Real-time Volumetric Surface Reconstruction

Seminar: Computer Graphics
Prof. Dr. Winfried Kurth
WS 16/17

Institute of Computer Science
Georg-August-Universität Göttingen

Author:Mian Athar Naqash
MatriculationNumber: 21570550
Email: mianathar.naqash@stud.uni-goettingen.de
Course Applied Computer Science (Master, 3rd semester)
Date:June 20, 2017

Original Article: Scalable Real-time Volumetric Surface Reconstruction
Original Authors: Jiawen Chen, Dennis Bautembach, Shahram Izadi

Abstract

The fundament challenges of scalability faced in the methods of volumetric reconstruction of real time objects are discussed in the original paper. The work in this paper is mainly based on the original article i.e. Scalable Real-time Volumetric Surface Reconstruction by Jiawen Chen, Dennis Bautembach, Shahram Izadi. According to [1] a data structure for graphic hardware and live reconstruction of large scale scenes is designed which is also memory efficient. The data structure takes overlapping depth maps which are then merged into a single volumetric representation of a moving object. The process is a two way process between GPU and host for streaming data. The main process is then divided in two sub-processes. The 1st process is to take the depth maps which are taken before processing, the 2nd is the estimation of camera pose and the last is volumetric fusion and extraction of surface. According to [1], a shallow hierarchy with a factor having large branching gives good memory improvement, certain experiments are carried out which show that this data structure actually consume less memory than a regular grid. A comparison between implementation of the data structure in the original paper and the existing methods is done and also a higher quality reconstruction on different scenes which are captured in real time scenario is demonstrated (see [1]). It is a study about the creation of high level large scale scenes reconstruction with fine geometric details with efficient memory and hierarchical data structure. A moving depth camera is used that fuses overlapping depth maps by the use of a sparse data structure and creates a single volumetric representation, from which detailed surface models are extracted.

Contents

1 Introduction **3**
1.1 Objectives of this research 4

2 Related work **5**
2.1 Volumetric fusion 5
2.2 Reconstruction from 2D images 5
2.3 Real-Time reconstruction using active sensor 6
2.4 Extending Kinect Fusion 7

3 3D Reconstruction Pipeline **7**

4 GPU Implementation **9**
4.1 Memory Layout 10
4.2 Integration 11
4.3 Summarization 11
4.4 Raycasting 11

5 Moving volumes and streaming **12**

6 Result **13**
6.1 Limitation and future work 15

7 Summary **15**

1 Introduction

The surface reconstruction problem is always under the focus in the field of Computer Graphics. It can be used in multiple applications for cultural heritage, special effect and especially for gaming it is very important. The reconstruction process takes multiple overlapping, depth measurement and objects' measurements or even a complete scene as input, the process performs different mathematics operations and returns a single 3D representation of that 2D image or object. The depth can be calculated from 2D images with the help of structure-from-motion (SfM) techniques (see [2]). Multi-View stereo or active sensors such as laser scanner or depth cameras can also be used to find out the depth of an image (see [3]).
According to the original article [1], one of the well-known surface reconstruction approaches is the volumetric method (see [4]). As compared to other methods this method uses relatively simple fusion method and gives high quality output results. This approach has the following points:

- It doesn't make any assumption about the surface topology.
- Uses the redundancy of overlapping depth samples.
- Captures the uncertainty of depth estimates.
- Fills small holes but leaves unobserved regions empty.

The introduction of depth cameras in the field of computer graphics and computer vision leads the researches' interest in the applications of real time surface reconstruction, which can be used in the following (see [1])

1. Augmented reality: Here the need is to combine the real world geometry with the virtual actions of a user.
2. Autonomous guidance: Where a robot is supposed to reconstruct the environment, get some data and respond back to the physical environment.

The reconstruction method adopted by Kinect Fusion - "KinectFusion provides 3D object scanning and model creation using a Kinect for Windows sensor" [5] - compelled live reconstruction from noisy Kinect depth maps and these were applied to a variety of interactive scenarios. The data structure that is used in the original volumetric method by Curless and Levoy [4], is a regular 3D

grid that is uniformly divided into a set of voxels (volume elements). These voxels are mapped on predefined physical dimensions. There is a problem in this data structure i.e. it puts a lot of load on the memory because the surface geometry and free spaces are all densely represented.
This densely represented data structure creates a problem in the world of volumetric reconstruction and that is the scalability problem which means to create high definition models with fine details without the loss of frames for the purpose of saving memory. The various issues related to the problem and their solutions mentioned by the original article are given as (see [1]):

1. The designing of a hierarchical GPU data structure which is not only fast and compact but is also able to update and support the live Kinect depth map's fusion. In this way a high resolution representation can be achieved by the use of comparatively less memory.

2. A mechanism is proposed that is used for the streaming of the proposed data-structure between the host and the GPU.

According to [1], the best memory and speed improvement can be achieved by using a shallow hierarchy of regular grids even with the same hardware. The original article demonstrates the reconstruction of different scenes taken from both indoor, out-door locations where the light provided was only the natural light. The experiments were carried out by using a Kinect camera having more speed, quality and scale as well. There is also a comparison between different methods that improved the reconstruction quality.

1.1 Objectives of this research

The issues that are stated above can be solved by achieving the following:

- A fast and compact hierarchical GPU data structure, capable of dynamic update, supporting fusion of live Kinect depth maps and rendering of surface in real-time. This increases the physical size and resolution of the reconstruction volume using an order of magnitude less memory than a regular grid.

- A mechanism for losslessly streaming subsets of our data structure between GPU and host, decoupling the active volume from a predefined physical space.

2 Related work

A surface is basically the result of creation of a single 3D model from different noisy measurements. Different approaches are used in general but they have some problems for example, they don't give any information about the process of capturing the objects, and also disregard the uncertainty in measurements. These informations are very important when it comes to the laser-range-sensors and depth cameras, especially for triangulation based methods because of achieving higher quality reconstruction.
Different methods are used to deal with such sensors that implement the techniques of combining different overlapping depth measurements into a single 3D representation but to discuss all these methods is out of the scope of this paper, however, these methods are partially discussed in the original paper (see [1]).

2.1 Volumetric fusion

The problems discussed above have a high impact on the process of reconstruction that is why different methods exist to solve these problems each with its own pros and cons. Some of these methods use volumetric data structures such as occupancy or samples of continuous functions. Occupancy generateds binary grids from multiple range images then these similar properties are shared with voxel carving (see [6]). In most cases only a low quality can be achieved (see [1]).
Different steps and techniques are used for the volumetric fusion. For example a representation of the surface related signed distance field is used. According to [7], the use of signed distance fields for the representation of surfaces for the physical simulation is very common. Curless and Levoy worked with the depth map fusion and dealt with the direction of a sensor in a noisy environment. Each depth map is converted into signed distance field and then converted into voxel grid. Furthermore the method of iso-surface or ray casting is used to get the zero level-set (see [1]).

2.2 Reconstruction from 2D images

This method is mostly used to reconstruct a large outdoor scenes. Because in the outdoor environment most of the scenes are captured with the help of 2D cameras. MVS or SfM techniques are used to create depth maps from the

passive 2D cameras. Because we don't have active sensors here, the depth estimation can give a result having non-systematic noise and also outliers and because of this most of the systems use some steps to regularize depth maps by testing photo-consistency, visibility, and shape priors of the image (see [3]). Some explicit spatial regularization is added to the method of Curless and Levoy (see [4]) which re-casted the problem, it gives impressive results but with the loss of memory and speed.
According to [1], the use of the above discussed system gives a very low quality depth map but the expense of depth estimation and fusion results in comparatively equal quality, speed, and scale however for this process a readily available active depth camera is used. This enables a wider applicability of subset of outdoor scenes, but still, reconstruction can be done in natural lighting. In the design of volumetric data structures scale, quality and speed can be demonstrated.

2.3 Real-Time reconstruction using active sensor

Some researchers left the scaling problem completely because of the trad-off between scale quality and speed and they started their research on the reconstruction of smaller scenes and objects by the use of active sensors.
The handheld object is scanned by a structured light sensor. Using a variant of ICP the algorithm first aligns the point cloud, creates a voxel grid from samples and renders. By the implementation of the Curless and Levoy algorithm, a high-quality surface reconstruction can be achieved. However, to scan a single small object, a low resolution hand-held and ToF camera was used by Cui et al. (see [8]).
This method doesn't focus on high-quality reconstruction but rather deals with robust localization, loop closure and correction for model drift (see [1]). KinectFusion used a modified version of Curless and Levoy to fuse noisy Kinect depth maps. The surface is stored as a regular voxel grid in the graphical memory. It is extracted by raycasting. According to [1], here the limitation for higher quality reconstruction is about $(3m)^3$, having a physical size of 512^3, and the requirement for graphics memory is 512MB, yet the spatial scale of KinectFusion can be extended because of the real time results.

2.4 Extending Kinect Fusion

We can use simple approaches to extend KinectFusion and that approaches are (see [1]):
Streaming: The current volume data of the graphical memory is streamed and then cleared, the volume of the camera position should be maintained. And later the different overlapping surfaces can be merged.
Motion Volume: There is a defined area where the camera can move with sensors. So this approach transforms the distance field into an active region or re-indexes the grid with a rolling buffer and reallocates the deactivated regions. A point cloud from deactivated regions is extracted by the system and creates a mesh on the host periodically. This method is dependent on a regular grid, making the active region small. The reconstruction is limited to scenes where geometric structures are closer by each other. User experience and tracking quality can also have effect on the clipped active region. Reintegration of the data into volume is not possible once it is streamed to the host. "Because of having no real time dynamic update for fusion and surface extension, these approaches are impractical". [1]

3 3D Reconstruction Pipeline

Figure 1 shows the 3D reconstruction pipeline. The main purpose here is to combine both the noisy depth maps and the memory efficient volumetric data structure to construct large and fine scale surface geometry. The surface will be encoded as SDF. The pipeline that is used here is based on the KinectFusion system (see [9]).

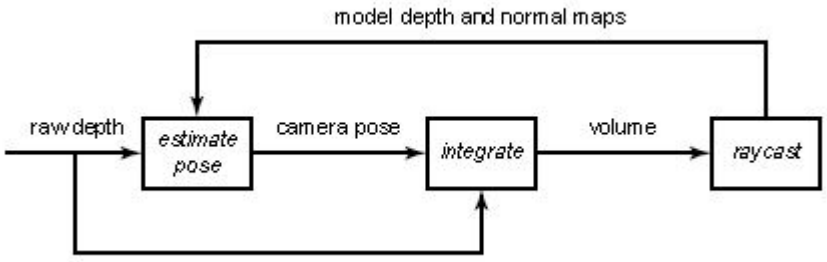

Figure 1: High level 3D reconstruction pipeline [1]

Suppose a regular dense 3D grid. We will give an input Z_i to our system.

The camera will be focused to the origin in the beginning. The volume is updated for each frame by fusing surface observations into the stored SDF and new data will be added in the empty spaces. The volume is re-casted by the use of camera pose estimation. We will use rays to find sign charges. The result will be a map. To estimate a new camera pose we will use a variant of the ICP (Iterative closest point) algorithm (see [1]).

1. Integration: The sample takes the surface as plane. Each of the voxel's center is projected to the same x, y coordinates. For each voxel the distance from center to the plane is saved.

2. Truncation and free space carving: When a sensor is moving it allows thin surface reconstruction. Then the voxels which are far away can be ignored as the SDF is useful only to those points which are near to the surface. Therefore a truncated SDF is used. This gives more information near the surface which means that the rays up to the surface are blocked or restricted. These locations are marked as free space (see [1]). Figure 2 shows the over steps of 3D reconstruction.

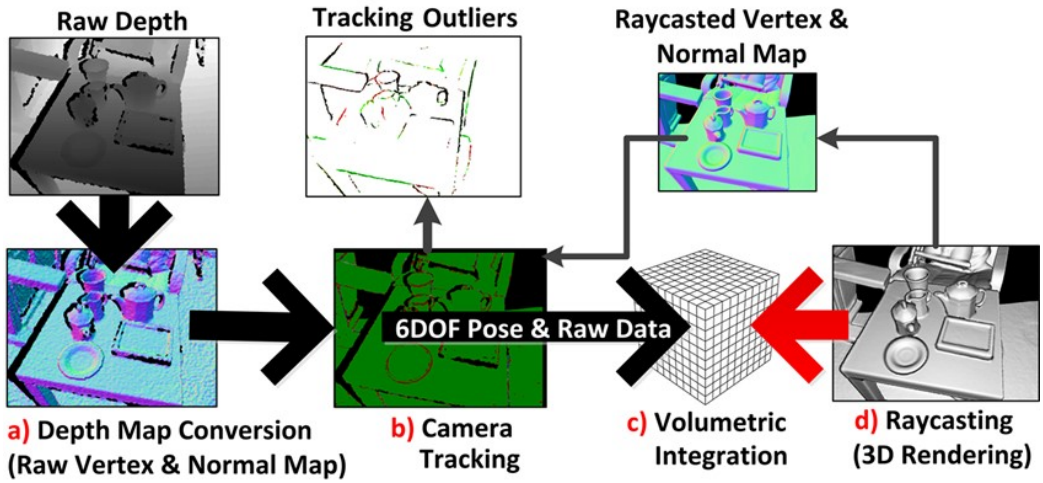

Figure 2: 3D reconstruction steps [5]

Data Structure Design: According to [1], the goal is to design a data structure that efficiently represents a TSDF (Truncated SDF) and permits well-organized integration and raycasting operations.
Figure 3 shows the view of the data structure design in the original research.

The design consists of three levels: The root is a fully allocated grid and provides a rough sub-division of the physical volume. The truncation region (in grey) needs to be refined. This process is repeated again and again until it reaches the leaf level i.e. in blue, each node is a small regular grid in this area (see [1]).

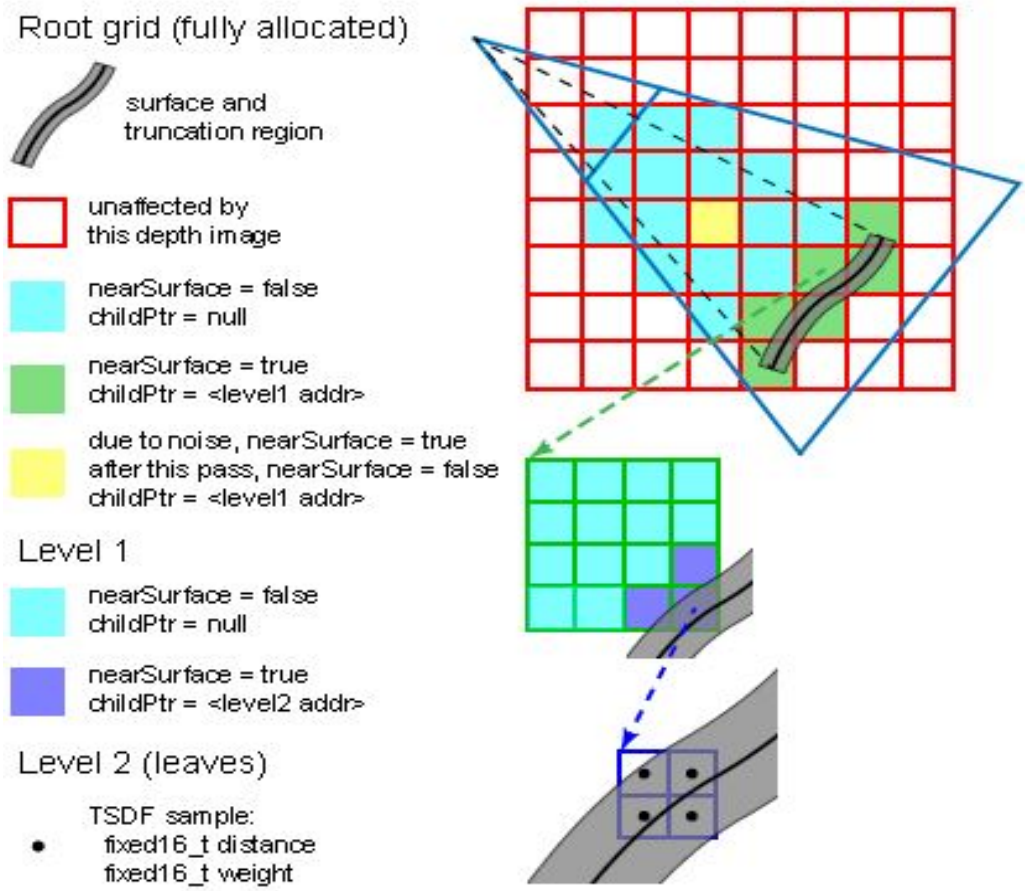

Figure 3: The data structure design [1]

4 GPU Implementation

The role of graphics memory and the integration and raycasting i.e. in parallelization is depicted in the figure 4. According to [1], the GPU implementation consists of the following processes:

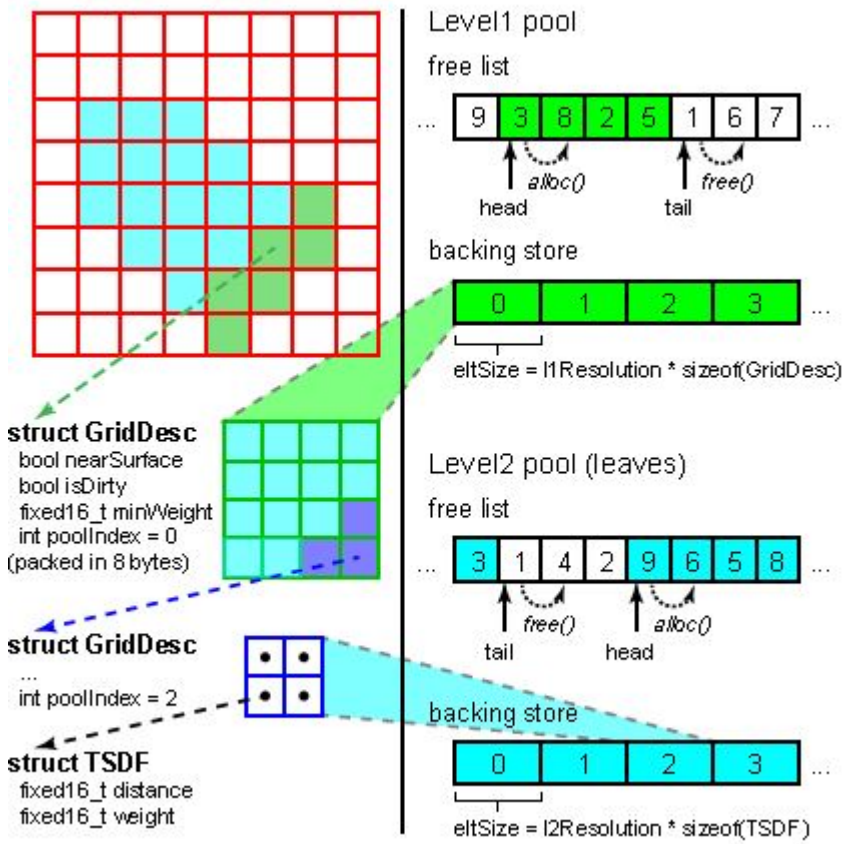

Figure 4: Memory view of hierarchy [1]

4.1 Memory Layout

The grid is stored in GPU memory. The root level grid is stored as a dense 3D array of GridDesc records and the initial values are null. For each hierarchy level a backing store, pool and free list is declared. The free list is a queue of block indices. The backing store is an array of n fixed blocks. A free block is removed from queue when a grid needs to be allocated i.e. assigned to the poolIndex and is marked as isDirty. When a garbage collection operation is performed, a block index is added to the queue.
Since the grid resolution and element size can be different in each level (flexible database [1]), a separate pool is required for each level.

4.2 Integration

For interior levels of the tree, the atomic queue has rasterized the footprints of the depth maps into finer voxel grids. The root algorithm and the interior levels' algorithm are nearly same. Now the voxels are projected to hexagons that are smaller, therefore one thread-block is applied per grid having only one thread per voxel. From the input queue the grid descriptor is read and is cleaned in parallel if 'isDirty' flag is set. A single thread is used for the Hexagon rasterization, while other threads proceeds with rest of the algorithms. In order to achieve continuous results, a very careful conservative rasterization as well as a test for intersection is very important. There is always a possibility to miss the depth samples in the interior as the voxels in that part are larger in size.

4.3 Summarization

We have to perform summarization in parallel to retain the job queues which ensures that large portions are skipped from raycasting operations. When the leaf grid is swept by parallel threads we perform a parallel reduction on any SDF value that is near to the surface geometry (see [1]) and set its grid to nearSurface. To find minimum weight in the leaf, a parallel reduction process is used. A minWeight field is used as an indication for the garbage collection.

4.4 Raycasting

When a perception of 3D object is created in 2D map it is called as Recasting. Back then when computers were slower, it wasn't possible to run real 3D engines in real-time, and raycasting was the first ever solution. It takes each of the vertical line of the screen and do calculation for single line which makes it very fast. Wolfenstein 3D is well-known game that is using this technique (see [10]). For the recasting of the data structure a DDA algorithm is used (see [11]). We have to maintain the previous and current distance, the previous SDF values and stack of voxel indices. DDA is first initialized and then for each iteration of DDA, we step to the next voxel's current level. If we are at the interior node it will be nearSurface. "We consider the gradient in SDF on Zero to compute surface normal using finite differences and trilinear interpolation. The majority of sample is in the same leaf grid as we use shallow trees with large branching factors" [1].

5 Moving volumes and streaming

Figure 5 shows the overall process of moving volume streams. According to [1], the idea is taken from the work of Crassin (see [12]) and the work of Whelan (see [13]) which is about the scale of unbounded physical dimension and decoupling from physical volume. This idea actually helps in few terms while decoupling from physical position of volume.

- The number of voxels and leaf level voxel size are always chosen while defining a hierarchy.
- The coordinate system of the world is quantitized in the unit root voxels and they are used as unique key for the hierarchy.
- The working set is defined as fixed 3D array indices in the memory of the GPU.
- A subset of the world coordinate system, i.e. the active region is centered on the camera view frustum. The active region effective resolution is enforce to be one root voxel less than that of the working set along each axis, this is done to get zero contention.

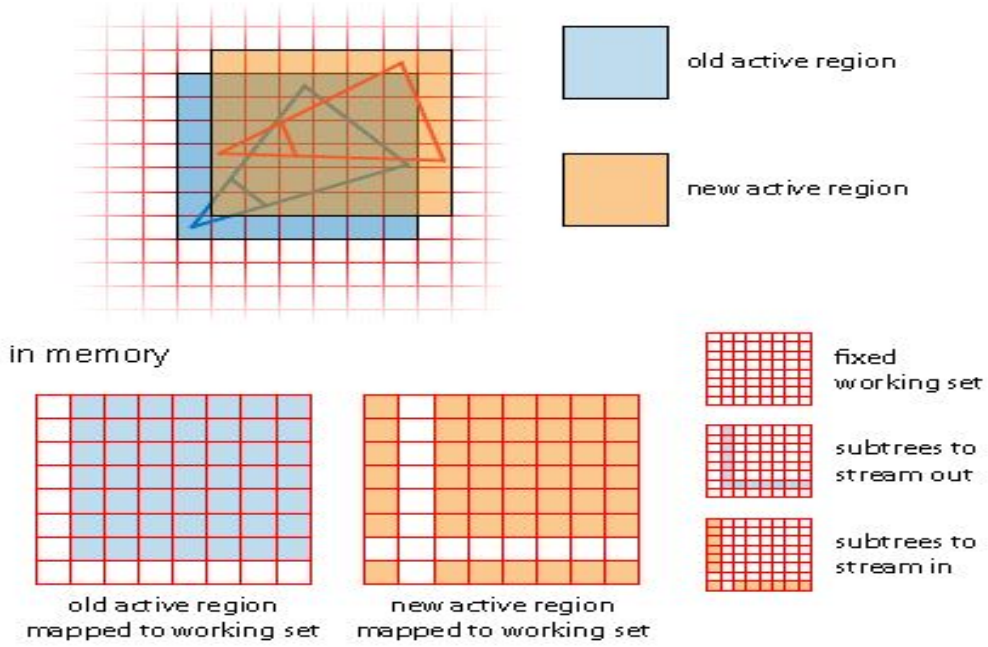

Figure 5: Illustration of bidirectional moving volume streaming.

Streaming from GPU is similar to the integration. It requires two breadth-first traversals of the hierarchy. To determine how much space is needed, the indices are copied into the GPU queue and tree traversal is performed. To calculate offset into a linear buffer a parallel prefix scan is performed and each of the subtrees is stored. A final tree traversal operation is performed to replace poolIndex with the byte offset from the beginning of each subtree (see [1]).

6 Result

A hierarchy is defined as a structure that carries out or facilitates the experimentation with the branching factors and tree depths. The optimal configuration is also shown for the existing hardware.
To capture large scale scenes a semi-mobile client server is designed. While users use the mobile-client which transmits the scene to the server where it is reconstructed, the raycasted images of the 3D model are displayed. "At the client side a laptop with a battery is used along with a Kinect camera that is controlled by the user. The server side H/W consists of a workstation with a GPU. The communication b/w client and server is carried out through a wireless 802.11n network. We use 140 Mbps bandwidth to transmit uncompressed depth frames" [1].
For the performance and memory consumption see Figure 6 is showing the combined integration and the execution time for raycasting on a 10243 grid function where the hierarchy configuration is up to level 5. The performance peaks at level 3 and decreases beyond level 4. Two factors are involved here. Integration requires synchronization through queues and offsets saving as it is performed breadth-first. "Raycasting is depth-first and requires stack space linear in the number of levels" [1].
According to [1], these performance numbers show that the algorithm is more flexible but the data structure is un-optimized.

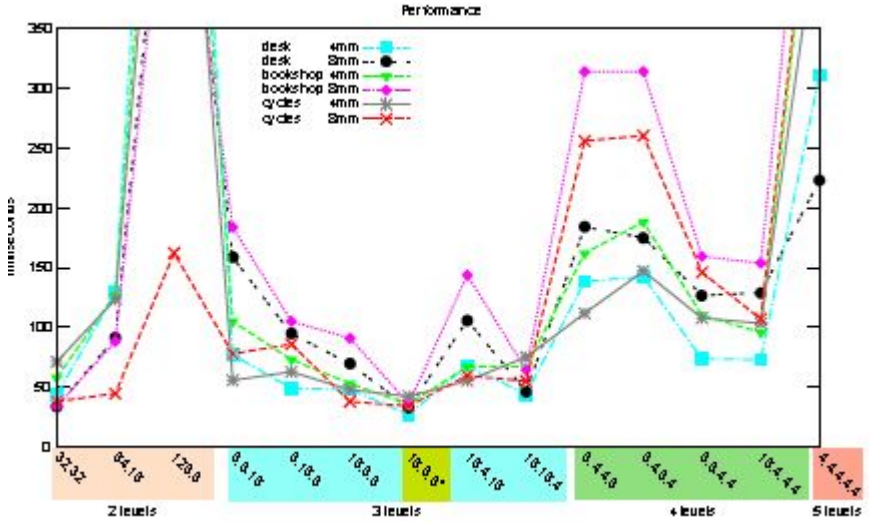

Figure 6: Average combined integration and raycasting runtime

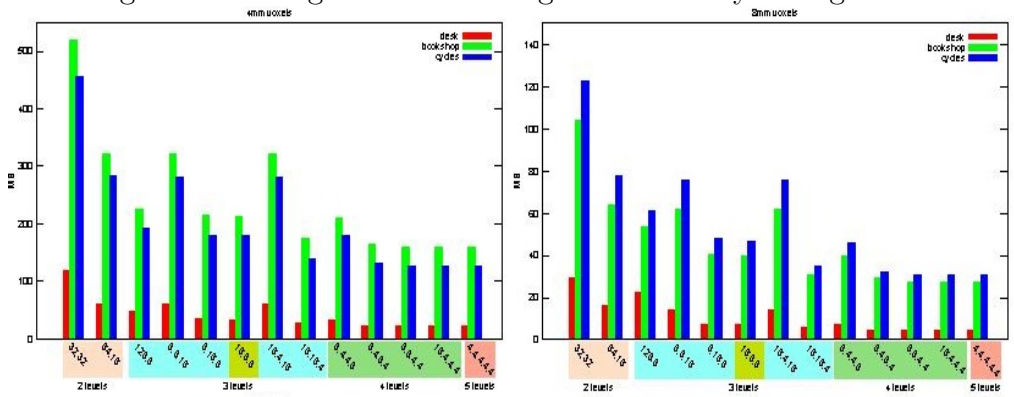

Figure 7: Memory consumption as a function of hierarchy configuration [1]

Figure 7 shows the relationship between consumption of memory and hierarchy configuration for the volume of 1024^3. The leaf grids which are smaller consume less memory because the nodes are tighter and thus they skip free spaces. After 4 levels there is diminishing returns and there is little variation between different factors of branching in the mentioned level. There is an overhead lead at level 5 and upper levels.
Interior levels are always extremely thinly allocated in all the levels. 128 MB memory is therefore allocated to the leaf level pool. But a 2048^3 volume with 4mm voxels is not plotted and it takes 245 MB memory in the worst case.

The figure above also shows a larger voxels requirement for memory (see [1]).

6.1 Limitation and future work

The main problem that is discussed and solved here is the data structure, but one of the issues is the camera drift that can be a work done in future. One issue is the re-localization in case of camera fail and the issue was somewhat solved by the use of ICP and when ICP fails the history is searched. Re-localization, loop closure and handling drift are some of the main problems in SLAM and we can say that these are actually some of the future works (see [1]).

7 Summary

The performance of the real-time volumetric reconstruction is discussed. Previously Curless and Levoy worked on this but this work is further extended to large and fine scale reconstruction. To achieve this scalability the most important of all i.e. the graphics hardware design covered here includes fast and volumetric data structure design which enables the making of good quality, scale and speed reconstruction.

References

[1] Jiawen Chen, Dennis Bautembach, and Shahram Izadi. Scalable real-time volumetric surface reconstruction. ACM Transactions on Graphics (TOG), 32(4):113, 2013.

[2] Marc Pollefeys, David Nistér, J-M Frahm, Amir Akbarzadeh, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, S-J Kim, Paul Merrell, et al. Detailed real-time urban 3d reconstruction from video. International Journal of Computer Vision, 78(2):143–167, 2008.

[3] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on, volume 1, pages 519–528. IEEE, 2006.

[4] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pages 303–312. ACM, 1996.

[5] `https://msdn.microsoft.com/en-us/library/dn188670.aspx20-06-2017`.

[6] Michael Potmesil. Generating octree models of 3d objects from their silhouettes in a sequence of images. Computer Vision, Graphics, and Image Processing, 40(1):1–29, 1987.

[7] Przemyslaw Musialski, Peter Wonka, Daniel G Aliaga, Michael Wimmer, L v Gool, and Werner Purgathofer. A survey of urban reconstruction. In Computer graphics forum, volume 32, pages 146–177. Wiley Online Library, 2013.

[8] Yan Cui, Sebastian Schuon, Derek Chan, Sebastian Thrun, and Christian Theobalt. 3d shape scanning with a time-of-flight camera. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pages 1173–1180. IEEE, 2010.

[9] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In Proceedings of the 24th annual ACM symposium on User interface software and technology, pages 559–568. ACM, 2011.

[10] `http://lodev.org/cgtutor/raycasting.html20-06-2017`.

[11] John Amanatides, Andrew Woo, et al. A fast voxel traversal algorithm for ray tracing. In Eurographics, volume 87, pages 3–10, 1987.

[12] Cyril Crassin, Fabrice Neyret, Sylvain Lefebvre, and Elmar Eisemann. Gigavoxels: Ray-guided streaming for efficient and detailed voxel rendering. In Proceedings of the 2009 symposium on Interactive 3D graphics and games, pages 15–22. ACM, 2009.

[13] T Whelan, M Kaess, MF Fallon, H Johannsson, JJ Leonard, and JBM Kintinuous. Spatially extended kinectfusion. In RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras, 2012.