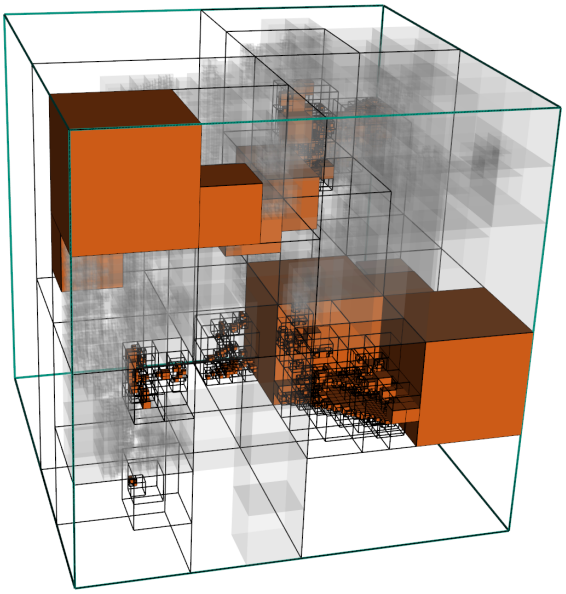


Figure 3. An exemplary demonstration of a tile, created from a real world dataset. Occupied cells are rendered in red, free cells in transparent gray. Unexplored space is not rendered at all.

The benefits resulting from this approach are numerous. Parts of the representation can be loaded, cached and processed dynamically. Only tiles required by the task at hand are initialized which avoids unnecessary memory consumption. The fusion of distinct representations is straightforward, since only overlapping tiles need special handling.

3.3 Model Parameter Selection

The model parameters that need to be selected are the edge length l_{index} of the global index bounding box, the tile edge length l_{tile} as well as d_{tile}, which is the number of the tiles octree hierarchy levels. The calculation is based on the desired approximate tile edge length r_{tile} and the maximum spatial resolution r_{max}. Equation 1 describes the basic geometric properties of an octree.

$$l = r \cdot 2^d \tag{1}$$

As stated above, l is the edge length of the octree's bounding cube and r is the resolution of a single cell at depth d. Rearranging leads to Equation 2, which allows calculation of d_{tile}.

$$d_{\text{tile}} = \left\lceil \log_2 \left(\frac{r_{\text{tile}}}{r_{\text{max}}} \right) \right\rceil \tag{2}$$

Based on d_{tile}, the tiles edge length l_{tile} can be determined using Equation 3.

$$l_{\text{tile}} = r_{\text{max}} \cdot 2^{d_{\text{tile}}} \tag{3}$$

The earth's diameter of approximately $12800\,\text{km}$ is a first starting point for the bounding box edge length, which can be determined by Equation 4.

$$l_{\text{index}} = \left\lceil \frac{12800\,\text{km}}{l_{\text{tile}}} \right\rceil \cdot l_{\text{tile}} \tag{4}$$

For an approximate tile resolution of about 10 meters and a spatial resolution of $0.1\,\text{m}$, $l_{\text{index}} = 12800\,\text{km}$, $l_{\text{tile}} = 12.8\,\text{m}$ and $d_{\text{tile}} = 7$ are determined. These values are considered as the framework's standard parametrization.

3.4 Real-time Visualization

From the four major volume rendering techniques mentioned in Section 2.3, Splatting turned out to fit most to the overall approach of the framework. One of the main reasons is that due to the octree used for volumetric modeling, a multi-resolution data structure is already available. Furthermore the chosen approach doesn't require any hardware support except from OpenGL for basic rendering. It builds entirely on a software implementation, allowing real-time rendering even on a single CPU.

During rendering first all existing tiles inside the view frustum are determined based on its bounding box. The rendering speed depends on the chosen resolution, hence on the depth inside the octree. This resolution is determined at runtime based on the length of the last cycle to permit real-time rendering. Therefore a heuristic is used that exploits the fact that the number of cells to render decreases by the factor of eight for every step up the hierarchy. An iterative refinement step increases the image quality stepwise once the camera stops moving.

4. DATA IMPORT

4.1 Adaption of the Import Algorithm

The algorithm that imports new range measurements into the volumetric representation of the environment is an extension of the one used by the OctoMap framework (Hornung et al., 2013). The utilized ray-based sensor model considers end points as corresponding to surfaces and further assumes the line of sight to be free of obstacles. Based on these assumptions, the occupancy probability of the voxels traversed by a ray is updated.

Due to the proposed space tessellation scheme an additional step for the selection of all tiles traversed by a ray is required. Therefore the tiles related to the start and end point of the ray are calculated. All tiles in between are determined by using a 3D variant of the Bresenham algorithm (Amanatides and Woo, 1987). In a following step another iteration of Bresenham is applied to every tile to determine the free and occupied cells. To avoid unnecessary computations, each ray is cut to fit the tiles' bounding box. Since the addressing of cells is based on Morton coding, addresses outside the bounding box generated due to numerical inaccuracies can easily be filtered out.

After all rays are processed, the occupancy octree of every tile is updated based on the corresponding free and occupied cells. In a final post-processing step, pruning is applied and the probabilities of the branch nodes are recalculated.

Due to the introduction of space tessellation, the parallelization strategy of the algorithm had to be reviewed. Therefore the initial ray based approach had been compared to a tile based one. The latter one first assigns all rays to the tiles traversed by them using multiple threads. Following this, the associated rays are used to directly update the tile's occupancy octree whereby a single thread for every tile is deployed.

Algorithm 1: Import of range measurements.

Data: Global model M, integration depth d_{int}, Observations in form of Vision Rays $O = \{o_1, \ldots, o_n\}$
Result: Updated global model M
begin
  $tiles \longleftarrow \emptyset$
  $cells_{\text{free}} \longleftarrow (\emptyset, \emptyset)$
  $cells_{\text{occ}} \longleftarrow (\emptyset, \emptyset)$
  for $o_i \in O$ **do**
    $T \longleftarrow determineTiles(o_i)$
    for $t_i \in T$ **do**
      $cells_{\text{free}}[t_i] \longleftarrow getFreeCells(t_i, o_i, d_{\text{int}})$
      $cells_{\text{occ}}[t_i] \longleftarrow getOccupiedCells(t_i, o_i, d_{\text{int}})$
    $tiles \longleftarrow tiles \cup T$
  for $t_i \in tiles$ **do**
    for $c_i \in cells_{free}[t_i]$ **do**
      $M \longleftarrow updateFreeCells(M, t_i, c_i)$
    for $c_i \in cells_{occ}[t_i]$ **do**
      $M \longleftarrow updateOccupiedCells(M, t_i, c_i)$
  $M \longleftarrow recompress(M)$

As it turned out the ray based approach performed best regarding runtime and memory consumption. The tile based approach turned out to be a few orders of magnitude slower, likely due to a huge memory consumption that forced the operating system to do memory swapping. For the reasons mentioned above, the ray-based key buffering strategy is considered the approach of choice.

4.2 Data Quality based Import

When processing data from multiple sensors, one also has to deal with the diversity in data quality. As the chosen approach models the environment based on discretization, the quality of the resulting model is ensured by avoiding discretization errors. These of course are high when placing uncertain data in a fine grid and decrease once the grid is replaced by a coarser one. As an octree provides multiple resolutions, the way of choice to reduce discretization errors is the selection of the appropriate depth for the import of new measurements.

One way for assessing data quality is the standard deviation of the overall sensor system. In the context of large scale modeling this not only concerns precision and accuracy of the sensor system, but also of its self-localization. Although this is not trivial to calculate, in most cases it is possible to obtain an accurate estimation.

$$d_{\text{int}} = \min\left(\left\lfloor \log_2\left(\frac{l_{\text{tile}}}{4\sigma}\right)\right\rfloor, d_{\text{tile}}\right) \tag{5}$$

Equation 5 proposes a way to estimate the octree resolution d_{int} for integration based on the standard deviation. Since the octree may contain nodes with a finer resolution, the modified update algorithm from (Jessup et al., 2014) is applied for model update.

5. COMPRESSION

5.1 General Approaches to Compression

The storage of large scale volumetric models is very memory intensive, which implies the application of compression techniques. Therefore, in this section approaches for memory reduction are introduced and two strategies utilizing them are proposed. A point worth considering when developing a compression technique is the inherent octree structure. The number of nodes increases exponentially with every additional layer. As shown in Figure 4, in a fully expanded octree 87.5% of all nodes are in the leaf layer. Taking this into consideration it becomes clear that the largest improvements can be done when reducing memory in the leaf layer. Still, optimizing the branch layer can also be beneficial.

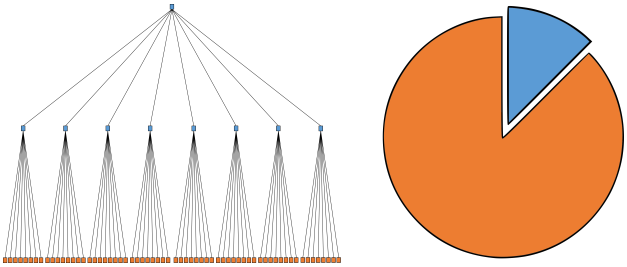


Figure 4. In a fully expanded octree 87.5% of all nodes are leaf nodes, therefore most compression efforts should focus on them.

The information encoded in a serialized node contains its payload in form of a 4 byte occupancy probability, a 3 byte color information and a 1 byte array describing whether or not the eight child nodes exist. As leaf nodes have no descendants, there is no need to store a child array. Therefore a *leaf bit* is introduced. Only if set to zero, a child array is expected. Since a majority of all nodes are leaf nodes, the memory reduction from dropping the child arrays outweighs the introduced overhead. The leaf bit also enables another way of compression. The payload of inner nodes can easily be reconstructed from the leaf nodes, as it is done during the import of range measurements. Consequently it doesn't have to be serialized whenever the leaf bit is zero. These two optimizations lead to a reduction of 15.6% of total memory consumption.

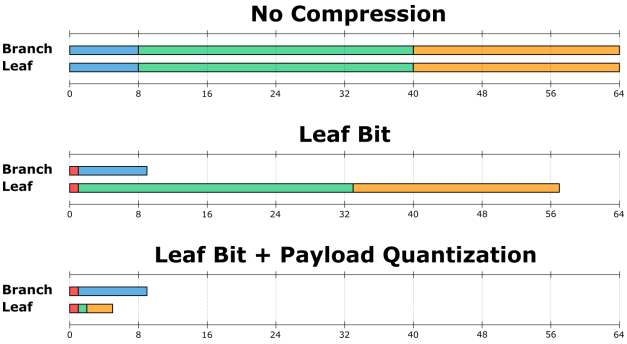
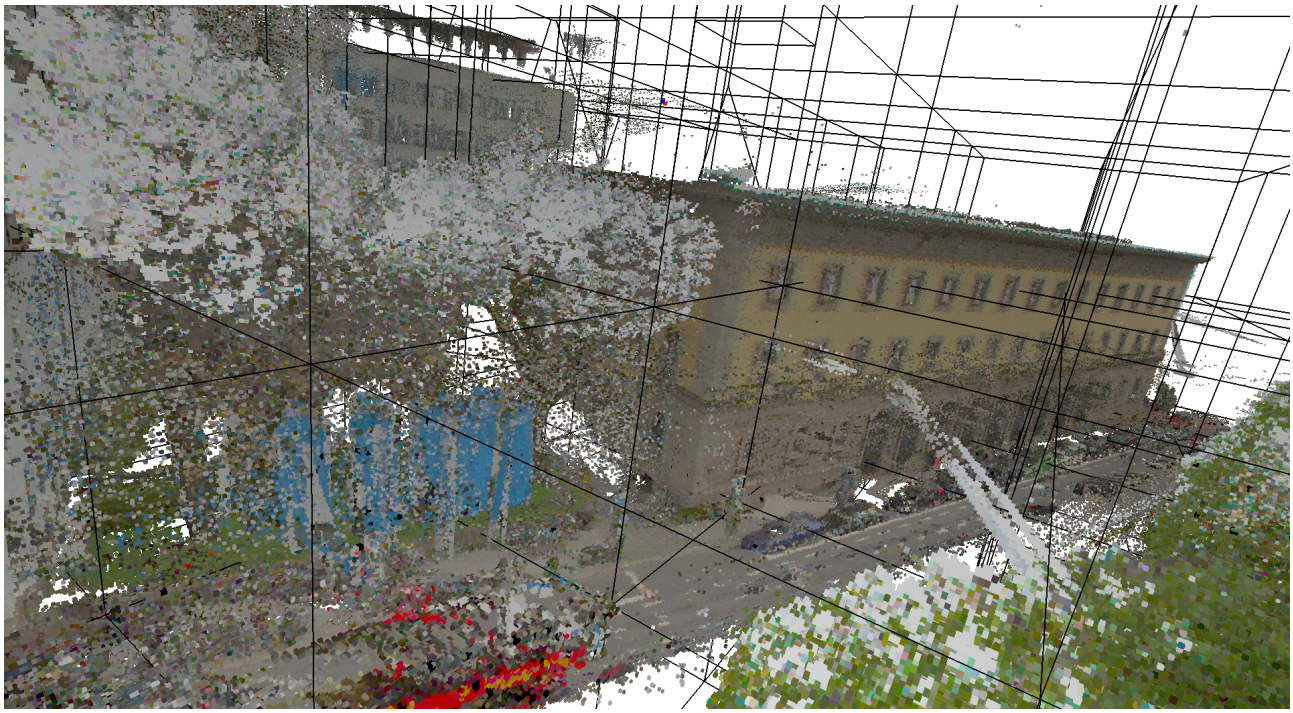


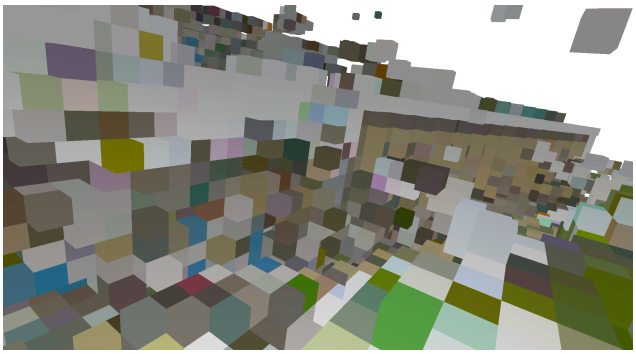
Figure 5. The memory consumption in bits for the uncompressed case, as well as after applying leaf bit and additional quantization, where probability and every color channel are represented by a single bit. Leaf bits are marked in red, child arrays in blue, probabilities in green and colors in orange.

Further compression can be achieved by additionally quantizing the payload. Considering the case that all the color channels and the probability are represented by just a single bit, a total compression rate of 91.4% can be archived. Since this process is lossy, the total loss of information is evaluated in Section 7. Figure 5 illustrates the memory reduction achieved by applying both the leaf bit and quantization technique.

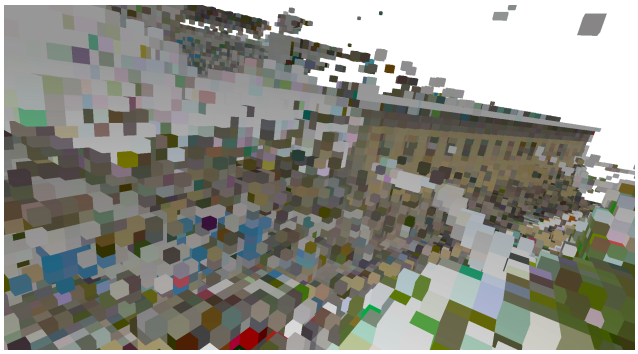
Based on the introduced techniques two compression strategies are proposed. The lossless strategy utilizes the leaf bit to waive the child arrays in the leaf nodes as well as the payload in the



(a)

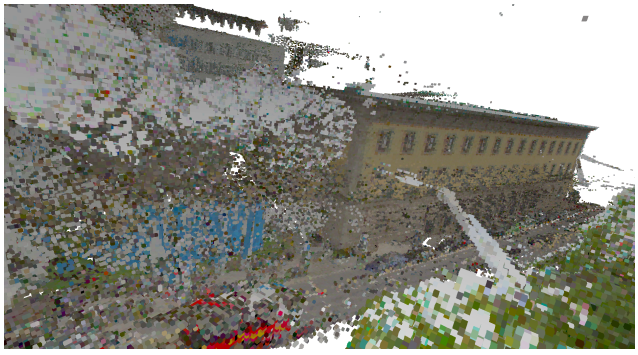


(b)



(c)

(d)



(e)

Figure 6. (a) The volumetric model created from the TUM dataset, visualized at a 10 cm resolution including tile bounding boxes. (b)-(e) The scene rendered at octree depths of 4, 5, 6 and 7. The corresponding resolutions vary from 80 cm to 10 cm.

branch nodes. Extending this approach by quantizing the probability leads to the lossy strategy. A detailed evaluation of the proposed compression strategies is done in Section 7.

6. EXPERIMENTAL RESULTS

The proposed framework has been tested using a mobile laser scanning dataset containing the inner-city campus of the Technical University of Munich (TUM) as well as two art museums and the urban area in the vicinity. The sensor vehicle utilized for this task is equipped with two Velodyne HDL-64E laser scanners located in a 25 degree angle on the front roof to be able to acquire both road and building facades. A Ladybug5 spherical camera is used for texturization. The navigational system is an Applanix POS LV encompassing two GNSS antennas, an inertial measurement unit and a distance measuring indicator. The navigational data was post-processed to increase accuracy.

Since only the part of the model that is currently worked with is loaded into memory, the critical factors for testing the capabilities of the framework are spatial resolution and geometric complexity. The TUM dataset fulfils both criteria. The area covered by it has a size of $0.2\,\mathrm{km}^2$. However, it consists of more than 35000 high resolution mobile laser scans, including a total amount of about 715 million points. The density of the accumulated point cloud varies from a few hundred to a few thousand points per square meter, mostly depending on the vehicle speed and the number of pass-bys.

All reported results were generated on a machine with $32\,\mathrm{GiB}$[1] of RAM and an Intel Core i7 processor with $3.5\,\mathrm{GHz}$ and 12 cores. The volumetric model generated from this dataset consists of 4619 tiles with an inner resolution of $10\,\mathrm{cm}$. The import of the complete dataset (using the standard parameters determined in Section 3.3) completed in $21\,\mathrm{h}40\,\mathrm{min}$ including free space traversed at a resolution of $80\,\mathrm{cm}$ to prevent fragmentation, utilizing all available cores. Time can be reduced dramatically to $37\,\mathrm{min}$ when only considering surface points. This corresponds roughly to the driving time spent to record the dataset. A reasonable large cache size was chosen to avoid delays.

Depth	Rendering	Real-time
0	3 ms	✓
1	9 ms	✓
2	19 ms	✓
3	47 ms	✓
4	165 ms	–
5	804 ms	–
6	3402 ms	–
7	10820 ms	–

Table 1. The rendering times of each octree depth for a frame rate of 10 Hz.

Figure 6a shows the volumetric model generated from the dataset. The Figures 6b, 6c, 6d and 6e show the same scene for different resolutions. The corresponding rendering times are presented in Table 1. Rendering was done without parallelization, using a field of view with a range of $100\,\mathrm{m}$. The results show that real-time rendering is possible at a resolution of $1.6\,\mathrm{m}$.

[1]Please note that in this paper memory is addressed using IEC symbols. GiB corresponds to Gigabyte and is defined as 2^{30} byte.

7. EVALUATION

7.1 Information Loss due to Compression

Due to the sensor model the probability mass is not equally distributed over the entire interval. Therefore, applying the lossy compression results in an information loss that depends on the number of bits used for encoding. The color channels were neglected in this consideration and fixed at 8 bit per channel. To determine the loss, a tile with representative features has been stored in both the compressed an uncompressed format. Both were reimported and the values of their cells were compared. It should be stated that no pruning was applied at the compressed tile to preserve the tree structure. Since all probabilities within the framework are based on log odds, the mean log odd discrepancy was chosen as a measure for the actual loss of information.

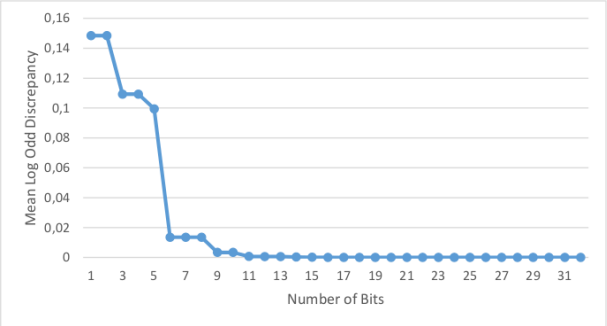


Figure 7. The mean log odd discrepancy in relation to the number of bits used for encoding is used to illustrate the information loss due to probability quantization.

The experimental results for encodings from 1 to 32 bit are shown in Figure 7. As can be seen already a 6-bit encoding results in a very low discrepancy from the original value. Considering the trade-off between accuracy and memory requirements, a 9-bit encoding resulting in a compression rate of 80.5% is the optimal choice.

7.2 Compression Rates

In this section the compression of both the volumetric representation as well as both compression algorithms are analyzed. Therefore the two representations mentioned in Section 6 were utilized. Their memory consumption for the generated representation are shown in Figure 8, whereas the lossy compression is based on the 9-bit encoding determined before. The point clouds used for generation have a size of $17.1\,\mathrm{GiB}$.

As can be seen, the memory consumption of even the largest model is an order of magnitude smaller than the point cloud. This demonstrates the inherent compression capabilities of the volumetric representation. The actual compression rates of both compression approaches turned out to be even higher than expected, a few percent for the lossy compression and about 15% for the lossless one. This may be due to the environment geometry and applied pruning.

8. CONCLUSION AND FUTURE WORK

In this paper, we presented a novel framework for Large Scale Modeling of arbitrary environments. In our approach, space is tessellated into tiles that are stored within a global uniform grid. A tile models the environment within its boundaries utilizing a

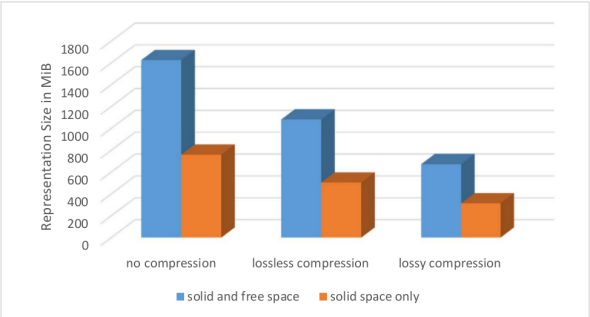


Figure 8. The requirements of disk memory for the representation generated from the TUM dataset. Memory consumption of the uncompressed format as well as lossless and lossy compression are shown.

propabilistic volumetric approach. Beside the framework we proposed a quality based algorithm to import range measurements into the global index. Furthermore a lossless and lossy compression technique were presented. We evaluated our framework with a real-world data sets. The loss of information based on the number of bits used for the lossy compression was evaluated and the compression rates for the modeling approach itself as well as both compression techniques were evaluated.

Our work has shown that large scale modeling can benefit from a volumetric representation. As shown in context of the real-time visualization, the inherent multi-level representation permits fast and efficient execution of algorithms. The overall memory consumption is less than that of a comparable point cloud. Importing only surface points is roughly possible in real-time, yet considering free space leads to a dramatic increase of the runtime. In the future, besides improving the framework we will focus on the extraction of static environment elements for an efficient change detection. Based on this we hope to achieve better results for the construction of 3D models of buildings and other urban structures.

REFERENCES

Amanatides, J. and Woo, A., 1987. A Fast Voxel Traversal Algorithm for Ray Tracing. In: *Eurographics '87*, pp. 3–10.

Cabral, B., Cam, N. and Foran, J., 1994. Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. In: *Proceedings of the 1994 Symposium on Volume Visualization*, VVS '94, ACM, New York, NY, USA, pp. 91–98.

Cole, D. M. and Newman, P. M., 2006. Using laser range data for 3D SLAM in outdoor environments. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1556–1563.

Crassin, C., 2011. GigaVoxels: A Voxel-Based Rendering Pipeline For Efficient Exploration Of Large And Detailed Scenes. PhD thesis, Grenoble University.

Fournier, J., Ricard, B. and Laurendeau, D., 2007. Mapping and Exploration of Complex Environments Using Persistent 3D Model. In: *CRV*, IEEE Computer Society, pp. 403–410.

Garcia-Dorado, I., Demir, I. and Aliaga, D. G., 2013. Automatic urban modeling using volumetric reconstruction with surface graph cuts. *Computers & Graphics* 37(7), pp. 896–910.

Herbert, M., Caillas, C., Krotkov, E., Kweon, I. S. and Kanade, T., 1989. Terrain mapping for a roving planetary explorer. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '89)*, Vol. 2, pp. 997–1002.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C. and Burgard, W., 2013. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Autonomous Robots*.

Jessup, J., Givigi, S. N. and Beaulieu, A., 2014. Merging of octree based 3D occupancy grid maps. In: *2014 IEEE International Systems Conference Proceedings*.

Leberl, F., Meixner, P., Wendel, A. and Irschara, A., 2012. Automated photogrammetry for three-dimensional models of urban spaces. *Optical Engineering*.

Lorensen, W. E. and Cline, H. E., 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, ACM, New York, NY, USA, pp. 163–169.

Meagher, D., 1982. Geometric modeling using octree encoding. *Computer Graphics and Image Processing* 19(2), pp. 129–147.

Meißner, M., Pfister, H., Westermann, R. and Wittenbrink, C., 2000. Volume Visualization and Volume Rendering Techniques. In: *Tutorials 6, Eurographics 2000*.

Moravec, H. and Elfes, A., 1985. High Resolution Maps from Wide Angle Sonar. In: *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pp. 116 – 121.

Morton, G., 1966. A computer oriented geodetic data base and a new technique in file sequencing. Technical report, IBM Ltd., Ottawa, Ontario, Canada.

Mulder, D., 2015. Automatic repair of geometrically invalid 3d city building models using a voxel-based repair method. Master's thesis, Delft University of Technology.

Nüchter, A., Lingemann, K., Hertzberg, J. and Surmann, H., 2007. 6D SLAM - 3D mapping outdoor environments: Research Articles. *J. Field Robot.* 24(8-9), pp. 699–722.

Payeur, P., Hebert, P., Laurendeau, D. and Gosselin, C. M., 1997. Probabilistic octree modeling of a 3D dynamic environment. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1289–1296.

Poullis, C., 2013. A Framework for Automatic Modeling from Point Cloud Data. *IEEE Trans. Pattern Anal. Mach. Intell.* 35(11), pp. 2563–2575.

Roth-Tabak, Y. and Jain, R., 1989. Building an environment model using depth information. *Computer* 22(6), pp. 85–90.

Rusinkiewicz, S. and Levoy, M., 2000. QSplat: A Multiresolution Point Rendering System for Large Meshes. In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, ACM, pp. 343–352.

Triebel, R., Pfaff, P. and Burgard, W., 2006. Multi-Level Surface Maps for Outdoor Terrain Mapping and Loop Closing. In: *In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '06)*, IEEE, pp. 2276–2282.