Thesis for the degree of Doctor of Philosophy

Techniques for Fast and High-Quality 3D Reconstruction of General Scenes

Sverker Rasmuson

CHALMERS

Division of Computer Engineering
Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2022

Techniques for Fast and High-Quality 3D Reconstruction of General Scenes
Sverker Rasmuson

Technical Report report no 213D
ISBN 978-91-7905-629-2
Doktorsavhandlingar vid Chalmers Tekniska Högskola, Ny serie nr 5095.
ISSN 0346-718X
Department of Computer Science and Engineering
Research group: Computer Graphics

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg, Sweden
Phone: +46(0)32 772 1000

Contact information:
Sverker Rasmuson
Department of computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg, Sweden

Phone: +46(0)31 772 16 78
Email: sverkerr@chalmers.se
URL: http://www.cse.chalmers.se/~sverkerr/

Cover:
A hand pump under 3D reconstruction

Printed in Sweden
Chalmers Reproservice
Göteborg, Sweden 2022

Techniques for Fast and High-Quality 3D Reconstruction of General Scenes

Sverker Rasmuson
Department of Computer Science and Engineering
Chalmers University of Technology

Thesis for the Degree of Doctor of Philosophy

Abstract

This thesis is a collection of techniques used for 3D reconstruction: the creation of 3D models from real world objects or scenes. Given the increase in accuracy, robustness and speed of modern methods and algorithms, new and exciting applications of this technology are constantly appearing. Asset creation for games and movies is a successful example, but there are numerous other applications in architecture, medicine, communication and more.

The contribution of each paper in this thesis aims to make the use of 3D reconstruction even more ubiquitous by addressing problems such as performance, memory usage, ease-of-use, robustness and quality.

Paper I presents a compression technique for volumetric video modeled with voxels. Memory consumption is an important issue when storing volume data, especially if the data is also varying with time.

Paper II describes an end-to-end pipeline for recording and rendering volumetric video. A simple and readily available setup of webcams and a single desktop computer is used to record and render scenes in real-time.

In **Paper III**, an interactive tool is developed that aims to help in modeling of real-world objects. Structured as a simple quad modeling program, the user can construct 3D models on top of a set of photographs of a chosen object. In the background, or after explicit activation, a multi-view stereo algorithm helps the user to align the geometry correctly to images in world space. This greatly simplifies the problem of modeling real world objects accurately, while levering the input from the user to help with topology and visibility.

Paper IV implements a direct solver for the problem of neural rendering. The reconstruction is formulated as a non-linear least-squares problem which is solved efficiently with the Gauss Newton method and the Preconditioned Conjugate Gradient algorithm. This formulation achieves a significant improvement to reconstruction times compared to previous methods, while

also being suitable for distributed computing due to needing three order of magnitudes fewer iterations until convergence.

Paper V addresses the shape-radiance ambiguity in neural rendering. Given infinite spatial resolution of view-dependent information, almost any shape can satisfy the incoming radiance to each camera, resulting in errors in the geometry. To address this problem, we propose a solution to separate Lambertian and view-dependent colors during reconstruction.

Keywords: voxel, compression, free-viewpoint video, 3D reconstruction

Acknowledgements

I want to thank my supervisor Ulf Assarsson for guidance and always being available for support throughout my doctoral education. I also want to thank my co-supervisor Erik Sintorn for the countless brainstorming and discussion sessions we have had over the years. I want to especially thank you both for teaching me about how a computer actually works. Knowing the inner workings and history of computers have made the work so much more interesting and fun over the years.

Further thanks goes to present and former colleagues Roc, Dan, Viktor, Markus, Ola, Alexandra, Nadja, Dmitry, Victor, and everyone else at our division, for creating an enjoyable work environment and always providing moral support.

I owe my parents and my sisters thanks for inspiring me and always encouraging and believing in me. Finally I want to thank Jonna and our daughters Valborg and Solveig, for your unending patience and support during this long journey.

This work was supported by the Swedish Research Council under Grant 2014-4559.

List of Appended Publications

This thesis is a summary of five publications.
References to the papers will be made with roman numerals.

Paper I - Viktor Kämpe, **Sverker Rasmuson**, Markus Billeter, Erik Sintorn, Ulf Assarsson,
Exploiting Coherence in Time-Varying Voxel Data,
I3D '16: Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games,
February 2016, Pages 15–21.

Paper II - Sverker Rasmuson, Erik Sintorn, Ulf Assarsson,
A low-cost, practical acquisition and rendering pipeline for real-time free-viewpoint video communication,
The Visual Computer, volume 37, pages 553–565, 2021.

Paper III - Sverker Rasmuson, Erik Sintorn, Ulf Assarsson,
User-guided 3D reconstruction using multi-view stereo,
I3D '20: Symposium on Interactive 3D Graphics and Games, May 2020,
Article No.: 10, Pages 1–9.

Paper IV - Sverker Rasmuson, Erik Sintorn, Ulf Assarsson,
PERF: Performant, Explicit Radiance Fields,
under submission.

Paper V - Sverker Rasmuson, Erik Sintorn, Ulf Assarsson,
Addressing the Shape-Radiance Ambiguity in View-Dependent Radiance Fields,
under submission.

Table of Contents

Abstract i
Acknowledgements iv
List of Appended Publications vi

I Summary

1 Introduction **1**
1.1 Thesis Structure 1

2 Sparse Voxel DAGs **3**
2.1 Object Representation 3
2.1.1 Sparse Voxel Octrees and DAGs 4
2.2 **Paper I** 4

3 Stereo Reconstruction **7**
3.1 **Paper II** 9
3.2 **Paper III** 11

4 Volume Reconstruction **15**
4.1 Continuous Optimization and Object Representation 15
4.2 Neural Radiance Fields 16
4.3 **Paper IV** 17
4.4 **Paper V** 20

5 Discussion and Future Work **23**

Bibliography **25**

II Appended Publications **29**
Paper I - Exploiting Coherence in Time-Varying Voxel Data . . . 33

Paper II - A low-cost, practical acquisition and rendering pipeline for real-time free-viewpoint video communication 43
Paper III - User-guided 3D reconstruction using multi-view stereo 59
Paper IV - PERF: Performant, Explicit Radiance Fields 71
Paper V - Addressing the Shape-Radiance Ambiguity in View-Dependent Radiance Fields 83

TABLE OF CONTENTS

Part I

Summary

Chapter 1

Introduction

3D reconstruction is the general name for methods that aim to construct a 3D model of objects or scenes from the real world. This could be for example a person, a chair, a house, a human heart, or a whole room with its contents. The scene could be static, or contain moving objects or a moving camera. In the latter case, the reconstruction is commonly referred to as *free-viewpoint video*. In both cases, this implies a great diversity of materials and geometries that such a method must be able to handle.

The most common input data to a 3D reconstruction is a set of photographs, or videos in the case of dynamic scenes. Seldom any detailed a priori information about the scene, such as materials or lighting, is available. Thus, 3D reconstruction of a general scene is a very challenging problem.

Nevertheless, algorithms and methods for 3D reconstruction have been making great strides during recent years. Clever use of AI-methods, leveraging the use of the increasing computing power and the availability of large sets of data, have made it possible to generalize 3D reconstruction in novel ways for many challenging scenes.

This thesis aims to add to that body of work by exploring different approaches to 3D reconstruction from a set of images or video, with focus on performance, robustness and quality.

1.1 Thesis Structure

This thesis is divided into two main parts: a summary of the research with some relevant background information, and the final research papers appended at the end of the thesis. The summary is divided into a few different chapters which cover the main areas of this thesis. After a brief background to each area, the main contributions of papers associated with each chapter

is presented.

Chapter 2

Sparse Voxel DAGs

2.1 Object Representation

The most common representation of 3D models is with a mesh of triangles representing the object's surface. Triangles are the simplest surface element in three dimensions, and representing objects as surfaces instead of volumes is more efficient when it comes to memory and storage.

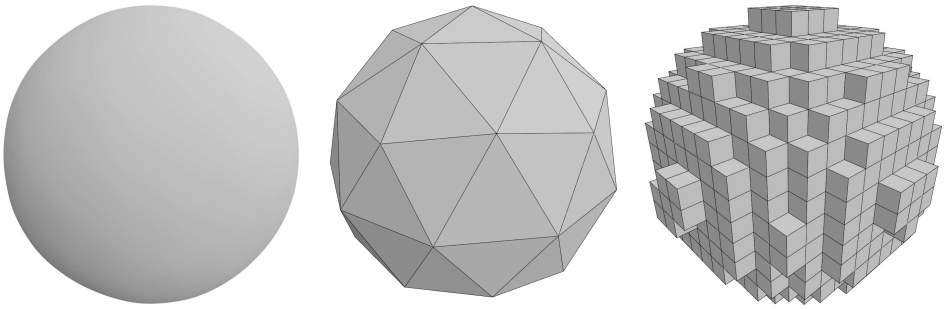

Figure 2.1: Representation of a sphere (left) with a triangle mesh (middle) and binary voxels (right).

A variant of these kinds of meshes uses quadrilaterals (quads) instead of triangles, see Figure 2.2. Quads have some desirable characteristics when it comes to modeling: they are line preserving and are better behaved upon subdivision. Quads are typically converted to triangles when rendering.

Another approach to representing objects is to use volume elements, commonly abbreviated voxels. In their simplest form, these elements can represent surfaces by just encoding existence for each voxel, i.e., one or zero.

Voxels can also represent volumes with other values, for example density, commonly used in medical applications, or for example signed distance fields.

Figure 2.2: An object represented with quads, with an increasing level of tesselation from left to right.

Since volume elements require much data at high resolutions, they are commonly paired with some kind of sparse data structure such as a Sparse Voxel Octree (SVO).

2.1.1 Sparse Voxel Octrees and DAGs

An octree is an acceleration structure commonly used in computer graphics, where a cuboid volume is iteratively subdivided into eight equally sized children, forming a tree-structure. This data structure can store sparse data by only creating children when it is required, leaving empty regions uninitialized. This data structure is known as a Sparse Voxel Octree.

An SVO can be efficiently compressed into a Directed Acyclic Graph (DAG) by identifying and merging nodes corresponding to identicals subtrees, see Figure 2.3. It is then referred to as a Sparse Voxel DAG.

2.2 Paper I - Exploiting Coherence in Time-Varying Voxel Data

Problem: 3D objects and scenes modeled with volume data often require excessive amounts memory at high resolutions. This problem is exacerbated if modeling time-varying data, e.g., data that is obtained from scanning with a depth camera. Storage is an obvious problem but also playback and streaming can be problematic due to the high bandwidth requirements.

To facilitate playback, the format also need to be uncompressed and rendered in real-time. Specifically it is necessary that playback frame by frame

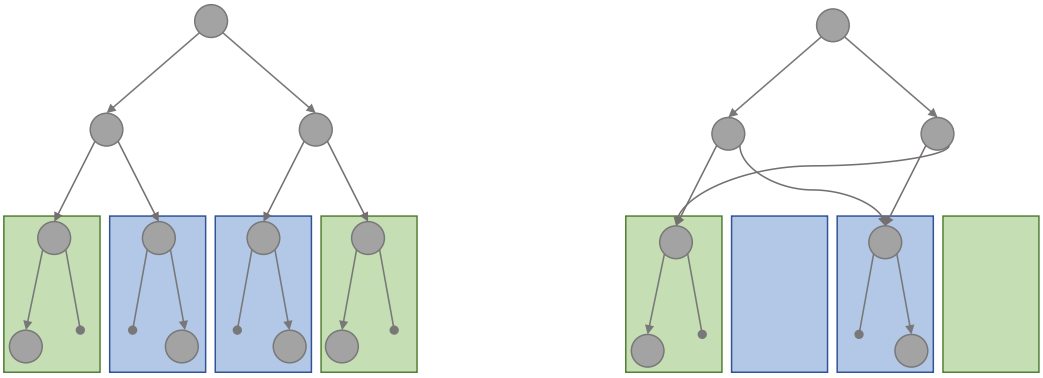

Figure 2.3: A comparison between a Sparse Voxel Octree (left) and a Sparse Voxel DAG (right) data structure. In a Sparse Voxel DAG identical subtrees are identified, and all but a unique copy of each such subtree is replaced with a pointer.

is supported.

In **Paper I**, we address this problem by applying an efficient lossless compression algorithm to time-varying voxel data, which significantly reduces the bandwidth and storage requirements, but which is still amenable to frame-by-frame playback.

Methodology: Efficient compression of voxel data has been shown to be achievable by initialising a Sparse Voxel Octree (SVO), and then applying tree compression by identifying and merging identical subtrees into a Directed Acyclic Graph (DAG) [6]. In **Paper I**, the same approach is applied to time-varying voxel data in free-viewpoint video.

Each frame of the video is assigned its own start node, while the algorithm searches for identical subtrees both in space *and* time. Firstly, all frames are concatenated in time order. Every level of this superstructure contain the nodes for all time steps. To find identical nodes, these lists are sorted level by level. The levels are then traversed in a bottom up fashion; all identical nodes are removed except one, and all child pointers in the level above are updated to point to this unique node, see Figure 2.4a.

If the data structure is used for streaming, it is important in which order unique nodes and the corresponding pointers arrive in time. For this scenario, the nodes are rearranged such that each unique node comes as early as possible in time, so that later occurrences can point back to this piece of data. This leads to the desirable property that a large chunk of the data is streamed at an early time, while later data has a higher ratio of references back to to this data, and require less bandwidth. To further reduce the band-

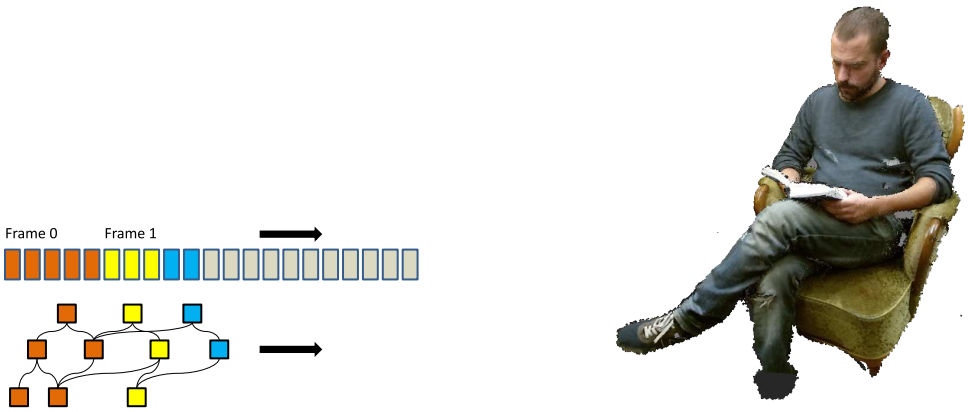

Figure 2.4: In **(a)**, an illustration of a temporal DAG is shown. It has a root node per frame, which are sorted in time. Each unique node is stored as early as possible in the stream, while subsequent occurrences just point back to this piece of data. In **(b)**, we see a frame from data captured with KinectV2 depth cameras to evaluate our algorithm in **Paper I**.

width required for streaming, implicit pointers are used when the structure of the DAG allows it.

The algorithm was tested on two artificial scenes rendered with virtual cameras, and two captures of real data (for an example see Figure 2.4b).

Contribution: **Paper I** presents an efficient lossless compression scheme for free-viewpoint video based on voxelized geometry. It achieves high compression ratios while keeping desirable properties such as the ability for frame-by-frame playback.

Intuitively, static geometry can be trivially compressed this way frame-by-frame, but the dynamic scenes show that also they exhibit a large amount of temporal coherence between frames, even if no geometry is static because e.g. the camera is moving.

Recorded real scenes do not compress quite as well due to being noisier, but still allow for significant memory and bandwidth savings compared to using an SVO per frame.

Chapter 3

Stereo Reconstruction

Given two or more images of a scene, it is possible to reconstruct the depth of each image using *triangulation*, in a similar way to how humans perceive depth using our binocular vision (see Figure 3.1a).

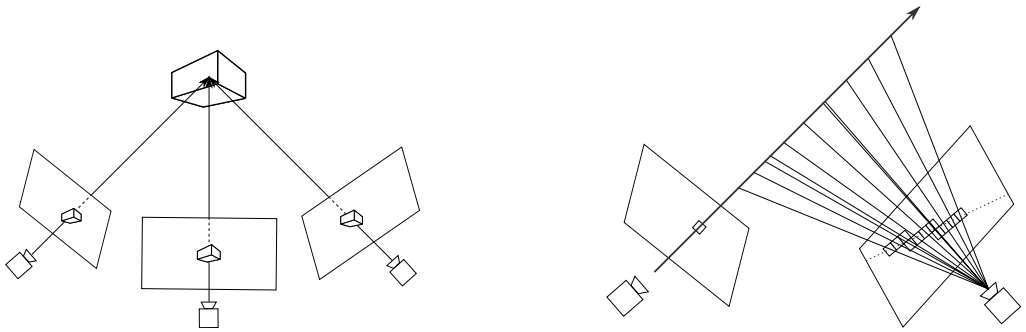

Figure 3.1: In **(a)**, an illustration of depth reconstruction via triangulation is shown. As long as the same point of an object can be correctly identified (and is visible) in each image, and that the camera intrinsics and poses are known, it is possible to infer the position of this point in 3D. In **(b)**, a ray is followed through a pixel of one image. This ray is projected on another image, and a matching pixel can be searched for along this line. Note that the angle between the two cameras determine the effective depth resolution of this search.

If the camera's intrinsic parameters are known, as well as the relative position where the images were taken, the images are said to be *calibrated*. If this is the case, triangulating 3D points from two or more cameras comes down to identifying these 3D points in each image; finding image *correspondences*.

This search for correspondences can be performed in a number of ways, the simplest of which is to, for each pixel in one of the images, search for the

matching pixel along the projected line for that corresponding ray on one of the other images (see Figure 3.1b). Comparing single pixels with each other is often prone to error, so typically a filter of some $\mathbf{N} \times \mathbf{N}$ pixels is used instead. Other image features such as gradients can also be used to find such correspondences.

The underlying assumption for this method to work is that a 3D point actually looks the same (e.g. same color or same gradient) from two different viewpoints. This is typically not true for most real-world materials, but can usually work reasonably given approximately diffuse materials and that the angle between the cameras is constrained. This, however, limits the resolution of the reconstructed depth maps (see Figure 3.1b), so typically the reconstruction is a compromise between resolution and robustness for any given scene. Another aspect to consider is that many 3D points, for example along edges, are not visible from both cameras, which obviously makes finding correspondences difficult. One solution to these problems is to add more cameras and/or more images, at the cost of performance and practicality.

3.1 Paper II - A low-cost, practical acquisition and rendering pipeline for real-time free-viewpoint video communication

Problem: Recording, streaming, and rendering free-viewpoint video is a challenging problem, which often requires large amounts of processing power, memory, and specialized equipment to achieve high quality.

Most existing setups use these kind of expensive and specialized setups to be able to control the environment and mitigate the effects of various error sources, and to achieve real-time performance.

In **Paper II** we show, by building an end-to-end pipeline from recording to rendering, that it is possible to achieve usable real-time free-viewpoint video using only commodity hardware. This is achieved by carefully analyzing the requirements and costs of each step of the pipeline, while optimizing the pipeline as a whole to fully utilize the power of graphics hardware.

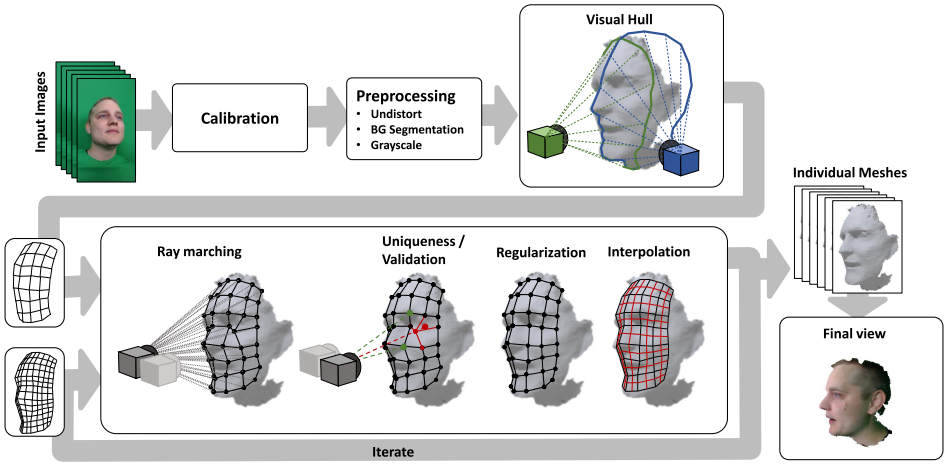

Figure 3.2: An overview of the pipeline described in **Paper II**. The pipeline includes multiple steps, starting with an input of one frame per camera, and ending with a final view rendered from a virtual camera. The whole pipeline is executed in ∼14ms.

Methodology: The pipeline is comprised of multiple steps, shown in Figure 3.2. The input cameras are calibrated, and the video streams are fed frame by frame as input to the pipeline.

The first stage of the pipeline is preprocessing of each incoming image. This preprocessing corrects for radial distortion, does a foreground/background segmentation and converts the images to grayscale.

This is followed by a visual hull algorithm, which efficiently constrains the geometry in a rasterization-based method by blending together each camera's silhouette cone [7].

At the heart of the pipeline is a stereo-reconstruction method, which produces a set of depth maps from pairwise matchings of camera frames. This algorithm is executed hierarchically from coarse to fine, by doubling the resolution in each dimension for each increasing level. Each level begins by stereo matching in the form of ray marching for each view-ray from one of the cameras in each camera pair. This is followed by a validation step, where rays along the other camera in each pair are followed to ensure correct visibility. The last steps for each level is a regularization step followed by interpolation for the next level of the hierarchy.

The output of this hierarchical scheme is a set of vertex buffers, one for each camera pair. These vertex buffers are triangulated producing one triangle mesh per buffer.

The final step of the pipeline is rendering these meshes from the current virtual camera position and blending the geometry and colors into a final view. To avoid artifacts in the seams between meshes, an algorithm weighs the meshes by distance to edge, since the geometry is typically of highest quality in the middle where the angle of incidence is close to perpendicular. For a sample of results of novel views rendered with our method, see Figure 3.3.

Contribution: With **Paper II**, we show that it is possible to achieve real-time free-viewpoint video using a modest setup of only webcams and a single desktop computer with a moderately powerful graphics card (Nvidia GTX 980). Given careful analysis of the different steps in the pipeline and their performance impact, usable quality is achievable without any specialized equipment or setups. We compare the output of our method with a KinectV2 depth camera with reprojected colors (see Figure 3.3), and with an offline face reconstruction method [1]. We outperform the first method in terms of quality, and come close to the second method, while reconstructing a frame in approximately 20ms instead of 20 minutes.

Figure 3.3: Novel views rendered with our method in **Paper II**. To the right is a comparison between our method (top) and using a depth-camera with re-projected colors (bottom).

3.2 Paper III - User-Guided 3D Reconstruction Using Multi-View Stereo

Problem: Fully automatic 3D reconstruction, such as photogrammetry pipelines, are increasingly being used to create assets for applications such as movies and video games. This is a compelling approach since large parts of the expense of modern media productions are due to asset creation. In practice, however, a non-trivial amount of work is still necessary for such automatically created assets in post-processing, where clean up and re-topologization is performed. We therefore propose a semi-interactive method where the user input is combined with an automatic Multi-View Stereo routine. This amounts to a similar amount of work as using a fully automatic method including post-processing, but with increased control and flexibility for the creator.

Methodology: We have created a simple tool for point and click quad modeling on top of images of an object of interest. The user maps out a coarse topology on top of the images, choosing views that have clear visibility towards the currently modeled surface region. Two or more views needs to be selected per region for epipolar geometry to be established. The user selects one of these views as the *reference view* where modeling will take place (see Figure 3.4 for an example).

At any time during this process, the user can choose to invoke an auto-

Figure 3.4: An example workflow used with our tool in **Paper III**. Left and middle: the user starts modeling by choosing two views and placing a quad, followed by more quads in the desired topology. To the right, more views have been used to ensure correct visibility for each part of the surface, color coded for easy identification.

matic optimization routine, which uses a Multi-View Stereo Algorithm to try to accurately map the geometry to the camera images in world space. The Multi-View Stereo algorithm uses the views that the user has selected to ensure correct visibility. The user may also choose to subdivide the geometry with Catmull-Rom subdivision when a coarse topology has been mapped out [2].

We compare the results of our method with the automatic photogrammetry pipeline Meshroom [4] (see Figure 3.5 for an example).

Contribution: We present an interactive method for quad modeling of real-world objects. This approach is compelling since it divides the problem in two parts, where the advantages of both user input and computer processing power is fully utilized. Topology and visibility is handled by the user, while aligning quads correctly in world space with respect to the images is handled by the computer.

This amounts to some extra work for the user compared to a fully automatic workflow. However, including the often necessary post-processing needed for usable assets, our method requires a similar amount of work and gives more power and flexibility to the user in the reconstruction process.

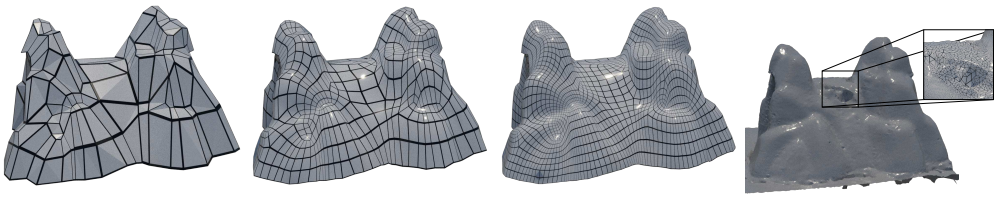

Figure 3.5: A comparison of our method in **Paper III** at different levels of subdivision (three left-most figures), compared to the photogrammetry pipeline Meshroom (right figure). The left-most figure is the topology created by the user, and the following two are after subdivison and optimization. Except for achieving a much nicer topology, our method is also capable to circumvent the problems in the texture-less region in the middle of the object, with help from user input.

Chapter 4

Volume Reconstruction

4.1 Continuous Optimization and Object Representation

Given the success of neural networks and AI-techniques in solving otherwise intractable problems, an increasingly important attribute of object and scene representation is *differentiability*. The basis for all neural networks is the optimization of parameters in the network by computing the error for each input and propagating it backwards through the layers. It is very important that it is possible to compute well-behaved derivatives with respect to this error for the method to work properly.

Representing scenes and geometry in a way that is fully differentiable can be tricky, since discontinuities in geometry (e.g. sharp edges), visibility (e.g. occlusion) and lighting (e.g. hard shadows) are common in practically every scene. Surface representations like triangles are often problematic since they can fall into degenerate states, e.g., that two or more vertices share the same value, or that they become inverted (inside out). Geometry represented by triangles is often not continuous to the second degree (with respect to triangle normals).

In **Paper III**, quads are used, which due to their similarity to triangles can be problematic to use in a minimization strategy. Care had to be taken to ensure that the error function was well behaved and reasonably continuous. An important part of this approach was to constrain the vertices to only allow for movement according to view rays in the selected reference view. Additional regularizing terms where also important to keep the surfaces continuous. Input from the user also helped to mitigate these problems by ensuring reasonable topology and visibility.

In **Paper IV** and **Paper V**, we also formulate the reconstruction as

an optimization problem. Here, we use a representation based on volume rendering of a semi-transparent volume. This representation is fully differentiable and make computation of analytical derivatives plausible. This volume representation behaves much better compared to triangles or quads for similar problems and has therefore become very popular for this kind of 3D-reconstruction and novel view synthesis.

4.2 Neural Radiance Fields

NERF [9] proposed a method of 3D reconstruction and novel view synthesis called Neural Radiance Fields. The method is based on differentiable volume rendering, first explored in Neural Volumes [8].

The basic idea is to use volume rendering, formulated as

$$\mathbf{H}(\mathbf{r}(t)) = \sum_{t_n}^{t_f} T(t)(1 - e^{(-\sigma(t)\delta(t))})\mathbf{c}(t) \tag{4.1}$$

where

$$T(t) = e^{(-\sum_{t_n}^{t} \sigma(t)\delta(t))} \tag{4.2}$$

for a ray $\mathbf{r}(t)$ going through a volume, with density σ, the distance between adjacent samples δ and the color $\mathbf{c}$. The volume is represented using these density and colors values, in same manner as, e.g., a CT scan. To render an image, Equation 4.1 is invoked for each pixel of a virtual camera.

Given a set of input images, a reconstruction problem can be formulated as minimizing the error between rays sent through the volume for each virtual pixel of each image and the reference color in that pixel. Each input position along the ray and the ray direction is fed into a Multi-Layer Perceptron (MLP). NERF employs positional encoding to the input, which enables reconstruction of finer details in the scene [12].

NERF performs remarkably well for novel view synthesis and 3D reconstruction, given the shape-radiance ambiguity of view-dependent colors. Given a high angular resolution of view-dependent colors, almost any geometry can satisfy the incoming radiance to each camera. In Nerf++ [14], the authors show that it is the MLP structure of NERF that help to mitigate this problem, where the view direction is inserted late in the network. This regularizes the color to vary smoothly with the view direction. The network also has limited parameters in describing the view-dependent color, which avoids over-fitting.

4.3 Paper IV - PERF: Performant, Explicit Radiance Fields

Problem: 3D reconstruction using neural rendering has shown unparalleled results in novel view synthesis, approaching photorealistic quality in a wide variety of complex scenes [9]. There are two major downsides with this otherwise excellent approach: rendering time and reconstruction time. Rendering a frame typically takes minutes while reconstruction of a whole scene can take days. The first problem has been addressed by a number of follow-up papers, but the second is still relatively unexplored. In **Paper IV**, we try to speed up the reconstruction time by using the same problem formulation as NERF, but instead of solving it with neural networks we opt for a direct approach storing the data explicitly in voxels. Even though it is not the primary aim of **Paper IV**, we also achieve real-time rendering by using this simpler scene model.

Methodology: The volume is represented using voxels that store color $\mathbf{c} = (c_r, c_g, c_b)$ and density σ in each cell. Such a volume can be rendered for each pixel in a virtual camera by integrating a ray going through the corresponding pixel i and through the volume such that

$$\mathbf{H}_i = \sum_t T(t)(1 - e^{(-\sigma(t)\delta(t))})\mathbf{c}(t) \tag{4.3}$$

for a step t, where

$$T(t) = e^{(-\sum_{tn}^{t} \sigma(t)\delta(t))} \tag{4.4}$$

is the throughput of the ray so far in the volume. $\delta(t)$ is the distance between the current and previous sample.

The reconstruction problem can be formulated as the problem of minimizing the difference of each such accumulated color from Equation 4.3, and the reference color $\mathbf{r}_i$ of the corresponding pixel. For each pixel i in the collection of images and each color channel k, this can be formulated as minimizing the error function F such that

$$\min_{\mathbf{c}_j;\sigma_j} F, \tag{4.5}$$

$$F = 0.5 \sum_i \sum_{k=1}^{3} (H_{i,k} - r_{i,k})^2 \tag{4.6}$$

for all colors $\mathbf{c}_j$ and densities σ_j in voxels j.

Equation 4.6 can be interpreted as a non-linear least-squares system, of which the general form is

$$S(\mathbf{x}) = 0.5 \sum_{i=1}^{m} r_i(\mathbf{x})^2, \tag{4.7}$$

where r_i are called *residuals*.

These type of equations can be solved efficiently with the Gauss-Newton method. This method is also amenable to large problems of this size given that an efficient method is used to compute the new step for each iteration. In **Paper IV**, we use the Preconditioned Conjugate Gradient Method for this purpose [10].

Solving Equation 4.6 in this manner requires the computation of partial derivatives with respect to all colors $\mathbf{c}_j$ and densities σ_j. Since we implement this solver ourselves in CUDA, we can not rely on automatic differentiation often used in AI methods. We show that we can compute analytical derivatives efficiently with a simple algorithm that computes each partial derivative traversing each ray backwards from the end-point.

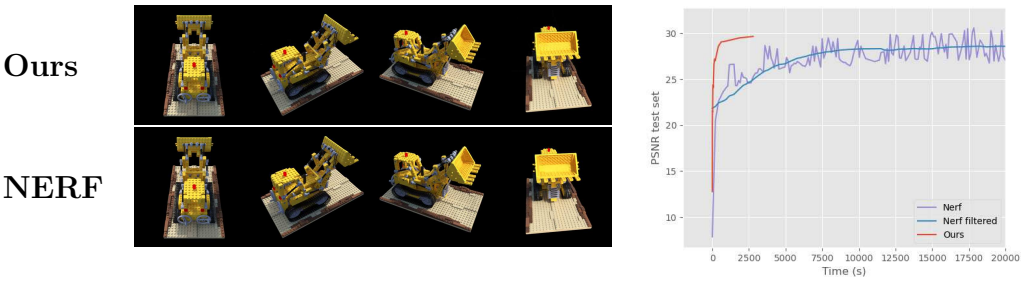

Figure 4.1: Results from our method in **Paper IV** compared to NERF without view-dependence. Our method produces novel views of similar quality as NERF, but in a fraction of the time as can be seen in the graph to the right.

Contribution: We show that it is possible to directly solve Equation 4.6 instead of going through a neural network. The problem is identified to be of non-linear least-squares type, which enables the usage of an efficient Gauss-Newton solver. The solver is implemented in CUDA, using a simple algorithm for computation of analytical partial derivatives.

With this approach we achieve a significant reduction of the required reconstruction time; from about a day to approximately half an hour for a

typical scene compared to standard NERF. Another benefit of our method is that it requires about three order of magnitudes fewer iterations until convergence. This could be of great significance in the use of distributed computing to further speed up the reconstruction.

4.4 Paper V - Addressing the Shape-Radiance Ambiguity in View-Dependent Radiance Fields

Problem: A problem when trying to reconstruct a 3D scene with geometry and view-dependent colors is the so called shape-radiance ambiguity. Given infinite angular resolution of outgoing light from each point of an object, almost any geometry can satisfy the incoming radiance to each camera. This can lead to artifacts in the result where the reconstruction has been able to substitute geometry with changes in color depending on view.

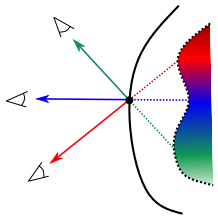

Figure 4.2: Illustration of the shape-radiance ambiguity. A single point on a false surface (solid) can satisfy the incoming radiance to the cameras through view-dependent colors, even though these rays correspond to three distinct points on the real surface (dotted).

We therefore propose to not jointly optimize both geometry and view-dependent colors, but instead try to separate Lambertian and view-dependent information while reconstructing. This view-dependent information can then be reprojected onto the resulting geometry or the result can be used as input for other methods handling view-dependent information. These methods should then have a higher probability of succeeding since the problem of shape-radiance ambiguity has been mitigated.

Methodology: In **Paper V**, we model the scene with voxels that we assume to be fully Lambertian, along with densities modeling geometry. We extend the volume reconstruction by adding a difference plane of matching resolution in front of each camera used for training. This plane stores one float value per pixel, which indicates how much this given pixel deviate from the Lambertian color in the volume.

This value, α_{s_i}, represents how to weigh in a view-dependent part of the camera image according to:

$$\hat{\mathbf{H}}_i = (1.0 - e^{(-\alpha_{s_i}\sigma_s)})(\mathbf{r}_i - \mathbf{H}_{d_i}) + \mathbf{H}_{d_i} \tag{4.8}$$

where r_i is the reference color in the corresponding pixel i, and

$$\mathbf{H}_{d_i} = \sum_j e^{-V_j}(1.0 - e^{(-\sigma_j \delta_j)})\mathbf{c}_j,$$
$$V_j = \sum_{k=0}^{j-1} \sigma_k \delta_k$$

is the Lambertian part of the color accumulated from the ray traversing the volume through voxels j according to equation 4.1

The modification in Equation 4.8 is added to the non-linear least-squares framework used in PERF [11].

We also add an additional computing pass to account for view-dependent lighting that does not fit the model of a separable Lambertian and view-dependent component well, e.g., mirror-like reflections. For every voxel, we train a low-resolution view-dependent function and compare the value of this function with the value from the reference camera including visibility. This error is added as weight w to the Cauchy loss following Plenoxels [13] and SnerG [5]:

$$\mathcal{L}_i = log(1 + w\sigma_i^2) \tag{4.9}$$

with density σ for each voxel i. This has the effect of lowering the density of each voxel relative to the fit of the view-dependent function.

Our additions can to a large degree separate Lambertian and view-dependent information as can be seen in Figure 4.3.

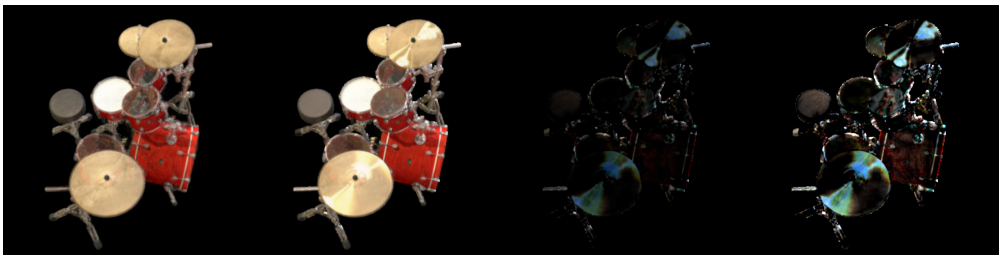

Figure 4.3: Results from our algorithm in **Paper V**. From left to right: Lambertian color, view-dependent + Lambertian color, view-dependent color only, view-dependent color brightened for easier inspection.

Contribution: We present a novel way of mitigating the shape-radiance ambiguity in volume reconstruction. Our method adds almost no performance penalty or memory overhead to the original framework in PERF, but helps with convergence and mitigation of artifacts in scenes with highly view-dependent components. After reconstruction of the geometry, a high-resolution view-dependent function could be used with less problems with the shape-radiance ambiguity.

Chapter 5

Discussion and Future Work

Recent development has been making great strides, especially when it comes to novel view synthesis. However, 3D reconstruction, which is more specifically addressed in **Paper III** and **Paper V**, still struggle with unsolved problems to become practical. We are still not at the point where we can just take a bunch of images of an object and get something out that we can immediately put into a game or movie.

Translating these newer methods to real time is also challenging, which is not surprising since the ratio between a day's reconstruction in an offline method such as NERF and the 33ms (assuming 30 frames per second) required for real time is about 2.6 million.

This thesis is in a way a testament to the rapid development within this field of research, and especially how tightly it correlates with available computing power. When I started this journey in 2015, the latest NVIDIA graphics cards where the 900-series, with e.g. a GTX 980 having 4GB of memory and 5 teraFLOPS of processing power. A recent graphics card such as RTX 3080, which is more than a year old now, has 10GB of memory and a processing power of almost 30 teraFLOPS. That is 2.5x as much memory and 6x the processing power, in only five years. And this is not counting the incredible performance gains from more specific hardware such as tensor cores.

This is also reflected in how the research field has changed in the same time period. Many modern techniques such as NERF are comparatively simple and high-level compared to, e.g., the complex pipeline of Microsoft's High-Quality Streamable Free-Viewpoint Video from 2015 [3][9]. In truth, the whole AI-boom and resurgence of neural networks would probably not have happened if we would not have had the hardware to support it.

I think that an important lesson to learn is to not be to near-sighted about which algorithms are most efficient on current hardware, but always have in

mind that changes in available computing capabilities inevitably changes which approaches that are most successful at any given time.

In this thesis a wide variety of scenarios for 3D reconstruction are explored, ranging from faces to toys to outdoor scenes. Both static scenes of individual objects and dynamic scenes of moving persons are considered. A challenging aspect of 3D reconstruction is the wide variety of types of objects and materials that can occur in a given scene. The work in this thesis show that many types of general scenes can be handled with modern methods and techniques, while still offering good performance and high quality of reconstruction.

My research group's background in Computer Graphics has given us a different perspective on much of the prior research, of which much has been closer to Computer Vision. While digesting a different research field has sometimes been challenging, we have also been able to approach problems in novel ways, always having performance in mind, and not being afraid of implementing our own solutions to problems from scratch.

There are many interesting avenues for further research following the papers in this thesis. The user interface in **Paper III** would be very interesting to put to the test, by implementing a version in production software such as Blender. It would also be interesting to see if it would be possible to implement a similar user interface, but instead use a volume reconstruction algorithm such as in, e.g., **Paper IV**.

The system of volume reconstruction used in **Paper IV** and **Paper V**, which is based on a non-linear least squares solver, has many interesting properties that demand further exploration. A few examples are: finding a more efficient preconditioner, implementing a sparse representation of scenes, and fully integrating view-dependent colors with help from the work in **Paper V**. An interesting property of the system in **Paper IV** is that it requires very few iterations to reach convergence compared to other methods. This makes it very amenable for distributed computing, which would be interesting to explore further.

Bibliography

[1] Thabo Beeler, Bernd Bickel, Paul Beardsley, Bob Sumner, and Markus Gross. High-quality single-shot capture of facial geometry. *ACM Trans. Graph.*, 29(4):40:1–40:9, July 2010.

[2] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350 – 355, 1978.

[3] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. High-quality streamable free-viewpoint video. *ACM Trans. Graph.*, 34(4):69:1–69:13, July 2015.

[4] Carsten Griwodz, Simone Gasparini, Lilian Calvet, Pierre Gurdjos, Fabien Castan, Benoit Maujean, Gregoire De Lillo, and Yann Lanthony. Alicevision Meshroom: An open-source 3D reconstruction pipeline. In *Proceedings of the 12th ACM Multimedia Systems Conference - MMSys '21.* ACM Press, 2021.

[5] Peter Hedman, Pratul P. Srinivasan, Ben Mildenhall, Jonathan T. Barron, and Paul Debevec. Baking neural radiance fields for real-time view synthesis. *ICCV*, 2021.

[6] Viktor Kämpe, Erik Sintorn, and Ulf Assarsson. High resolution sparse voxel dags. *ACM Trans. Graph.*, 32(4):101:1–101:13, July 2013.

[7] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(2):150–162, February 1994.

[8] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 38(4), July 2019.

[9] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[10] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

[11] Sverker Rasmuson, Erik Sintorn, and Ulf Assarsson. Perf: Performant, explicit radiance fields, 2021.

[12] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Adv. Neural Inf. Process. Syst.*, 2020-December:1–24, 2020.

[13] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks, 2021.

[14] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields, 2020.