Surface Reconstruction from Arbitrarily Large Point Clouds

Thomas Wiemann, Isaak Mitschke, Alexander Mock, Joachim Hertzberg

Knowledge Based Systems Group
Institute of Computer Science
University of Osnabrück
Osnabrück, Germany
Email: twiemann@uni-osnabrueck.de

***Abstract*—Generating 3D robotic maps from point cloud data is an active field of research. To handle high resolution data from terrestrial laser scanning to generate maps for mobile robots is still challenging, especially for city scale environments. In this short paper, we present the results of an approach for surface reconstruction from arbitrarily large point clouds. To achieve this, we serialize the large input data into suitable chunks, that are serialized to a shared hard drive. After computation, the partial results are fused into a globally consistent reconstruction.**

I. Introduction

Today it is possible to scan large outdoor environments with 3D laser scanners in short time and high precision. Terrestrial laser scanners can measure objects from large distances – typically hundreds of meters – with sub-centimeter accuracy. Mounted on mobile platforms, the acquired point clouds can be aligned automatically by combining information from inertial sensors and satellite based global positioning with SLAM algorithms [11]. This allows to digitize large areas in short time. However, in robotic applications like path planning and autonomous exploration [2], [1], the bare size of the point clouds prohibits the use of the raw input data for such purposes.

To make the carried geometric information accessible for computations on platforms with limited resources, the input data has to be converted in a representation that can be handled algorithmically. Our approach to achieve this goal is to compute polygonal representations of the scanned environments. Polygonal meshes represent the environment in a compact but precise continuous representation and can be used for path planning [6], tracking and localization [12] or autonomous exploration. Some preliminary results based on the approaches presented in [6] are shown in Fig. I. Here, a mesh reconstructed from a terrestrial laser scanner mounted on a mobile robot, was used estimate derivable surfaces (pink) and to compute next best view poses for further exploration (green markers).

Surface reconstruction is based on the well known Marching Cubes algorithm [4], which requires that the scanned volume is subdivided into a regular voxel grid, as the triangulation is computed locally per voxel. For our setup, a size of 5 cm per voxel has proven to deliver accurate and compact results [9].

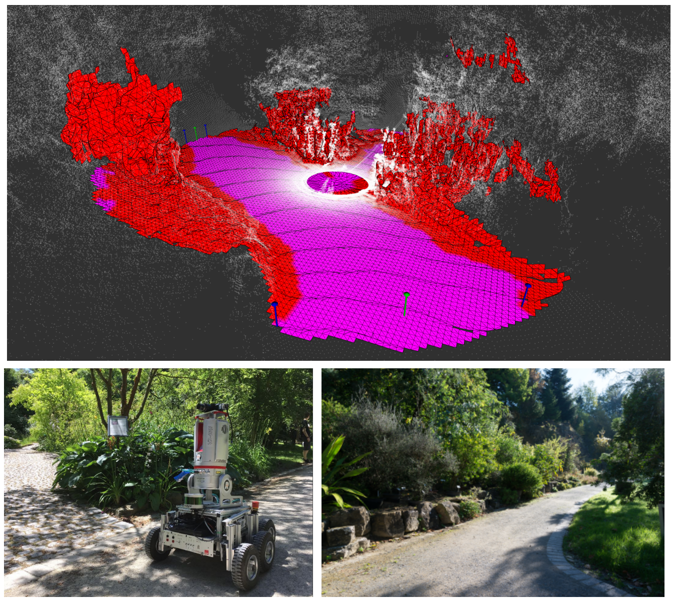

Fig. 1. Top: A triangle mesh used for path planning in unknown environments. The surface was reconstructed from a part of a 3D point cloud (white) taken with a terrestrial laser scanner mounted on a mobile robot. Driveable surfaces (pink) are classified via roughness estimation based on the triangle normals [6]. New scan positions for further exploration are indicated by the green markers. Bottom: The robot with laser scanner and a photo of the scanned environment.

However, in city scale scenarios it is not feasible to generate voxel grids with that resolution at once, even if state of the art methods like spatial trees or hashing are used, since the number of generated voxels would break the memory limit of current personal computers. In this paper we present exemplary results of an approach for large scale reconstruction that splits the input data into chunks that can either be computed sequentially on a single computer (at the cost of higher run time) or in parallel in a computing cluster via MPI. It is implemented in the Las Vegas Surface Reconstruction Toolkit. Software and data sets are freely available the the project website [8].

II. Related Work

For most purposes, octrees [5] are the standard data structure to manage voxels and have been successfully applied

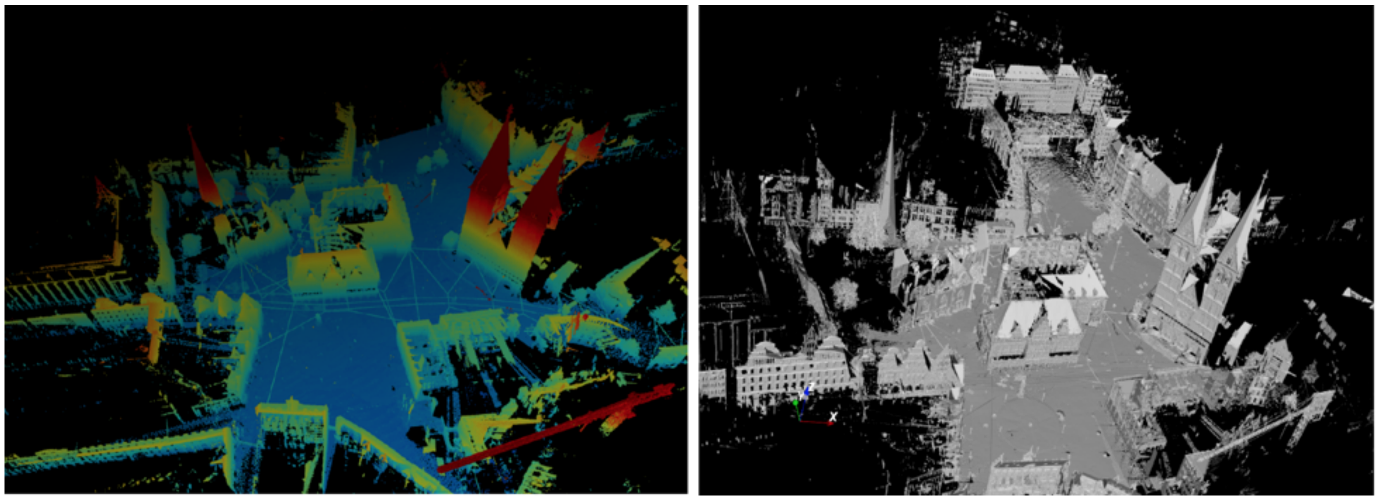

Fig. 2. The Bremen City data set was used to benchmark our reconstruction procedure. The input point cloud is shown on the left with elevation above ground rendered in a blue to red color gradient. The image on the right shows the complete polygonal reconstruction with 5 cm voxel size.

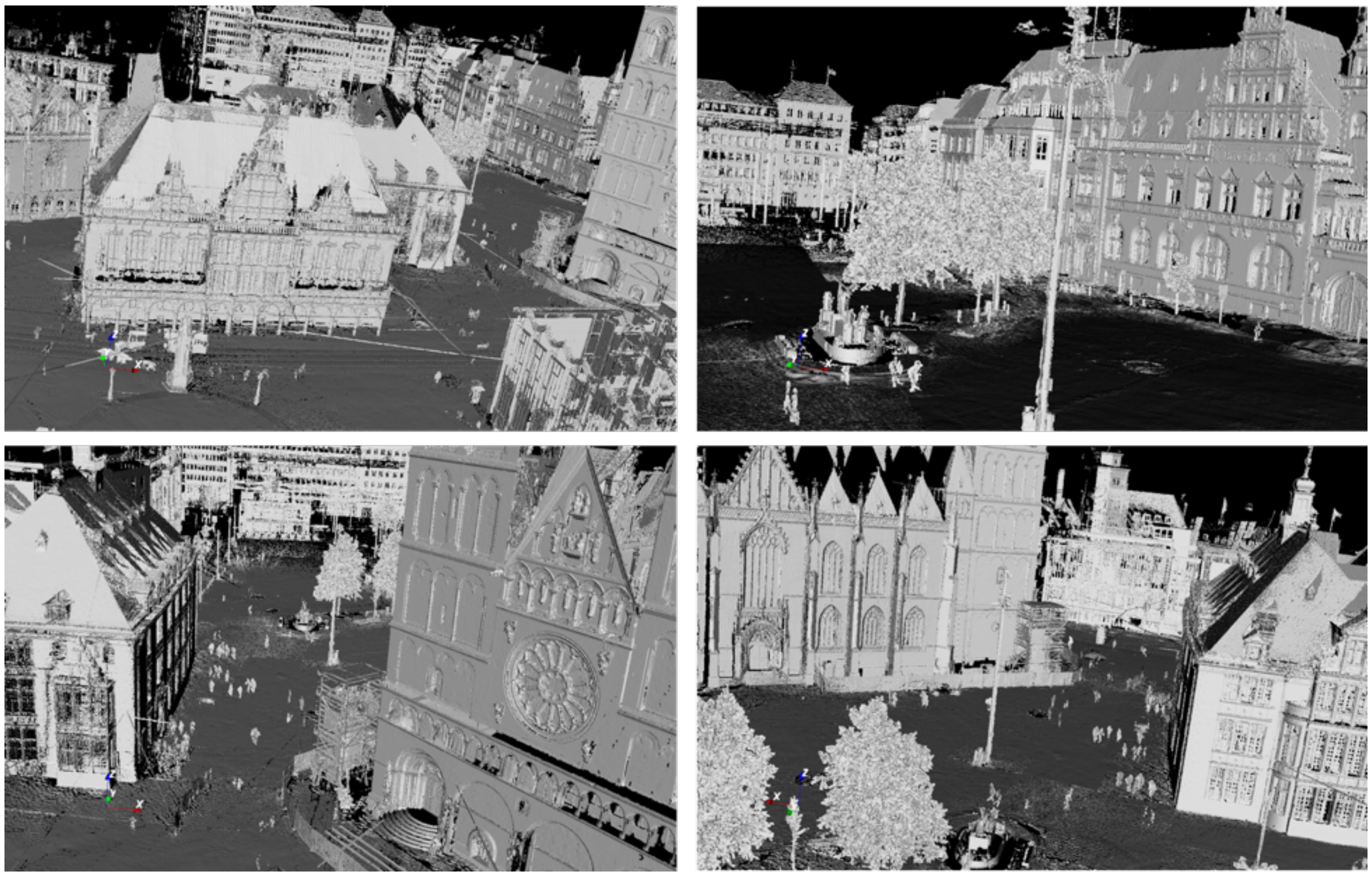

Fig. 3. Detailed views from different angles at the final reconstruction. Chunking and re-integration did not cause any visible artifacts.

in surface reconstruction. In principle, such a representation could be used for reconstruction and extended to support partition of the data, but the main problem with octrees in this context is that it is quite tedious to search for adjacent voxels due to the overlying tree structure. Finding adjacent voxels however is helpful to avoid the creation of redundant vertices during reconstruction [10]. To overcome this, we use a simple collision free hash function to manage voxels in a hash map. Our hash function allows to find adjacent cells directly if their position within the grid is known.

Triangulation within the voxels is done via Marching Cubes using Hoppe's signed distance function [3] (SDF). This requires efficient search for the k nearest points (kNN search). In our software, we implemented a GPU-accelerated nearest neighbor search based on the work of Qiu et al. [7]. This allows us to compute surface normals for the scan points on the GPU, which speeds up the computations significantly.

III. Large Scale Reconstruction

Large scale reconstruction is realized using several processes that run concurrently either on a single computer or in a computing cluster with a shared data storage and a certain amount of RAM per process. For inter process communication, we use the well known MPI standard. The data is serialized into partitions on the shared storage into geometrically coherent chunks. These partitions are then sent to the slave nodes, which perform the necessary computations on the their assigned parts and on GPUs if available.

After partitioning, the chunks are sent to the slave nodes for reconstruction. Now, the the actual cells in the grid holding the signed distances are computed. Each slave loads the assigned chunk from the hard disk and instantiates the cells within its local grid, which are managed via spatial hashing. For each sub-grid, the SDF values for the vertices are either computed classically on the CPU or – since we have pre-partitioned chunks that can be configured to fit into the memory of a GPU – on a graphics card.

IV. Experiments

We performed several experiments to evaluate our distributed reconstruction. First, we present qualitative results to show that the global fusion produces consistent reconstructions. In the second part of the evaluation we analyze the run time and memory consumption on standard PCs and a cluster of several high performance computers.

A. Qualitative Results

Fig. 2 shows an exemplary data set that we used for evaluation. The raw point cloud is rendered in the left image. This data set was taken with a Riegl VZ400i terrestrial laser scanner on the market place in Bremen, Germany and contains a total of 214 million data points. The extension of the measured volume was approximately $300\,m \times 500\,m \times 300\,m$. The reconstructed triangle mesh is shown on the right image in Fig. 2. Reconstruction was done using a cell size of 5 cm. The resulting triangle mesh consists of 10 million triangles.

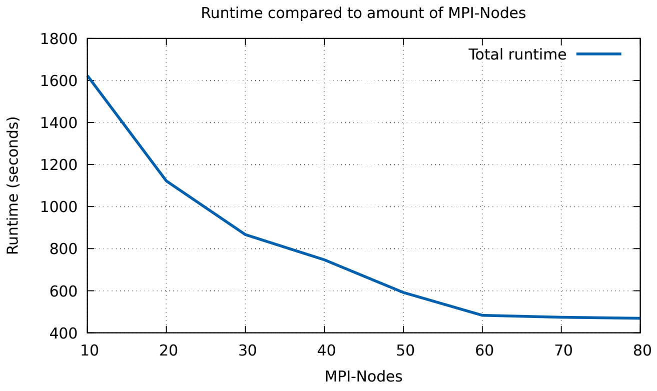

Fig. 5. Reconstruction time for the Bremen City data set with increased number of slave nodes.

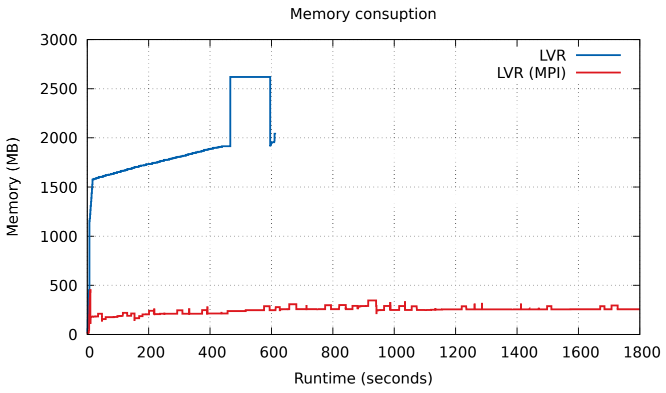

Fig. 6. Memory trace of the reconstruction without segmentation (LVR) and the new approach. Note that LVR finished after approximately 600 s, while the distributed version needed 1800 s, but only used a predefined amount of memory.

Details from the reconstruction are shown in Fig. 3. With the chosen voxel size of 5 cm the reconstruction preserves most details from the original point cloud data while reducing the memory consumption drastically. In the final mesh, there are no visible artifacts that can be ascribed to the initial partition of the data.

B. Run Time Analysis

First, we benchmarked the time for distributed normal estimation using only CPUs and compared it to the GPU implementation. The results are shown in Fig. 4. It shows the time spent for tree preparation and actual normal estimation. The run time on the GPU is several order of magnitudes lower than the CPU-based search.

To evaluate the scaleability of our approach, we benchmarked it on a cluster computer consisting of three Dell Power edge R530 servers with 2 Intel Xeon E5-2680 CPUs with 28 cores and 192 GB RAM each that were connected via ethernet in a local network, totaling in a cluster with 84 MPI nodes. In this setup, each node had approximately 6 GB RAM available for reconstruction. In the experiment, we used the full resolution version of the Bremen City data set and reconstructed it with a voxel size of 5 cm and increased the number of used slave nodes. The results of this experiment are shown in Fig. 5. In the beginning, the run time scales well with the number of added slave nodes. With increasing number of slave nodes, the efficiency decreases. This is mainly due to the increased I/O operations on the shared data volume and the increasing network traffic that is generated when more data is sent between the master node and the slaves. This is an effect that is typical for this kind of simple cluster layout.

C. Memory Consumption

In a last experiment we compared the memory consumption of the old sequential approach implemented in LVR on the desktop PC with GPU. Fig. 6 displays the memory consumption over time on the data set reduced to 80 million points. This size was chosen, because it is the maximum number of points that can be handled on the used GPU. The evaluation clearly shows, that the memory needed for reconstruction is drastically reduced when the data is split up. However, the overall run time nearly triples due to the segmentation process and I/O overhead. But the memory consumption remained almost constant using pre-segmentation. This clearly indicates, that our approach can in principle handle arbitrarily large point clouds.

V. Conclusion

In this paper we presented an approach to compute polygonal reconstructions from arbitrarily large point clouds. The approach was designed to allow parallel computation on MPI clusters as well standard personal computers. To further speed up the computation, we implemented a GPU based normal estimation approach that can be integrated into the reconstruction process. Our approach mainly concentrates on overcoming the RAM limitation that usually occurs when handling large scale point cloud data on standard computers.

The aim of this research is to generate polygonal maps that can be used on mobile robots. At this point, we are able to generate such maps from high resolution point clouds of city scale environments. Although the computed triangles meshes are a much compacter representation than the initial point cloud data, the generated meshes still contain far more triangles than necessary to represent the environments. Hence we plan o further compress the representation, by integrating mesh optimization algorithms into the map generation pipeline.

References

[1] M. Al khawaldah and A. Nüchter. Multi-Robot Cooperation for Efficient Exploration. *AUTOMATIKA – Journal for Control, Measurement, Electronics, Computing and Communications*, 55(3):276–286, 2014.

[2] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on robotics*, 21(3):376–386, 2005.

[3] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics*, 26(2), 1992.

[4] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *ACM SIGGRAPH '87*, 1987.

[5] Donald Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129 – 147, 1982.

[6] S. Pütz, T. Wiemann, J. Sprickerhof, and J. Hertzberg. 3d navigation mesh generation for path planning in uneven terrain. *IFAC-PapersOnLine*, 49(15):212 – 217, 2016. 9th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2016.

[7] D. Qiu, S. May, and A. Nüchter. GPU-accelerated Nearest Neighbor Search for 3D Registration. In *Proceedings of the 7th International Conference on Computer Vision Systems (ICVS '09)*, number 5815 in LNCS, pages 194–203, October 2009.

[8] T. Wiemann. The las vegas surface reconstruction toolkit, 2014. http://www.las-vegas.uni-osnabrueck.de.

[9] T. Wiemann, H. Annuth, K. Lingemann, and J. Hertzberg. An extended evaluation of open source surface reconstruction software for robotic applications. *J. Intelligent and Robotic Systems*, 77(1):149–170, 2015.

[10] T. Wiemann, M. Mrozinski, D. Feldschnieders, K. Lingemann, and J. Hertzberg. Data handling in large-scale surface reconstruction. In *Intelligent Autonomous Systems 13*, pages 499–511. Springer, 2016.

[11] O. Wulf, A. Nüchter, J. Hertzberg, and B. Wagner. Ground truth evaluation of large urban 6d slam. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 650–657, Oct 2007.

[12] J. Wülfing, J. Hertzberg, K. Lingemann, A. Nüchter, T. Wiemann, and S. Stiene. Towards real time robot 6d localization in a polygonal indoor map based on 3d tof camera data. *IFAC Proceedings Volumes*, 43(16):91 – 96, 2010. 7th IFAC Symposium on Intelligent Autonomous Vehicles.

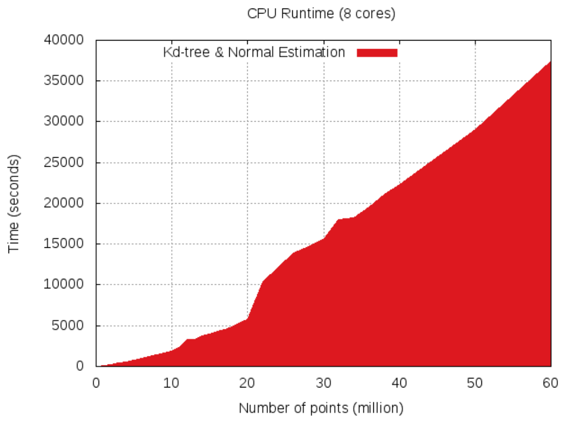
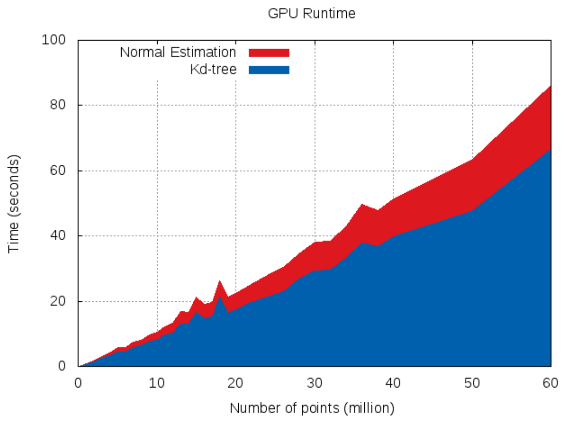

Fig. 4. Run time comparison for normal estimation on CPU (left) and GPU (right). Normal estimation was done using 200 nearest neighbors. The time spent for tree generation on the whole process is indicated in blue. Note that the time spent on tree generation on the CPU was too small to be visible in the plot.