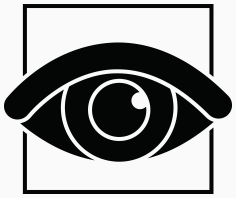


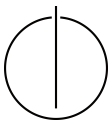
Dissertation

High-Quality 3D Reconstruction from Low-Cost RGB-D Sensors

Robert Maier

Chair for
Computer Vision and
Artificial Intelligence

Department of
Informatics



Technische
Universität
München

Technische Universität München

Fakultät für Informatik
Lehrstuhl für Bildverarbeitung und Künstliche Intelligenz

High-Quality 3D Reconstruction from Low-Cost RGB-D Sensors

Robert Maier

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften
(Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Prof. Dr. Matthias Nießner
Prüfer der Dissertation:	1. Prof. Dr. Daniel Cremers
	2. Prof. Dr. Lourdes Agapito University College London, UK

Die Dissertation wurde am 17. September 2019 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 2. März 2020 angenommen.

To my family.
– For all the love and support.

To my beloved brother.
– I miss you dearly.

Abstract

This thesis explores the reconstruction of high-quality 3D models of real-world scenes from low-cost commodity RGB-D sensors such as the Microsoft Kinect. State-of-the-art methods accurately estimate the camera motion and subsequently fuse the obtained depth maps in real-time into volumetric Signed Distance Fields (SDF) in order to handle the strong noise characteristics of the depth measurements.

While the generated dense 3D models are geometrically accurate, the ℓ_2-regularization property involved in the depth and color fusion results in oversmooth 3D surface geometry and blurry per-voxel surface colors. Firstly, we introduce an effective practical method for reconstructing sharp model textures by deblurring and fusing low-resolution RGB-D input frames in super-resolution keyframes and subsequently texture mapping them onto the reconstructed 3D meshes. Secondly, we propose a surface reconstruction approach that jointly optimizes for surface geometry encoded in a SDF, surface material properties and textures from keyframes along with their camera poses. The employed Shape-from-Shading technique utilizes spatially-varying lighting estimation to recover fine-scale geometric details; joint optimization of both geometry and image formation model leads to sharp and consistent surface textures as a by-product.

State-of-the-art 3D reconstruction frameworks estimate globally consistent camera motion in real-time even for larger scenes by detecting loop closures and continuously performing a global pose graph optimization. However, dense 3D model fusion is mostly designed as a post-process since it is computationally demanding and the final camera poses are required in advance. We present an efficient surface correction method for updating 3D reconstructions of large-scale scenes on-the-fly on pose graph updates to enable up-to-date model previews during the reconstruction process. Consecutive RGB-D frames are fused locally into keyframes, which are incorporated directly into a sparse SDF volume on the GPU. As the RGB-D SLAM system detects loop closures and globally optimizes the camera poses, the SDF volume is corrected online using an intelligent keyframe re-integration strategy with reduced GPU-host streaming.

Extensive quantitative and qualitative evaluations show the effectiveness and practicality of the proposed methods for high-quality geometry and appearance reconstruction and efficient on-the-fly surface correction.

Zusammenfassung

Diese Arbeit befasst sich mit der Rekonstruktion hochauflösender 3D-Modelle realer Szenen mit kostengünstigen RGB-D Sensoren wie der Microsoft Kinect. Moderne Verfahren schätzen die Kamerabewegung und fusionieren die ausgelesenen Tiefenkarten in Echtzeit in volumetrische vorzeichenbehaftete Distanzfunkionen (SDF) zum Reduzieren des charakteristischen Rauschens der Tiefenmessungen.

Während die generierten 3D-Modelle geometrisch genau sind, führt die ℓ_2-Regularisierung der Tiefen- und Farbfusion zu überglätteter Oberflächengeometrie und verschwommenen Voxel-Farben. Zuerst führen wir eine praktische Methode zur Rekonstruktion scharfer Modelltexturen ein. Hierbei werden RGB-D Eingabe-Frames niedriger Auflösung geschärft und in hochauflösende Keyframes fusioniert, die anschließend mit Texture Mapping auf die rekonstruierten Dreiecksnetze abgebildet werden. Des Weiteren stellen wir einen neuen Ansatz zur Oberflächenrekonstruktion vor, der gleichzeitig die in einer SDF gespeicherte Oberflächengeometrie, Materialeigenschaften sowie Texturen von Keyframes und deren Kameraposen optimiert. Die verwendeten Shape-from-Shading-Techniken verwenden eine räumlich variierende Beleuchtungsschätzung zum Wiedergewinnen feiner geometrischer Details; die gemeinsame Optimierung von Geometrie und Abbildungsmodell führt zu scharfen und konsistenten Texturen als Beiprodukt.

Moderne 3D-Rekonstruktionsverfahren sind in der Lage, global konsistente Kamerabewegungen sogar für großflächige Szenen in Echtzeit zu schätzen, indem sie Schleifenschlüsse erkennen und eine kontinuierliche globale Optimierung des Posengraphes durchführen. Da die Fusion eines dichten 3D-Modells jedoch mit einem hohen Berechnungsaufwand einhergeht und die finalen Kameraposen im Vorhinein bekannt sein müssen, ist diese hauptsächlich als Post-Prozess konzipiert. Zum Bereitstellen einer aktuellen Voransicht des Modells während des Rekonstruktionsvorgangs präsentieren wir eine effiziente Oberflächenkorrektur, mit der 3D-Rekonstruktionen von großen Szenen bei Änderungen des Posengraphs zur Laufzeit aktualisiert werden. Hierbei werden aufeinanderfolgende RGB-D Frames lokal in Keyframes fusioniert und anschließend direkt in ein speichereffizientes SDF-Volumen auf der GPU integriert. Nach dem Erkennen von Schleifenschlüssen und der globalen Optimierung der Kameraposen wird das SDF-Volumen zur Laufzeit korrigiert unter Verwendung einer intelligenten Strategie zum Re-Integrieren von Keyframes durch reduzierte Kommunikation zwischen GPU und Host.

Umfassende Experimente demonstrieren die Effektivität der vorgestellten Methoden für hochauflösende 3D-Rekonstruktion und effiziente Oberflächenkorrektur.

Acknowledgements

This dissertation would not have been possible without the continuous support and encouragement of many amazing people.

Firstly, I would like to thank my doctoral advisor Prof. Daniel Cremers for his excellent supervision and guidance throughout the course of this PhD. I am very grateful for the exciting opportunity to pursue a PhD on a research topic of my choice, and for believing in my skills and capabilities. I feel lucky for becoming a part of such an outstanding research group with so many talented and awesome people to learn from.

Secondly, I want to thank Prof. Matthias Nießner and Prof. Lourdes Agapito for agreeing to serve as members of the examination committee.

I would like to express my gratitude to Sabine Wagner for all the invaluable assistance with administrative matters, and to Quirin Lohr for his continuous technical support that always solved seemingly irreparable things in the most efficient way possible.

The papers published throughout this degree would not have been possible without the contributions of all my excellent collaborators: Erik Bylow, Daniel Cremers, Maksym Dzitsiuk, Vladislav Golyanik, Fredrik Kahl, Jan Kautz, Kihwan Kim, Lingni Ma, Matthias Nießner, Carl Olsson, Raphael Schaller, Didier Stricker, Jörg Stückler and Jürgen Sturm. I am truly grateful for their very hard work (even throughout quite a few all-nighters), their enormous experience and all the fruitful discussions.

I also want to thank all my former office mates* and colleagues at the Computer Vision Group at TUM for the amazing time, all the memorable moments and the nice discussions about research and completely unrelated topics: Mohammed Brahimi, Erik Bylow*, John Chiotellis, Niko Demmel, Patrick Dendorfer, Julia Diebold, Csaba Domokos, Marvin Eisenberger, Jakob Engel, Virginia Estellers Casas, Thomas Frerix, Vladimir Golkov, Björn Häfner, Philip Häusser, Caner Hazirbas, Mariano Jaimez Tarifa, Christian Kerl*, Maria Klodt, Georg Kuschk*, Zorah Lähner, Emanuel Laude, Laura Leal-Taixe, Benedikt Löwenhauser, Lingni Ma*, Tim Meinhardt, Thomas Möllenhoff, Michael Möller, Martin Oswald, Yvain Queau, Emanuele Rodola, Frank Schmidt, David Schubert, Raluca Scona*, Yuesong Shen, Christiane Sommer, Mohamed Souiai*, Frank Steinbrücker, Evgeny Strekalovskiy, Jan Stühmer, Lukas von Stumberg, Jürgen Sturm, Jörg Stückler, Rudolph Triebel, Vladyslav Usenko, Matthias Vestner*, Rui Wang, Patrick Wenzel, Thomas Windheuser, Tao Wu, Nan Yang, Zhenzhang Ye and Qunjie Zhou. Please forgive me the simple alphabetical enumeration without further details or prioritization - but trust me that I will surely never forget the great moments we had; may it be conference trips, group retreats, Weißwurst breakfasts, coffee breaks, afternoon walks, table soccer, Karaoke nights or Oktoberfest visits. Moreover, I am grateful to Erik

Bylow for proof-reading parts of this thesis.

I am more than happy that I got the chance to do fascinating internships at two world-class research labs during my PhD. Firstly, I want to thank Kihwan Kim and Jan Kautz for the internship at NVIDIA in Santa Clara, CA (USA), and the invaluable experience of working in a cutting-edge research environment in the Silicon Valley. Secondly, I would like to thank Richard Newcombe, Tom Whelan and Steven Lovegrove for the internship at Facebook Reality Labs (Oculus Research) in Redmond, WA (USA), and the opportunity to work in a team with world-renowned Computer Vision experts on the AR/VR technology of the future. And of course, the time in both labs would not have been the same without some true companions, Vlad Golyanik at NVIDIA and Jesus Briales at FRL.

I am very thankful to all my great friends for always being by my side and enriching my life with joy and fun: Georg and my close friends from Lappen Holledau; my dear cousin Vera; my friends from soccer, school, karaoke, etc. etc.; it is hard to explicitly name all of you here, but I dearly appreciate having all of you in my life.

Lastly, and most importantly, I want to especially thank my family and my dear Vero from the bottom of my heart, for all their love, patience and unconditional support. My beloved parents have always had my back, strongly supported me in whichever goal I had and encouraged me to follow my dreams. I could always count on them in good times and in not so easy times. My sister Gisela and her family, in particular her kids Johannes and Katharina, always brighten up my day whenever I see them. I am thankful to my dearly missed brother Peter, who has always been a role model; his example has always motivated me to give everything and to try to make a difference. Vero has always supported me with her unconditional love and patience during the lengthy process of this dissertation in every way possible.

While the journey of this PhD has been both challenging and exciting, paved with invaluable experiences and many lessons learned, I am equally excited about a promising future.

— Robert

Contents

Part I

Introduction and Preliminaries

Chapter 1

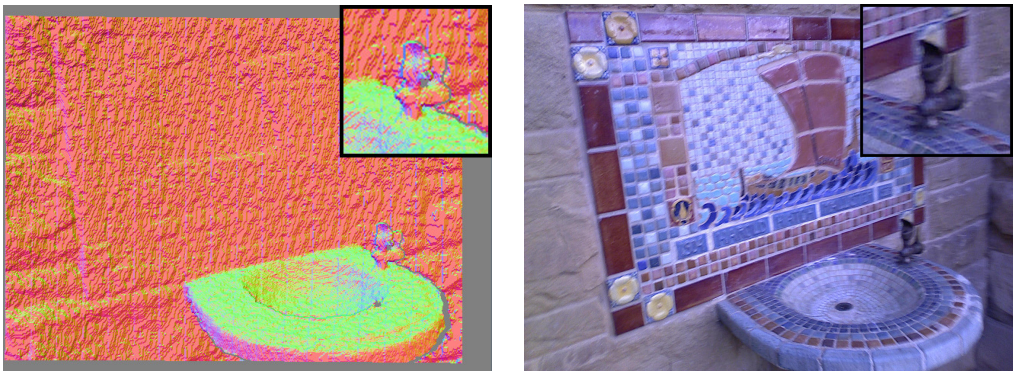
Introduction

Recent advances in Augmented Reality (AR) and Virtual Reality (VR) hardware have led from originally futuristic and bulky prototypes to portable and practical head-mounted displays (HMDs). This development has recently manifested itself in commercially available devices such as the *Microsoft Hololens 2*[1] or the *Oculus Quest*[2], which also contain advanced resource-efficient multi-camera setups. Herein, 3D reconstruction methods, along with other Computer Vision algorithms, play an essential role and drive key innovations in this field. In AR on the one hand, it is crucial to coarsely reconstruct the observed surface geometry using the cameras integrated in the device. This allows to augment the scene by overlaying additional information in a physically correct manner in order to facilitate a smooth interaction of users with the scene, and to potentially enable super-human capabilities. On the other hand, VR applications require substantially more photo-realistic 3D models of real-world objects to facilitate a realistic perception of virtual environments. The generated 3D reconstructions need to exhibit a high level of photo-realism for faithful rendering into an opaque high-resolution display, which serves as the main source of visual input to the person.

While humans can effortlessly perceive the three-dimensional environment around them, it is highly challenging for machines such as AR/VR devices to obtain a dense 3D representation of the world from captured 2D images. This sophisticated inverse problem of reconstructing 3D scene geometry from visual data has been investigated by researchers for decades and is considered as one of the classical problems in Computer Vision. While this research has resulted in the discovery of many underlying fundamental principles, there are still many open problems to be solved.

As an alternative to classical offline Structure-from-Motion (SfM) methods [9, 34, 93, 99], modern real-time 3D reconstruction algorithms are often based on Simultaneous Localization and Mapping (SLAM) approaches. Visual SLAM tackles the ill-posed

[1] https://www.microsoft.com/en-us/hololens/
[2] https://www.oculus.com/quest/



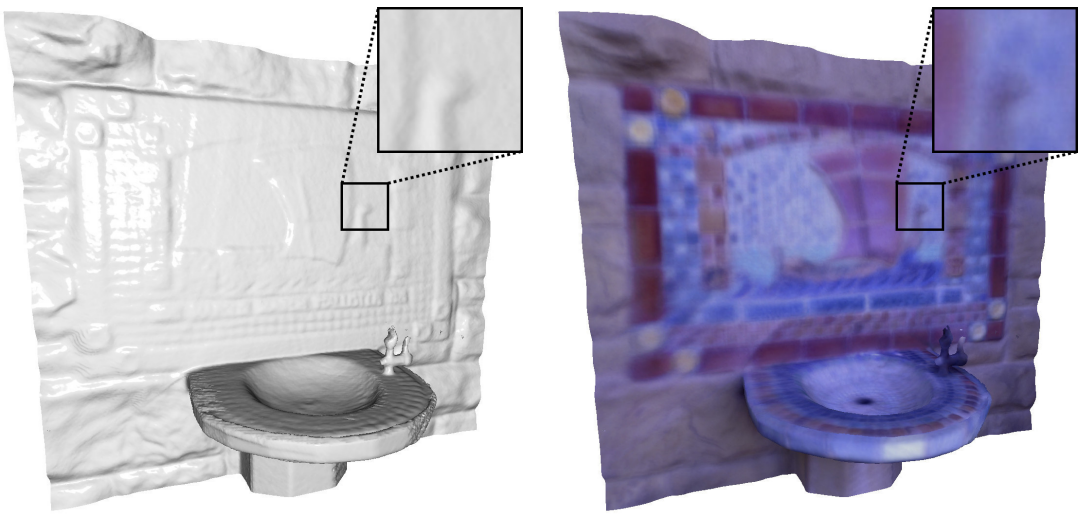
(a) Input depth (normals rendering) (b) Input color

Figure 1.1: Input frame acquired from a low-cost RGB-D sensor. The depth map (a) exhibits strong noise without fine details, the associated low-quality color image (b) often shows motion blur. This dissertation contributes methods for obtaining high-fidelity 3D reconstructions from such deficient input data.

problem of simultaneously building a map of an unknown environment and localization relative to this map from a 2D image stream, usually in a real-time setting. Low-cost commodity RGB-D cameras such as the Microsoft Kinect have proven to be especially useful in SLAM and 3D reconstruction and have led to a significant boost of research in recent years. This resulted in a large number of impactful scientific publications that revisited and adopted classical 3D reconstruction methods. Such active depth sensors capture color images along with metric per-pixel depth at real-time frame rates, which simplifies various challenging Computer Vision tasks. Compared to Visual SLAM from 2D images, the available depth information removes the scale ambiguity and allows to estimate the relative camera motion in absolute metric world coordinates. By fusing incoming frames in a common dense volumetric surface representation, it has become possible to automatically reconstruct dense 3D models of large-scale scenes of unprecedented quality in real-time. In addition to AR/VR applications, these generated 3D scans have also shown great potential in robotics, e.g. for autonomous navigation of flying drones, as well as in industrial inspection scenarios.

1.1 Motivation

State-of-the-art RGB-D based 3D reconstruction methods [28, 53, 84, 86, 120] accurately track the camera motion in real-time. The frames captured by the RGB-D sensor are fused in a volumetric Signed Distance Function (SDF) using their estimated camera poses to generate a full 3D model from multiple views and to cope with the noise in the depth measurements. For illustration, Figure 1.1 visualizes the strong noise characteristics of



(a) Oversmooth 3D surface geometry (b) Blurry surface colors

Figure 1.2: Example 3D reconstruction generated by fusing RGB-D frames in a volumetric Signed Distance Function using Voxel Hashing [86]. Despite its metric accuracy, the fused 3D surface geometry (a) is oversmooth without fine details. Furthermore, volumetric averaging leads to blurry per-voxel surface colors (b) that impair the overall visual 3D model appearance.

input depth maps and the low quality of color images that additionally often suffer from motion blur.

The 3D reconstructions produced from such input data are geometrically accurate, but the ℓ_2-regularization property in the depth and color fusion inherently leads to oversmooth surface geometry and blurry surface colors (Figure 1.2). The blurry per-voxel colors are a consequence of averaging inconsistent color samples from the input images due to slightly erroneous surface geometry and marginally inaccurate camera poses, substantially impairing the visual appearance of the 3D model. We approach this research gap in Chapters 4 and 5 and present methods for reconstructing both sharp model textures and high-quality surface geometry from low-quality RGB-D input data.

Since pure camera tracking methods inevitably accumulate drift over time, efficient global optimization methods have been developed to reduce drift whenever loop closures are detected, resulting in globally consistent camera poses even for large-scale scenes. However, as incoming RGB-D frames are integrated immediately into the SDF volume using their estimated poses, the fused model becomes outdated when camera poses are updated by the global camera pose optimization. Because the integration of RGB-D frames is computationally demanding, it is not possible to entirely re-fuse the SDF volume with the new poses during the scanning process. As a consequence, globally consistent dense 3D reconstructions are usually generated in an offline model fu-

(a) Without surface correction (b) With on-the-fly surface correction

Figure 1.3: In dense real-time 3D reconstruction, RGB-D frames are immediately integrated into an SDF volume. However, the fused 3D model becomes outdated whenever camera poses are optimized for global consistency on loop closures. Without 3D surface correction, the 3D reconstruction becomes increasingly inconsistent during scanning (a). Computationally efficient methods for correcting the surface of large-scale scenes on-the-fly (e.g. Chapter 6) enable up-to-date 3D model previews in real-time (b).

sion once the final camera poses are determined. Chapter 6 effectively addresses the research gap of enabling consistent previews of dense SDF volumes on updated camera poses. Figure 1.3 compares real-time visualizations of a 3D reconstruction during scanning without and with the proposed on-the-fly surface correction.

1.2 Literature Overview

In the following, we provide a literature overview of the state-of-the-art in high-quality 3D reconstruction from commodity RGB-D cameras, with a particular focus on the research gaps identified above. We first present related works on dense RGB-D based 3D surface reconstruction and relevant fundamentals. Afterwards, we detail modern approaches for obtaining high-quality model textures, as well as Shape-from-Shading techniques for recovering fine-scale geometry. Finally, we reference related work for on-the-fly surface correction of large environments during scanning.

1.2.1 Dense RGB-D based 3D Reconstruction

In recent years, many impressive dense 3D reconstruction frameworks have been developed that are usually composed of an RGB-D SLAM system for estimating camera poses and a common 3D model for fusing the input data. For a detailed literature overview on this topic, we refer the interested reader to the survey presented in [136].

1.2.1.1 RGB-D SLAM

By definition, a SLAM system incrementally builds and maintains a map of the environment, which it simultaneously uses for localizing itself. With RGB-D data, the 3D map can be determined in metric coordinates and the problem of scale drift in the monocular case vanishes. Modern RGB-D SLAM methods accurately estimate the absolute motion of the sensor from a continuous stream of input frames. These approaches usually combine incremental frame-to-frame tracking (odometry) with a continuous global camera pose optimization in the background. Specifically, efficient Pose Graph Optimization (PGO) or Bundle Adjustment (BA) techniques are used to reduce accumulated drift and to establish global consistency on loop closures, i.e. when places are re-visited.

Dense RGB-D SLAM methods use dense direct odometry for aligning two frames. Herein, all image pixels are leveraged to minimize a photometric [58, 104, 105] or combined photometric and geometric objective [57, 79] using projective data association. Robust norms (e.g. Huber norm) and Iteratively Reweighted Least-Squares (IRLS) effectively suppress the influence of outliers [58]. Various state-of-the-art frameworks such as RGBD-SLAM [32] or DVO-SLAM [57] use the computed relative poses as constraints in an efficient continuous PGO, which is feasible in real-time for even large-scale scenes.

Sparse SLAM systems [28, 32, 82] extract and match color image features, e.g. SIFT [76] or ORB [89], and integrate them in global BA schemes. Offline BA methods are able to achieve high accuracy, but often also exhibit unpractical runtimes [23, 7]. Advanced schemes and trade-offs (e.g. sparse keyframes only, fixed intervals, local optimization) have been developed in order to approach real-time performance with comparable accuracy [22, 45, 82]. The well-engineered GPU-based framework BundleFusion [28] achieves state-of-the-art pose estimation accuracy; its combined local and hierarchical global BA scheme allows to cope even with large-scale scenes in real-time. The dense direct BA proposed in BAD-SLAM [94] has photometric and geometric constraints and runs for smaller environments in real-time on a CPU.

Instead of optimizing camera poses using PGO or BA, it is also possible to establish global model consistency through *map deformation* of the underlying surface [120, 123]. Herein, the 3D reconstruction implicitly moves with its attached deformation pose graph that can be efficiently optimized on loop closures.

1.2.1.2 3D Surface Representations

In order to reconstruct a complete model of a real-world scene, it is necessary to fuse the noisy input RGB-D frames from multiple views in a common surface representation using the camera poses estimated by the SLAM system. This 3D data structure should effectively regularize out the depth noise and simultaneously enable real-time performance for practical applicability.

RGB-D Keyframes The depth maps of keyframes are memory efficient 2.5D representations of three-dimensional surfaces. While this limits their ability to represent occluded surfaces, keyframes are highly suitable for fusing multiple consecutive RGB-D frames locally and reducing depth noise. Keyframe Fusion (KF) has consequently shown its potential for reducing camera tracking drift, e.g. in [57, 78, 79].

Signed Distance Functions Volumetric SDFs [26, 84] are the most popular representation for fusing dense 3D models from RGB-D data and for compensating depth noise. These volumetric grids partition space into small voxels that store the surface implicitly using the signed distance to the closest surface. While real-time performance is achieved using General Purpose GPU (GPGPU) programming, scarce GPU memory limits the size and resolution of regular dense volumetric grids. Voxel Hashing [86] enables large-scale 3D reconstruction with improved memory efficiency by only allocating smaller fixed-size voxel blocks close to the surface. Sparse hierarchical SDF representations based on octrees [104, 106, 129] provide a memory efficient alternative for large-scale environments. For SDF-based models, the Marching Cubes algorithm [75] extracts a triangle mesh at the zero-crossing of the SDF.

Other 3D Representations In point-based [22, 56] and surfel-based [87, 109, 123] 3D models, memory is only allocated at the surface in the form of points with varying radius, or oriented disks respectively. OctoMap [47] uses an octree to map occupied and unoccupied space and is focused on robotic applications.

1.2.1.3 3D Reconstruction using Signed Distance Functions

Since the KinectFusion approach [84], researchers have demonstrated that dense uniform SDF voxel grids are highly suitable for reconstruction of objects and small workspaces of limited dimensions [21, 110]. The camera poses of incoming RGB-D frames are estimated using frame-to-model camera tracking, often against a raycasted view of the fused 3D model [84, 86, 119, 120] using the Iterative Closest Point (ICP) algorithm [18].

These methods have been extended to large-scale environments. In [120], the fixed-sized SDF volume on the GPU is shifted along with the camera motion while inactive parts are streamed to CPU memory. [53, 86] employ a voxel-hashed sparse 3D volume to better exploit limited GPU memory. While frame-to-model tracking accumulates drift, global consistency can be established by coupling dense 3D surface fusion with more advanced RGB-D SLAM systems that include global pose optimization. For example, Steinbrücker et al. [104] combine DVO-SLAM [57] with an octree-based 3D model. BundleFusion [28] performs real-time large-scale BA on a GPU and fuses the RGB-D frames in a voxel-hashed SDF, resulting in impressive large-scale reconstructions.

In scans of indoor environments, a semantic scene analysis may help to detect planar surfaces such as walls or floor; these plane priors can be exploited to denoise and complete room geometry [2] or even to virtually relight, remove and add furniture [130].

While the above 3D reconstruction frameworks assume static scenes, methods for dealing with dynamic objects in the scene [90, 91, 96, 108] and for the complex scenario of non-rigid template-free 3D reconstruction [50, 85, 97, 98] have been proposed.

Useful RGB-D benchmark datasets have been provided in order to quantitatively evaluate the localization accuracy of RGB-D SLAM systems [23, 42, 111]. The (augmented) ICL-NUIM benchmarks [23, 42] also contain synthetic ground truth 3D geometry, additionally allowing to evaluate the dense surface reconstruction quality.

1.2.2 High-Quality Surface Textures

The 3D models generated by fusing RGB-D frames in SDF volumes exhibit high metric accuracy with reduced noise. However, they also suffer from a lack of photo-realism, since the problem of reconstructing high-quality surface colors is mostly not approached appropriately. Per-voxel colors are computed using a weighted average of observations in the RGB input images, with simple weights proportional to (1) viewing angle, (2) inverse squared depth and (3) proximity to depth discontinuities [84, 110, 119]. In combination with the low color image quality of RGB-D sensors, this commonly leads to an overall blurry visual appearance.

Texture Mapping and Super-Resolution In Computer Graphics, methods for mapping high-resolution textures onto simplified meshes with low geometric complexity have been studied extensively over the last decades. Texel colors are computed either by blending observations from multiple input color images [24, 83, 103], including optical flow corrections [30], or by selecting the best per-face input views with minimal seams between views [36, 66, 116]. The complex variational super-resolution method in [39] achieves impressive texturing results in a controlled and well-calibrated setup.

[79] investigates super-resolution keyframe fusion using RGB-D sensors, while [62] uses screen space color filtering techniques.

Color Map Optimization Zhou and Koltun [133] optimize the mapping between color images and 3D mesh geometry for consistent surface colors. An alternating optimization of camera poses and non-rigid 2D image deformations accounts for (otherwise inexplicable) inaccuracies in the geometric image formation model. The use of per-vertex colors of an upsampled mesh is a practical limitation due to increased geometric complexity compared to texture mapping. Jeon et al. [51] present a texture coordinate optimization to maximize the photometric consistency of multiple blended keyframes projected onto a texture map of the 3D model. [35] follows a similar objective by optimizing per-vertex transformations on chart boundaries. A variational framework for jointly optimizing geometry and color in a SDF using its associated raw RGB-D input frames is presented in [55].

Most recently, visually compelling large-scale 3D reconstructions with an unprecedented level of photo-realism have been demonstrated in [107, 118]. However, instead of using low-cost commodity RGB-D sensors, a well-calibrated custom camera rig was integrated into a state-of-the-art dense 3D reconstruction framework, with extensions for reconstructing mirror and glass surfaces.

1.2.3 Shading-based Geometry Refinement

The individual depth maps acquired by commodity RGB-D sensors are characterized by strong noise. This problem of noise is successfully addressed by fusing the depth maps in a common volumetric 3D representation such as a SDF volume successfully. However, both the input depth maps as well as the over-smoothed fused 3D model do not exhibit fine-scale geometry details.

While Shape-from-Shading (SfS) [46, 131] is highly ill-posed in general, it is possible to elegantly integrate the depth maps of RGB-D frames as priors for guiding the under-constrained SfS process. Shading-based Refinement (SBR) methods are based on SfS and refine an initial coarse geometry estimate using the input color images in order to reconstruct fine-scale surface details. In particular, the depth maps help to differentiate between illumination, geometry and albedo and to resolve the Bas-Relief ambiguity [17]. Recent RGB-D based approaches often follow a pipeline that first estimates the scene lighting using Spherical Harmonics (SH) [88] and subsequently optimizes surface geometry and material albedo.

The methods in [16, 124, 125] leverage shading cues in a single-view or multi-view color video stream to (track and) enhance a coarse 3D shape model, leading to high-

quality surfaces. Single-frame methods for SBR recover fine-scale geometry from an initial dense depth map obtained from an RGB-D sensor [14, 15, 41, 128], which has been shown to be feasible even in real-time [31, 126]. Häfner et al. [40] combine SfS and single-frame depth super-resolution within a variational framework for generating high-fidelity depth maps. There are also methods for additionally reconstructing advanced material reflectance with specularities, e.g. for an acquired shape template in a dynamic setting [72], or for RGB-D input [127]. Langguth et al. [64] propose a combined shading-aware Multi-View Stereo (MVS) approach, successfully demonstrating the complementary nature of both techniques.

Volumetric Shading-based Refinement [135] allows to refine even full 3D models stored in an SDF volume, leading to visually impressive reconstruction quality. The approach is limited by its very simple lighting estimation using only nine SH parameters for the entire scene, and by a sequential pipeline that potentially cannot recover from bundle adjusted, but possibly imperfect initial camera poses.

A detailed survey on techniques for 3D reconstruction of both surface geometry and reflectance properties is given in [117].

1.2.4 Large-scale Online Surface Correction

Modern 3D reconstruction methods estimate globally consistent camera poses for large-scale scenes in real-time. RGB-D frames are immediately fused into a common SDF volume, which in most approaches is not corrected on poses updates on loop closures due to the high computational demands. Instead, a full integration pass in a separate post-processing step finally fuses all RGB-D frames again into the SDF with the final camera poses. Only few methods manage to correct the dense 3D surface fused in Signed Distance Functions on-the-fly during the reconstruction process.

In [33], dense SDF subvolumes of fixed size are fused locally and globally registered in a global optimization of subvolume poses. The surfaces in the subvolumes are updated on subvolume shifts using volume blending. The lack of an explicit loop closure detection fails to compensate for drift in large-scale settings. Similarly, the method of Kähler et al. [52] relies on subvolumes that are updated online during raycasting. The camera is tracked against multiple submaps independently and the estimated poses are used as constraints in a global pose optimization. However, the runtime of subvolume-based methods does generally not scale well with an increasing number of subvolumes.

Some approaches attach the reconstructed surface to a deformation graph, e.g. dense SDF models [121, 122] or a dense surfel map [95, 123]. On loop closures, the model is corrected online along with the pose graph using an as-rigid-as-possible map deformation. Problematically, it is not possible to recover surface information that has been lost

through inconsistent fusion, for example due to significant drift in the camera poses before loop closures are detected.

BundleFusion [28] has a sparse voxel-hashed SDF volume at its core, which is corrected on-the-fly on pose graph updates by re-integrating RGB-D frames. To gradually adapt the 3D model to the pose changes, frames are first de-integrated and then re-integrated with their new poses. The framework is computationally very demanding, since (1) almost all of the previous RGB-D frames possibly need to be re-integrated on loop closures, while (2) they simultaneously have to be kept in limited GPU memory. The RKD-SLAM system [71] follows the main idea of our keyframe re-integration method proposed in Chapter 6.

1.3 Outline of the Thesis

This cumulative dissertation is structured into four parts as follows:

Part I introduces the research problem motivating this thesis and provides the underlying theoretical background and methodology. In particular, Chapter 1 gives an introductory motivation of the research topic as well as a detailed literature overview of the state-of-the-art in RGB-D based 3D reconstruction and online surface correction. In Chapter 2, the main contributions of this work are presented along with an overview of the respective peer-reviewed publications. Chapter 3 establishes the mathematical tools and fundamental concepts for 3D reconstruction from RGB-D sensors.

Part II presents the three peer-reviewed publications that form the cumulative content of this thesis. Firstly, Chapter 4 presents a method for improving the visual appearance of SDF-based 3D models [6] by combining modern texture mapping approaches with advanced filtering techniques to obtain consistent and sharp surface textures. Secondly, a joint surface reconstruction approach [4] based on Shape-from-Shading techniques is introduced in Chapter 5; it improves the color sampling in the input frames by simultaneously optimizing for geometry encoded in an SDF, textures from keyframes, and their camera poses along with material and spatially-varying scene lighting. Lastly, Chapter 6 proposes an efficient online surface correction method for updating dense 3D reconstructions of large-scale scenes on-the-fly on pose graph updates [5].

Part III provides a summary of the thesis in Chapter 7. Finally, Chapter 8 discusses the limitations of the proposed approaches and shows future research directions.

Part IV contains the supplementary material for two of the included publications in Chapters A and B, as well as the original versions of the papers [4, 5, 6] along with detailed disclaimers in Chapter C.

Chapter 2

Contributions

The general aim of this thesis is to develop novel methods for high-quality 3D reconstruction of real-world objects and scenes from low-cost RGB-D sensors. In this chapter, we summarize the detailed contributions of the publications that investigate this problem and form the cumulative content of this dissertation.

2.1 List of Publications

The main contributions of this cumulative thesis are based on the three full-length publications [4, 5, 6] included in Chapters 4, 5 and 6. These peer-reviewed papers are joint work with Daniel Cremers, Jan Kautz, Kihwan Kim, Matthias Nießner, Raphael Schaller and Jörg Stückler and were published in highly ranked international conferences. Table 2.1 provides a full list of the peer-reviewed publications that contribute to this dissertation. Additionally, the table contains other co-authored papers published while pursuing this degree, which are however not included as a contribution.

The research in [4] was in most parts conducted during a research internship at NVIDIA in Santa Clara, CA. The paper [5] includes results of Raphael Schaller's Master's thesis [92], which was supervised by the author of this dissertation. The publication [7] is based on the Master's thesis [77]. The paper [1] received the *Honourable Mention award* at the Scandinavian Conference on Image Analysis (SCIA) 2019.

Submap-based Bundle Adjustment for 3D Reconstruction from RGB-D Data. Robert Maier, Jürgen Sturm and Daniel Cremers. In: *German Conference on Pattern Recognition (GCPR) 2014* [7].

Super-Resolution Keyframe Fusion for 3D Modeling with High-Quality Textures. Robert Maier, Jörg Stückler and Daniel Cremers. In: *International Conference on 3D Vision (3DV) 2015* [6] (Chapter 4).

De-noising, Stabilizing and Completing 3D Reconstructions On-the-go using Plane Priors. Maksym Dzitsiuk, Jürgen Sturm, Robert Maier, Lingni Ma and Daniel Cremers. In: *International Conference on Robotics and Automation (ICRA) 2017* [2].

Efficient Online Surface Correction for Real-time Large-Scale 3D Reconstruction. Robert Maier, Raphael Schaller and Daniel Cremers. In: *British Machine Vision Conference (BMVC) 2017* [5] (Chapter 6).

Multiframe Scene Flow with Piecewise Rigid Motion. Vladislav Golyanik, Kihwan Kim, Robert Maier, Matthias Nießner, Didier Stricker and Jan Kautz. In: *International Conference on 3D Vision (3DV) 2017* [3].

Intrinsic3D: High-Quality 3D Reconstruction by Joint Appearance and Geometry Optimization with Spatially-Varying Lighting. Robert Maier, Kihwan Kim, Daniel Cremers, Jan Kautz and Matthias Nießner. In: *International Conference on Computer Vision (ICCV) 2017* [4] (Chapter 5).

A Non-invasive 3D Body Scanner and Software Tool towards Analysis of Scoliosis. Susmita Roy, Alexander T. D. Grünwald, Ana Alves-Pinto, Robert Maier, Daniel Cremers, Daniela Pfeiffer and Renee Lampe. In: *BioMed Research International (BMRI) 2019* [8].

Combining Depth Fusion and Photometric Stereo for Fine-Detailed 3D Models. Erik Bylow, Robert Maier, Fredrik Kahl and Carl Olsson. In: *Scandinavian Conference on Image Analysis (SCIA) 2019* [1].

Table 2.1: Full list of peer-reviewed publications done within the course of this degree, in chronological order. The three publications that contribute to this cumulative dissertation are listed in black, with references to the respective chapters. Research papers not included in this thesis are marked in gray.

2.2 Major Contributions

The key contributions of this cumulative dissertation are twofold. First, we explore algorithms for recovering high-quality textures and fine-scale geometric details from 3D models fused in Signed Distance Fields with oversmooth geometry and blurry per-voxel surface colors. Second, we present an efficient real-time surface correction method that re-integrates keyframes to update dense 3D reconstructions of large-scale scenes on-the-fly on loop closures, enabling up-to-date model previews already during 3D scanning.

2.2.1 Recovering High-Quality Textures and Geometric Details

In order to reduce the strong noise in the depth data from consumer-grade RGB-D sensors, state-of-the-art approaches commonly fuse multiple depth maps in a volumetric Signed Distance Function (SDF). While the reconstructed 3D models are geometrically accurate, their oversmooth 3D surface geometry lacks fine-scale details. Furthermore, the fused per-voxel colors are blurry and consequently not photorealistic enough for many real-world use cases (e.g. AR/VR).

Chapter 4 is based on [6] and presents a novel fast and robust color mapping technique to obtain 3D models with high-quality visual appearance. To achieve this, it combines modern texture mapping approaches with super-resolution keyframe fusion. During the scan, a direct keyframe-based SLAM frontend [57] estimates a globally consistent camera trajectory. After fusing the input RGB-D frames into a global dense SDF volume [26], a 3D triangle mesh is extracted. In parallel, these low-resolution RGB-D frames are first deblurred using advanced filtering techniques and then fused into super-resolution keyframes. Finally, a texture mapping method [67] textures the reconstructed 3D mesh from these high-quality keyframes, resulting in consistent and sharp model colors. The experimental evaluation proves that the proposed method is highly effective and that the obtained visual model appearance has superior quality compared to the state-of-the-art.

Chapter 5 is based on [4] and introduces a joint surface reconstruction approach that achieves both high quality geometry and appearance. We follow established Shape-from-Shading techniques and first introduce a novel lighting estimation method that estimates spatially-varying spherical harmonics from subvolumes of the reconstructed scene. A shading-based refinement then simultaneously optimizes for per-voxel geometry and material albedo stored in a sparse SDF, and for camera poses (and camera intrinsics) of the utilized input keyframes. The keyframes are selected automatically from all input frames based on a blurriness measure in order to discard harmful frames with motion blur or other artifacts. Since the joint optimization recovers fine-scale geometry

details and also refines the camera poses in the image formation model, we implicitly obtain consistent high-quality surface textures as a by-product through the improved color sampling in the input frames. An extensive evaluation demonstrates that the developed lighting estimation better approximates complex global scene illumination and that the proposed joint surface optimization generates impressive results with fine-scale details of scene geometry along with sharp surface textures.

2.2.2 Efficient On-the-fly Surface Correction

As the methods for recovering fine geometry and sharp textures presented in Section 2.2.1 are designed as a post-processing step, there are no practical hard constraints w.r.t. runtime or memory efficiency. However, reconstructing globally consistent dense 3D reconstructions becomes significantly more challenging in real-time scenarios. Modern real-time RGB-D based 3D reconstruction frameworks such as [57] detect loop closures and use a continuous pose graph optimization to reduce drift and accurately estimate the global camera motion even for larger scenes. As the fusion of the input RGB-D frames in a dense SDF is very costly, it is highly challenging to maintain a correct 3D model whenever camera poses are updated on loop closures. As a consequence, dense surface fusion is often implemented as a post-process using only the final camera poses.

Chapter 6 is based on [5] and investigates an efficient method for correcting the reconstructed surface on-the-fly on pose updates computed by the SLAM framework. Updating large-scale dense 3D reconstructions in real-time immediately increases the global consistency of the geometry, e.g. for collision detection in AR or in robotic scenarios, and enhances the visual quality of online model previews. As frames are streamed from the RGB-D sensor, the real-time RGB-D SLAM system [57] estimates the respective camera poses. In a background thread, a continuous pose graph optimization compensates for global drift on loop closures. We directly fuse consecutive input RGB-D frames into keyframes, similar to Chapter 4, and integrate them into a sparse SDF volume on the GPU based on Voxel Hashing [86]. As the SLAM system detects loop closures and updates the keyframe poses, the dense SDF-based 3D model becomes outdated. We therefore correct the SDF volume on-the-fly by de-integrating keyframes using their previous poses and re-integrating them again using their updated camera poses. To improve efficiency and enable real-time performance, we introduce a novel intelligent keyframe selection and re-integration strategy, which significantly reduces streaming between GPU and host. A quantitative evaluation shows that our method outperforms the state-of-the-art w.r.t. runtime efficiency (up to 93%) and memory requirements. This allows to perform large-scale online surface correction in real-time using only a single GPU, while simultaneously maintaining a comparable surface quality.

Chapter 3

Fundamentals

This chapter establishes the fundamental concepts and mathematical tools for dense 3D reconstruction from RGB-D data. We therefore first introduce the mathematical preliminaries used throughout this dissertation, derived from standard literature such as [43, 112]. Additionally, we present the technical principles of RGB-D sensors as well as theoretical background on camera tracking, dense 3D surface reconstruction and Shape-from-Shading techniques.

3.1 Mathematical Preliminaries

Notation Throughout this work, scalars are denoted as $s \in \mathbb{R}$ with regular (lowercase) letters, while vectors $\boldsymbol{x} \in \mathbb{R}^n$ (lowercase) and matrices $\boldsymbol{M} \in \mathbb{R}^{m \times n}$ (uppercase) are written in bold letters. If not specified otherwise, we denote 3D points as $\boldsymbol{p} = (x, y, z)^\top \in \mathbb{R}^3$ and 2D pixels as $\boldsymbol{x} = (u, v)^\top \in \mathbb{R}^2$; their respective homogenous coordinates are $\tilde{\boldsymbol{p}} = (x, y, z, 1)^\top \in \mathbb{R}^4$ and $\tilde{\boldsymbol{x}} = (u, v, 1)^\top \in \mathbb{R}^3$.

3.1.1 Rigid Body Motion

3D reconstruction tries to recover geometry and appearance of objects in a three-dimensional world. Different rigid objects are hereby spatially related through *rigid body motions*, which preserve both distance and orientation for any point pair on a rigid body and are consequently highly relevant for all 3D reconstruction methods. The space of all rigid body motions forms a Lie group, which is also known as *Special Euclidean group* $\mathbb{SE}(3)$. A 3D Euclidean transform with 6 Degrees-of-Freedom (DoF) consists of a 3-DoF translation and a 3-DoF rotation. We discuss different relevant representations in the following.

Transformation Matrix We commonly use a 4×4 *transformation matrix* $\boldsymbol{T}_A^B \in \mathbb{SE}(3)$ to transform a 3D point $\boldsymbol{p}_A$ from coordinate frame A into frame B. The Special Euclidean group $\mathbb{SE}(3)$ is formally defined as follows:

$$\mathbb{SE}(3) = \left\{ \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^\top & 1 \end{bmatrix} \;\middle|\; \boldsymbol{R} \in \mathbb{SO}(3), \boldsymbol{t} \in \mathbb{R}^3 \right\}, \tag{3.1}$$

where $\boldsymbol{t} \in \mathbb{R}^3$ is a 3D translation vector and the orthonormal 3×3 rotation matrix $\boldsymbol{R}$ belongs to the *Special Orthogonal group* $\mathbb{SO}(3)$ with 3 DoF:

$$\mathbb{SO}(3) = \left\{ \boldsymbol{R} \in \mathbb{R}^{3\times3} \;\middle|\; \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I}_3 \wedge \det(\boldsymbol{R}) = +1 \right\}. \tag{3.2}$$

More concretely, we can write the full homogenous 4×4 transformation matrix $\boldsymbol{T}_A^B$ from frame A to frame B as

$$\boldsymbol{T}_A^B = \begin{bmatrix} \boldsymbol{R}_A^B & \boldsymbol{t}_A^B \\ \boldsymbol{0}^\top & 1 \end{bmatrix}, \tag{3.3}$$

and the respective inverse transformation $\boldsymbol{T}_B^A \in \mathbb{SE}(3)$, which maps back from frame B to frame A, as

$$\boldsymbol{T}_B^A = (\boldsymbol{T}_A^B)^{-1} = \begin{bmatrix} \boldsymbol{R}_A^{B\top} & -\boldsymbol{R}_A^{B\top} \boldsymbol{t}_A^B \\ \boldsymbol{0}^\top & 1 \end{bmatrix}. \tag{3.4}$$

This matrix representation allows to easily transform a 3D point $\boldsymbol{p}_A$ in frame A to the respective 3D point $\boldsymbol{p}_B$ in coordinate frame B:

$$\boldsymbol{p}_B = \boldsymbol{R}_A^B \boldsymbol{p}_A + \boldsymbol{t}_A^B, \tag{3.5}$$

or through matrix-vector-multiplication using homogenous coordinates:

$$\tilde{\boldsymbol{p}}_B = \boldsymbol{T}_A^B \tilde{\boldsymbol{p}}_A. \tag{3.6}$$

Analogously, two (or more) transformation matrices $\boldsymbol{T}_A^B$ and $\boldsymbol{T}_B^C$ can be concatenated to transform from frame A to frame C:

$$\boldsymbol{T}_A^C = \boldsymbol{T}_B^C \boldsymbol{T}_A^B. \tag{3.7}$$

While the translation vector $\boldsymbol{t} \in \mathbb{R}^3$ is already a minimal representation with 3 parameters for the 3 translational DoF, the rotation matrix $\boldsymbol{R} \in \mathbb{SO}(3)$ with 9 parameters is an over-parametrization of the underlying 3 rotational DoF. This can lead to problems during numerical optimization of transformations, as additional constrains need to ensure the mathematical properties of the orthonormal rotation matrix.

Twist Coordinates The theory of Lie Groups yields an elegant minimal representation for 3D Euclidean transformations. Concretely, the Special Euclidean group $\mathbb{SE}(3)$ and Special Orthogonal group $\mathbb{SO}(3)$ introduced above are Lie groups, which are smooth differentiable manifolds with group operations that are compatible with the smooth structure. Every Lie group has a corresponding *Lie algebra* that defines the *tangent space* around the group's identity element $\boldsymbol{I}$. We loosely follow the tutorial in [19] to explain the Lie algebras $\mathfrak{so}(3)$ and $\mathfrak{se}(3)$ associated with the groups $\mathbb{SO}(3)$ and $\mathbb{SE}(3)$ respectively. With only 6 parameters for 6 DoF, the so-called *twist coordinates* $\boldsymbol{\xi} \in \mathbb{R}^6$ in the Lie algebra $\mathfrak{se}(3)$ are highly suitable for numerical optimization:

$$\boldsymbol{\xi} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3, \boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \boldsymbol{\omega}_3)^\top = \left(\boldsymbol{v}^\top, \boldsymbol{\omega}^\top\right)^\top \in \mathbb{R}^6, \tag{3.8}$$

where $\boldsymbol{v} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3)^\top$ encodes the linear velocity and $\boldsymbol{\omega} = (\boldsymbol{\omega}_1, \boldsymbol{\omega}_2, \boldsymbol{\omega}_3)^\top$ determines the angular velocity. With the operator $[\cdot]_\times$, which generates the skew-symmetric matrix of a vector, we can describe the twist as follows:

$$\hat{\boldsymbol{\xi}} = \begin{bmatrix} 0 & -\boldsymbol{\omega}_3 & \boldsymbol{\omega}_2 & \boldsymbol{v}_1 \\ \boldsymbol{\omega}_3 & 0 & -\boldsymbol{\omega}_1 & \boldsymbol{v}_2 \\ -\boldsymbol{\omega}_2 & \boldsymbol{\omega}_1 & 0 & \boldsymbol{v}_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} [\boldsymbol{\omega}]_\times & \boldsymbol{v} \\ \boldsymbol{0}^\top & 0 \end{bmatrix}. \tag{3.9}$$

The tangent space $\mathfrak{so}(3)$ of the Special Orthogonal group $\mathbb{SO}(3)$ is defined as the set of all skew-symmetric matrices:

$$\mathfrak{so}(3) = \left\{ [\boldsymbol{\omega}]_\times \in \mathbb{R}^{3\times 3} \;\middle|\; \boldsymbol{\omega} \in \mathbb{R}^3 \right\}. \tag{3.10}$$

The set of all twist coordinates forms the tangent space $\mathfrak{se}(3)$ of the group $\mathbb{SE}(3)$:

$$\mathfrak{se}(3) = \left\{ \hat{\boldsymbol{\xi}} = \begin{bmatrix} [\boldsymbol{\omega}]_\times & \boldsymbol{v} \\ \boldsymbol{0}^\top & 0 \end{bmatrix} \;\middle|\; [\boldsymbol{\omega}]_\times \in \mathfrak{so}(3), \boldsymbol{v} \in \mathbb{R}^3 \right\}. \tag{3.11}$$

The *logarithm map* and the *exponential map* transform elements from a Lie group to its associated Lie algebra and vice versa; i.e. in the case of a transformation matrix $\boldsymbol{T} \in \mathbb{SE}(3)$ and its twist $\boldsymbol{\xi} \in \mathfrak{se}(3)$:

$$\boldsymbol{T} = \exp(\boldsymbol{\xi}) : \mathfrak{se}(3) \to \mathbb{SE}(3) \tag{3.12}$$

$$\boldsymbol{\xi} = \log(\boldsymbol{T}) : \mathbb{SE}(3) \to \mathfrak{se}(3) \tag{3.13}$$

Close to the identity (i.e. for small motions with $\boldsymbol{\xi} \approx \mathbf{0}$), the exponential map can be approximated such that $\boldsymbol{T} \approx \boldsymbol{I}_4 + \hat{\boldsymbol{\xi}}$. We obtain a closed-form solution for the rotation $\boldsymbol{R} \in \mathbb{SO}(3)$ and the translation $\boldsymbol{t} \in \mathbb{R}^3$ using the Rodrigues' formula (with the angle $\theta = \|\boldsymbol{\omega}\|$):

$$\boldsymbol{R} = \exp([\boldsymbol{\omega}]_\times) = \boldsymbol{I}_3 + \frac{\sin\theta}{\theta}[\boldsymbol{\omega}]_\times + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\omega}]_\times^2 \tag{3.14}$$

$$\boldsymbol{t} = \left(\boldsymbol{I}_3 + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\omega}]_\times + \frac{\theta-\sin\theta}{\theta^3}[\boldsymbol{\omega}]_\times^2\right)\boldsymbol{v}. \tag{3.15}$$

The logarithm map for obtaining the twist $(\boldsymbol{v}^\top, \boldsymbol{\omega}^\top)^\top \in \mathbb{R}^6$ from $\boldsymbol{R}$ and $\boldsymbol{t}$ has the following closed-form expression:

$$\theta = \arccos\left(\frac{\mathrm{tr}(\boldsymbol{R})-1}{2}\right) \tag{3.16}$$

$$\log(\boldsymbol{R}) = \frac{\theta}{2\sin\theta}\left(\boldsymbol{R} - \boldsymbol{R}^\top\right) \tag{3.17}$$

$$\boldsymbol{\omega} = [\log(\boldsymbol{R})]^\vee \tag{3.18}$$

$$\boldsymbol{v} = \left(\boldsymbol{I}_3 - \frac{1}{2}[\boldsymbol{\omega}]_\times + \frac{\left(1 - \frac{\theta\cos(\theta/2)}{2\sin(\theta/2)}\right)}{\theta^2}[\boldsymbol{\omega}]_\times^2\right)\boldsymbol{t}, \tag{3.19}$$

where the *vee operator* $[\cdot]^\vee$ extracts the underlying generating 3D vector from a skew-symmetric 3×3 matrix.

3.1.2 Pinhole Camera Model

As cameras capture information of real objects in an image, it is crucial to accurately model the imaging process that relates the 3D scene with the 2D image. On the one hand, the *extrinsic camera parameters* describe the pose, i.e. 3-DoF orientation and 3-DoF position (Section 3.1.1), of the camera relative to a global reference coordinate frame. On the other hand, the *intrinsic camera parameters* describe the mapping of 3D points in the local camera coordinate frame to 2D pixels using a projection function $\pi : \mathbb{R}^3 \to \mathbb{R}^2$.

All methods presented in this work are based on the well-known *pinhole camera model* as projection function, which assumes an infinitely small aperture (i.e. only a single point) without a lens for focusing light.

Projection Function In the basic pinhole camera model, a 3D point $\boldsymbol{p} = (x, y, z)^\top$ is first projected onto a plane located at $z = 1$ using the *general perspective projection* $\pi_g : \mathbb{R}^3 \to \mathbb{R}^3$:

$$\pi_g(\boldsymbol{p}) = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{z} \begin{pmatrix} x \\ y \\ z \end{pmatrix}. \tag{3.20}$$

This is followed by an affine mapping with the so-called intrinsic *camera calibration matrix* $\boldsymbol{K} \in \mathbb{R}^{3\times3}$:

$$\boldsymbol{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.21}$$

where f_x, f_y describe the focal length and c_x, cy describe the principal point (image center pixel) of the camera in x- and y-direction respectively, expressed in pixel units. The homogenous 2D pixel position $\tilde{\boldsymbol{x}}$ of a 3D point is then given by:

$$\tilde{\boldsymbol{x}} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \boldsymbol{K}\pi_g(\boldsymbol{p}). \tag{3.22}$$

We combine equations 3.20, 3.21 and 3.22 into the projection function $\pi : \mathbb{R}^3 \to \mathbb{R}^2$:

$$\boldsymbol{x} = \pi(\boldsymbol{p}) = \begin{pmatrix} f_x \frac{x}{z} + c_x \\ f_y \frac{y}{z} + c_y \end{pmatrix} \tag{3.23}$$

Inverse Projection With the associated depth z for a 2D pixel $\boldsymbol{x} = (u, v)^\top$ given, e.g. as it is usually the case for RGB-D sensors, we can re-project the pixel back to 3D with the *inverse projection* $\pi^{-1} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$ (*back-projection*):

$$\boldsymbol{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \pi^{-1}(\boldsymbol{x}, z) = \boldsymbol{K}^{-1}(\boldsymbol{p}) \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} z = \begin{pmatrix} \frac{(u-c_x)}{f_x} z \\ \frac{(v-c_y)}{f_y} z \\ z \end{pmatrix}. \tag{3.24}$$

Radial and Tangential Lens Distortion In practice, the pinhole camera model is not sufficient to fully model the behavior of imperfect real-world lenses. To this end, non-linear functions $\gamma : \mathbb{R}^3 \to \mathbb{R}^3$ have been developed to remove *radial and tangential lens distortion*. The distorted 2D pixel coordinates $\tilde{\boldsymbol{x}}_d = (u_d, v_d, 1)^\top = \pi_g(\tilde{\boldsymbol{p}}_d)$ of a distorted 3D point $\boldsymbol{p}_d$ are mapped to undistorted coordinates $\tilde{\boldsymbol{x}} = (u, v, 1)^\top$ as follows:

$$\tilde{\boldsymbol{x}} = \boldsymbol{K}\gamma(\tilde{\boldsymbol{x}}_d). \tag{3.25}$$

There are various closed-form approximations of γ that approximate the distortion of real lenses using polynomials. While higher-degree approximations are more expressive, they become more difficult to calibrate and are often numerically unstable. The following distortion function is very common and effective for consumer-grade cameras, with 6 coefficients $\kappa_1, \ldots, \kappa_6$ for radial distortion, 2 coefficients ρ_1, ρ_2 for tangential distortion and $r_d^2 = u_d^2 + v_d^2$:

$$\gamma_0(\tilde{\boldsymbol{x}}) = \begin{pmatrix} u_d \frac{1+\kappa_1 r_d^2+\kappa_2 r_d^4+\kappa_3 r_d^6}{1+\kappa_4 r_d^2+\kappa_5 r_d^4+\kappa_6 r_d^6} + 2\rho_1 u_d v_d + \rho_2(r_d^2 + 2u_d^2) \\ v_d \frac{1+\kappa_1 r_d^2+\kappa_2 r_d^4+\kappa_3 r_d^6}{1+\kappa_4 r_d^2+\kappa_5 r_d^4+\kappa_6 r_d^6} + \rho_1(r_d^2 + 2v_d^2) + 2\rho_2 u_d v_d \\ 1 \end{pmatrix}. \tag{3.26}$$

However, there is no closed-form inverse function for γ_0.

In Chapter 5, we use the following simple and yet effective distortion model

$$\gamma_1(\tilde{\boldsymbol{x}}) = \begin{pmatrix} u_d(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4) + 2\rho_1 u_d v_d + \rho_2(r_d^2 + 2u_d^2) \\ v_d(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4) + \rho_1(r_d^2 + 2v_d^2) + 2\rho_2 u_d v_d \\ 1 \end{pmatrix}, \tag{3.27}$$

which is incorporated into the overall camera projection function $\pi : \mathbb{R}^3 \to \mathbb{R}^2$:

$$\tilde{\boldsymbol{x}}' = (u', v', 1)^\top = \gamma_1(\pi_g(\boldsymbol{p})) \tag{3.28}$$

$$\boldsymbol{x} = \pi(\tilde{\boldsymbol{x}}') = \begin{pmatrix} f_x u' + c_x \\ f_y v' + c_y \end{pmatrix}. \tag{3.29}$$

As a remark, the distortion coefficients are invariant w.r.t. upscaling and downscaling of images. All intrinsic camera parameters above (i.e. focal length, center pixel, radial and tangential distortion coefficients) are in practice estimated using a calibration rig.

3.1.3 Non-Linear Least-Squares Optimization

Non-Linear Least-Squares (NLS) is a commonly used mathematical tool to solve inverse problems in Computer Vision, often with the goal of finding parameters for a best fit between a given model and noisy measurements. In particular, we want to find unknown model parameters $\boldsymbol{\theta} \in \mathbb{R}^m$ that minimize an energy function:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} E(\boldsymbol{\theta}). \tag{3.30}$$

At the core of the energy is a vector-valued function $\boldsymbol{r} : \mathbb{R}^m \to \mathbb{R}^n$:

$$E(\boldsymbol{\theta}) = \frac{1}{2} \sum_i r_i(\boldsymbol{\theta})^2 = \frac{1}{2} \left\| \boldsymbol{r}(\boldsymbol{\theta}) \right\|^2, \tag{3.31}$$

where the n *residuals* are defined using the vector function $r_i(\boldsymbol{\theta}) : \mathbb{R}^m \to \mathbb{R}$. We mainly deal with non-linear and non-convex functions $r_i(\boldsymbol{\theta})$ in this thesis. The non-convexity of the energy function requires a good initial estimate of the optimization parameters in order to avoid convergence to local minima.

Linearization In the case of a linear cost function $\boldsymbol{r}(\boldsymbol{\theta})$ (*Linear Least-Squares*), there are efficient methods to solve for the parameters $\boldsymbol{\theta}$. Depending on the number of parameters and the structure of the problem, numerical techniques such as the *Moore-Penrose Pseudoinverse*, *Singular Value Decomposition*, or *Cholesky Decomposition* can be applied.

However, in the non-linear case, a common strategy is to iteratively linearize the non-linear function $\boldsymbol{r}(\boldsymbol{\theta})$ around the current parameter state and approximate the energy $E(\boldsymbol{\theta})$ quadratically. Starting with an initial estimate $\boldsymbol{\theta}_0$, the parameters are iteratively updated through incremental updates Δ in a way that $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \Delta$ minimizes the linearized objective function at each iteration k.

Gauss-Newton The *Gauss-Newton (GN)* algorithm is an iterative method to solve NLS problems that uses a linear approximation of the vector function $\boldsymbol{r}(\boldsymbol{\theta})$. It makes use of the Jacobian matrix $\boldsymbol{J}_{\boldsymbol{r}}$, which contains the function's partial derivatives (i.e. gradient) evaluated at $\boldsymbol{\theta}$:

$$\boldsymbol{J}_{\boldsymbol{r}} \coloneqq \boldsymbol{J}_{\boldsymbol{r}}(\boldsymbol{\theta}) = \left[J(\boldsymbol{\theta})_{ij} \right] = \left[\frac{\partial r_i}{\partial x_j}(\boldsymbol{\theta}) \right] \in \mathbb{R}^{n \times m} \tag{3.32}$$

We employ the *first-order Taylor expansion* to approximate the objective function linearized around $\boldsymbol{\theta}$:

$$E(\boldsymbol{\theta} + \Delta) = \frac{1}{2} \left\| \boldsymbol{r}(\boldsymbol{\theta} + \Delta) \right\|^2 \tag{3.33}$$

$$\approx \frac{1}{2} \left\| \boldsymbol{r}(\boldsymbol{\theta}) + \boldsymbol{J}_{\boldsymbol{r}} \Delta) \right\|^2 \tag{3.34}$$

$$= \frac{1}{2} \boldsymbol{r}(\boldsymbol{\theta})^\top \boldsymbol{r}(\boldsymbol{\theta}) + \Delta^\top \boldsymbol{J}_{\boldsymbol{r}}^\top \boldsymbol{r}(\boldsymbol{\theta}) + \frac{1}{2} \Delta^\top \boldsymbol{J}_{\boldsymbol{r}}^\top \boldsymbol{J}_{\boldsymbol{r}} \Delta. \tag{3.35}$$

In order to obtain the optimal value of Δ that minimizes the energy in Equation (3.35), we set its derivative to zero. This gives rise to the so-called update equation of the Gauss-Newton algorithm:

$$\boldsymbol{J}_{\boldsymbol{r}}^\top \boldsymbol{J}_{\boldsymbol{r}} \Delta = -\boldsymbol{J}_{\boldsymbol{r}}^\top \boldsymbol{r}(\boldsymbol{\theta}). \tag{3.36}$$

We solve these *normal equations* above for the parameter update Δ and increment the estimate of the optimization parameters:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \Delta. \tag{3.37}$$

To obtain the final parameter estimate $\hat{\boldsymbol{\theta}}$, the steps above are iterated until convergence.

Levenberg-Marquardt While the Gauss-Newton algorithm has fast quadratic convergence close to the local minimum, it can also show unstable behaviour for poor parameter initializations. On the other hand, *Gradient Descent (GD)* shows slow linear convergence, but always decreases the function for a sufficiently small step size. To overcome the weaknesses of both GN and GD, the *Levenberg-Marquardt (LM)* algorithm smoothly switches between both methods through an adaptive dampening parameter $\lambda > 0$. Automatically adjusting λ from iteration to iteration makes the method highly adaptive. The update equation of the LM algorithm is written as:

$$(\boldsymbol{J_r}^\top \boldsymbol{J_r} + \lambda \ \mathrm{diag}(\boldsymbol{J_r}^\top \boldsymbol{J_r}))\Delta = -\boldsymbol{J_r}^\top \boldsymbol{r}(\boldsymbol{\theta}), \tag{3.38}$$

where the diagonal matrix $\mathrm{diag}(\boldsymbol{J_r}^\top \boldsymbol{J_r})$ contains the diagonal entries of $\boldsymbol{J_r}^\top \boldsymbol{J_r}$.

For estimates far from the local minimum, the normal equations are approximated by the GD update step through a large λ. Despite slower convergence, this allows to decrease the cost also in problematic situations. For small λ, the method behaves like GN with fast convergence close to the optimum. There are different, mostly problem-specific algorithms for adjusting the dampening parameter. Generally, if the obtained parameter update Δ reduces the overall error, λ is decreased before the next iteration. Otherwise, if the computed Δ leads to an increased error, we repeatedly increase λ and solve again for Δ until the error is decreased. The dynamic adjustment hence involves solving the augmented normal equations multiple times for different values of λ.

We use the Levenberg-Marquardt implementation in *Ceres Solver* [10] to tackle the large-scale optimization problem developed in Chapter 5.

Iteratively Reweighted Least-Squares The ℓ_2-norm in least-squares problems assumes Gaussian distributed measurements, which is however unrealistic in practice. To mitigate this insufficiency and additionally cope with problematic large outlier residuals, robust optimization uses robust error norms to resemble more suitable residual distributions. For example, the *Huber norm* effectively penalizes outlier residuals linearly, instead of quadratic costs associated with the ℓ_2-norm.

In *Iteratively Reweighted Least-Squares (IRLS)*, the robust norm is used to pre-compute fixed per-residual weights in every iteration. While this implies that the Jacobian is computed without dependency of the robust weight on the residual, the problem then becomes a simple-to-solve weighted least-squares problem.

3.2 RGB-D Sensing

Low-cost commodity RGB-D sensors have led to a major boost in research in the fields of SLAM and 3D surface reconstruction over the last decade. They combine an RGB color camera with a depth camera that additionally provides dense per-pixel depth measurements of the observed 3D scene geometry. This simplifies many tasks in the application fields mentioned above. This dissertation fundamentally evolves around dense 3D reconstruction from consumer-grade RGB-D cameras. We will therefore explain the technical working principles of these range sensors more in-depth in the following.

RGB-D Cameras An RGB-D sensor produces frames at real-time rates of 30 Hz or more. A single RGB-D frame consists of a dense depth map $\mathcal{Z} : \Omega_{\mathcal{Z}} \to \mathbb{R}$ with usually 640×480 pixels and an associated three-channel color image $\mathcal{C} : \Omega_{\mathcal{C}} \to \mathbb{R}^3$ of 640×480 pixels or higher. Both $\mathcal{Z}$ and $\mathcal{C}$ within a frame are taken (almost) simultaneously in time, we neglect the small time offsets due to missing hardware synchronization. The estimated depth maps have metric scale and contain acceptably accurate per-pixel distance values for each color image pixel.

Color and depth images are captured by separate lenses at different spatial locations, which are both described using the pinhole camera model. To align color and depth for direct pixel correspondences, their calibrated relative pose is used to warp the depth map to the color image. We rely on the usually slightly inaccurate factory calibration, commonly leading to small RGB-D misalignments.

Passive vs. Active *Passive depth sensing* techniques are based on stereo systems and perform well in highly textured scene parts. When enough color and intensity features are present, detected correspondences allow to reliably estimate depth values, even in challenging outdoor scenarios. However, unsuccessful correspondence search in featureless regions makes the depth estimation fail.

In contrast, modern low-cost commodity depth sensors *actively* illuminate the scene with infrared IR light. This successfully addresses the challenging problem of reconstructing even uniformly colored regions and allows to obtain more complete depth measurements for indoor environments. Please note that both passive and active methods generally capture more accurate depth measurements for close range, while the depth uncertainty increases for larger distance. In the following, we will discuss the two main technical working principles for active depth sensing, namely *Structured Light* and *Time-of-Flight (ToF)* sensors.

Structured Light Cheap range sensors based on the Structured Light principle have been omnipresent in research since the release of the Microsoft Kinect V1 in 2010. Throughout this thesis, we mainly used the similar Asus Xtion Pro Live and the Occipital Structure Sensor.

In Structured Light, an IR projector projects a known unique structured pattern onto the scene. These additional artificial features significantly alleviate correspondence matching and remove the dependency on natural texture features. The detected correspondences between the projected pattern and the deformed observed IR pattern are then used to compute depth values using triangulation.

Structured Light sensors can only be applied in indoor scenarios, as the IR radiation in strong sunlight oversaturates the IR camera. This renders the projected pattern imperceivable and leads to missing depth estimates. Additionally, thin structures smaller than the pattern resolution, partial occlusions close to object boundaries as well as dark or reflective surfaces are problematic. Nonetheless, the ability to robustly estimate dense depth maps in real-time in almost arbitrary indoor scenes has largely alleviated these limitations. Within the working range ($\sim 0.3 - 5.0m$), the depth error increases quadratically with the distance. A detailed analysis of the noise characteristics is provided in [59].

Time-of-Flight ToF sensors such as the Microsoft Kinect V2 were initially less common than Structured Light devices, but have recently received more attention. *Pulsed ToF* cameras emit an IR light pulse and measure the travel time to the object and back to the sensor using rapid shutters and a high-precision clock. With knowledge of the constant speed of light, we can compute the distance as the half of the measured round trip distance. *Modulated ToF* devices emit a time-modulated IR signal and employ a phase detector to measure the phase shift of the returning light pulse. The distance to the scene is obtained from the phase shift and the known modulation frequency.

As in Structured Light, the IR camera in ToF devices is heavily influenced by sun radiation and by dark materials that do not reflect light. Furthermore, spurious measurements ("flying pixels") occurring at discontinuities deteriorate the depth map quality.

Depth Map Processing While the depth maps acquired from RGB-D sensors with the working principles above are sufficiently accurate, they exhibit strong noise characteristics and lack fine-scale geometric features. Firstly, a *bilateral filter* [114] smoothes the noisy depth maps, with a special treatment of depth discontinuities. Secondly, we compute a *vertex map* by back-projecting the per-pixel depth values using Equation (3.24) of the pinhole camera model. A *normal map* is obtained from the vertex map using finite differences. *Per-pixel weights* encode the uncertainty of depth measurements, in particular the frontoparallelity of observations and their inverse squared depth are considered.

3.3 Camera Motion Estimation

Online and offline dense 3D reconstruction methods require accurate camera motion estimates as input to generate a high-quality 3D model. To track the 6-DoF camera poses, RGB-D SLAM systems employ input frames to simultaneously build and update a metric 3D map of an environment and localize relative to it. *RGB-D odometry* robustly estimates the absolute camera pose of the current frame relative to the previous (key)frame or relative to a fused model. As these camera tracking strategies inevitably accumulate drift over time, a *global map optimization* achieves global consistency when *loop closures* are detected.

In the following, we exemplarily present the core components of the DVO-SLAM system [57], since it is tightly coupled with Chapter 6. It combines robust camera tracking with a global pose graph optimization and reconstructs a highly accurate camera motion in real-time on a CPU. For the rest of this section, we assume that color and depth in a frame have the same image resolution and are pre-aligned.

3.3.1 Dense RGB-D Odometry

The robust dense visual odometry approach used in [57] robustly estimates the relative camera motion $\mathcal{T} \in \mathbb{SE}(3)$ between two input RGB-D frames $(\mathcal{I}_1, \mathcal{Z}_1)$ and $(\mathcal{I}_2, \mathcal{Z}_2)$ by minimizing the photometric and geometric error of all pixels. Pixel correspondences between the frames are found dynamically through projective data association. We first define a function $g(\boldsymbol{\xi}, \boldsymbol{p}) = \boldsymbol{R}\boldsymbol{p} + \boldsymbol{t}$ that transforms a 3D point $\boldsymbol{p}$ using its pose $\mathcal{T} = (\boldsymbol{R}, \boldsymbol{t}) = \exp(\boldsymbol{\xi})$ into another frame. The warp function $\tau(\boldsymbol{x}, \boldsymbol{\xi}) : \mathbb{R}^2 \times \mathbb{R}^6 \to \mathbb{R}^2$ back-projects a pixel $\boldsymbol{x}$ in the original frame to 3D using its associated depth and projects it, after transforming, onto the 2D image plane in the second frame:

$$\tau(\boldsymbol{x}, \boldsymbol{\xi}) = \pi(g(\pi^{-1}(\boldsymbol{x}, \mathcal{Z}_1(\boldsymbol{x})), \boldsymbol{\xi})) \tag{3.39}$$

The *photometric error* measures the intensity differences between corresponding pixels and avoids tracking failure in planar but textured scenes:

$$\boldsymbol{r}_{\mathcal{I}}(\boldsymbol{\xi}) = \mathcal{I}_2(\tau(\boldsymbol{x}, \boldsymbol{\xi})) - \mathcal{I}_1(\boldsymbol{x}) \tag{3.40}$$

The *geometric error* ensures robust tracking even in textureless scenes, as long as geometric features are sufficiently present:

$$\boldsymbol{r}_{\mathcal{Z}}(\boldsymbol{\xi}) = \mathcal{Z}_2(\tau(\boldsymbol{x}, \boldsymbol{\xi})) - \left[g(\pi^{-1}(\boldsymbol{x}, \mathcal{Z}_1(\boldsymbol{x})), \boldsymbol{\xi})\right]_z, \tag{3.41}$$

where $[\cdot]_z$ extracts the z-coordinate of a 3D point. For each pixel, we combine photometric and geometric error in a stacked residual $\boldsymbol{r}_i(\boldsymbol{\xi}) = (\boldsymbol{r}_{\mathcal{I},i}(\boldsymbol{\xi}), \boldsymbol{r}_{\mathcal{Z},i}(\boldsymbol{\xi}))^\top$. Large outlier residuals are mitigated by integrating a robust loss function $\rho : \mathbb{R} \to \mathbb{R}$, for example based on Huber weights or t-distribution, resulting in the final optimization problem:

$$\hat{\boldsymbol{\xi}} = \arg\min_{\boldsymbol{\xi}} \sum_{i}^{n} \rho(\boldsymbol{r}_i(\boldsymbol{\xi})). \tag{3.42}$$

We minimize this energy with the Iteratively Reweighted Least-Squares algorithm, where the per-residual weights are re-computed after each iteration. To improve convergence, the optimization is embedded in a coarse-to-fine pyramid scheme. As a side-product, we obtain an estimate for the pose's covariance $\Sigma = (\boldsymbol{J}_{\boldsymbol{r}}^\top \boldsymbol{W} \boldsymbol{J}_{\boldsymbol{r}})^{-1} \in \mathbb{R}^{6\times 6}$, with the matrix $\boldsymbol{W}$ containing robust per-residual weights.

To regularize the effect of noisy depth maps and limit odometry drift, we perform camera tracking against the previous keyframe with incrementally fused depth maps. A new keyframe is created whenever (1) alignment of a new frame against the previous keyframe fails, (2) a specified pose distance w.r.t. the previous keyframe is exceeded, or (3) based on a differential entropy criterion of the motion estimate.

3.3.2 Global Map Optimization

Incremental frame-to-(key)frame or frame-to-model tracking methods unavoidably accumulate drift over time in practice. Online SLAM systems overcome this general problem by performing a global camera pose optimization in a continuous background process. Whenever loop closures are detected, i.e. already visited places are re-visited, the overall camera trajectory error is corrected to maximize global consistency.

Map Representation The two most common methods for global map optimization in SLAM system are *Bundle Adjustment (BA)* [115] and *Pose Graph Optimization (PGO)*.

The map in BA consists of absolute camera poses and a large number of 3D landmarks; the landmarks are observed as repeatable sparse feature points in the input color images. These feature observations are then used as constraints in the global BA problem to jointly optimize for landmark positions and poses by minimizing the reprojection error. Since BA scales poorly with increasing map sizes, it quickly becomes too computationally expensive for real-time reconstruction of large-scale environments, especially on CPUs. Only few well-engineered RGB-D SLAM systems [28, 82] manage to achieve real-time performance, hence global BA is commonly used rather for offline post-processing.

In contrast, the graph-based map representation of DVO-SLAM contains solely absolute keyfames poses as nodes/vertices and relative pose-pose constraints between the keyframes as edges. With this lean pose graph, even large-scale maps with thousands of nodes can be optimized in real-time using a generic graph optimization framework like *g2o* [63].

Loop Closures The relative pose-pose constraints in the pose graph stem from incremental frame-to-frame odometry and from detected loop closures of previously visited keyframes. To find loop closures, we first search for potential loop closure candidates for the current keyframe. Firstly, a nearest neighbor search returns spatially close keyframes within a sphere of predefined radius. Secondly, a place recognition method based on bags-of-words uses extracted visual feature descriptors to retrieve visually similar previous keyframe candidates from a database built online. In a final post-processing step, an extensive brute-force matching between all keyframes yields additional pose-pose constraints. For every loop closure candidate, we perform a first dense alignment of the candidate with the current keyframe on a coarse pyramid level. With the estimated relative pose and covariance matrix, we discard false candidates with the entropy ratio test also employed for keyframe selection. To incorporate valid loop closures into the map, we compute a refined pose and covariance estimate over all pyramid levels and add a new pose-pose constraint as edge into the pose graph.

Pose Graph Optimization In practice, the vertices $\mathcal{V} = \{\mathcal{K}_i\}$ of a pose graph are the absolute poses $\mathcal{T}_i$ of the keyframes $\mathcal{K}_i = (\mathcal{C}_i, \mathcal{I}_i, \mathcal{Z}_i, \mathcal{T}_i)$, transforming from frame i to the global world coordinates frame. The graph edges are pose-pose constraints $\mathcal{E} = \{\mathcal{E}_{ij}\}$ between keyframes, where an edge $\mathcal{E}_{ij} = (\mathcal{T}_{ij}, \Sigma_{ij})$ includes the relative pose from frame j to frame i along with its estimated covariance matrix. To obtain optimal absolute poses $\mathcal{T}_i$, we formulate the energy of the Pose Graph Optimization as a NLS problem as follows:

$$E_{\text{PG}} = \sum_{(\mathcal{T}_{ij}, \Sigma_{ij}) \in \mathcal{E}} \log(\mathcal{T}_{ij} \mathcal{T}_j^{-1} \mathcal{T}_i)^\top \Sigma_{ij}^{-1} \log(\mathcal{T}_{ij} \mathcal{T}_j^{-1} \mathcal{T}_i), \tag{3.43}$$

where the residuals measure the relative pose error in the tangent space of $\mathbb{SE}(3)$. The optimization distributes the overall error over the edges of the pose graph. Hereby, the covariance matrix of the relative motion serves as residual weight and gives less weight to uncertain relative pose estimates.

DVO-SLAM continuously runs the PGO in the background, leading to improved global consistency as loop closures are added on-the-fly. The final corrected camera trajectory is highly accurate and allows to generate a dense 3D model of the scene with compelling quality.

3.4 Dense 3D Surface Representation

The dense RGB-D input data of a static scene and their estimated camera poses allow to reconstruct a complete global 3D model. To fuse noisy depth maps from varying viewpoints, diverse dense 3D surface representations have been developed. In this thesis, we focus on dense *Signed Distance Functions (SDFs)* [26, 84] that have proved particularly successful in combination with commodity RGB-D sensors.

3.4.1 Signed Distance Functions

Implicit surface representations based on SDFs are usually implemented as discrete 3D voxel grids (*Signed Distance Fields*) in practice.[1] Voxels $\boldsymbol{v} \in \Lambda \subset \mathbb{R}^3$ are defined in a discrete voxel space domain Λ of a 3D volume. To access locations between discrete voxel positions, tri-linear interpolation is used.

The Signed Distance Function $\mathbf{D} : \mathbb{R}^3 \to \mathbb{R}$ stores the signed distance of a voxel $\boldsymbol{v}$ to its closest surface. Points $\boldsymbol{v} \in \Lambda^-$ in occupied space (i.e. behind the object surface) have a negative SDF with $\mathbf{D}(\boldsymbol{v}) < 0$, while $\mathbf{D}(\boldsymbol{v}) > 0$ holds for points $\boldsymbol{v} \in \Lambda^+$ in free space outside. The reconstructed iso-surface between Λ^- and Λ^+ is hence defined as the zero level set of the SDF where $\mathbf{D}(\boldsymbol{v}) = 0$. In addition to the signed distance $\mathbf{D}$, each voxel commonly stores an integration weight $\mathbf{W} : \mathbb{R}^3 \to \mathbb{R}$ and its fused RGB color $\mathbf{C} : \mathbb{R}^3 \to \mathbb{R}^3$.

Truncated Signed Distance Function (TSDF) In a TSDF[2], only voxels with $\mathbf{D}(\boldsymbol{v})$ within a given truncation threshold μ around the actual surface are updated with each frame. This truncation effectively regularizes noise in the fused depth by averaging locally, but it also leads to smoothed out surface details that are smaller than μ.

Since operations on the TSDF volume are well-parallelizable over its voxels, real-time performance is easily achieved using GPGPU programming with graphics cards. However, scarce GPU memory limits both spatial extents and resolution of dense voxel grids, since voxels are also allocated for free space. To address this problem, Voxel Hashing is introduced as a sparse volumetric surface representation in Section 3.4.4.

3.4.2 Surface Fusion

By integrating incoming partial RGB-D frames into a single TSDF volume, we ultimately obtain a more complete 3D reconstruction. Multiple (temporally consecutive) depth

[1] We treat the terms *Signed Distance Function* and *Signed Distance Field*, both abbreviated as SDF, equivalently without any further distinction in this work.

[2] The term SDF generally refers to *Truncated SDFs* throughout this thesis.

samples are averaged per voxel using a running weighted average update scheme in order to effectively regularize noisy depth measurements. For an input RGB-D frame with depth map $\mathcal{Z}_k$, color image $\mathcal{C}_k$ and absolute pose $\mathcal{T}_k \in \mathbb{SE}(3)$ that transforms from world coordinates to the local camera frame, we can update all voxels of the 3D voxel grid independently in parallel.

Computing the TSDF We first transform each voxel $\boldsymbol{v} \in \Lambda$ into the input camera coordinate system using $\mathcal{T}_k$ and also extract its z-coordinate in the local frame as $\bar{z}_k(\boldsymbol{v})$:

$$\boldsymbol{v}_k = \mathcal{T}_k \boldsymbol{v}, \qquad \bar{z}_k(\boldsymbol{v}) = [\boldsymbol{v}_k]_z. \tag{3.44}$$

Next, we project the local voxel coordinate $\boldsymbol{v}_k$ onto the image plane, sample the input depth and color measurements at the respective 2D pixel location $\boldsymbol{x}_k = \pi(\boldsymbol{v}_k)$ and compute an additional integration weight:

$$c_k(\boldsymbol{v}) = \mathcal{C}_k(\boldsymbol{x}_k), \qquad z_k(\boldsymbol{v}) = \mathcal{Z}_k(\boldsymbol{x}_k), \qquad w_k(\boldsymbol{v}) = \frac{\cos(\theta)}{z_k(\boldsymbol{v})^2}. \tag{3.45}$$

The integration weight $w_k(\boldsymbol{v})$ aims to approximate the uncertainty of the sample, where θ is the angle between the viewing direction and the normal (computed from the depth map). The measure $\cos(\theta)$ favors frontoparalellity, while $1/z_k(\boldsymbol{v})^2$ quadratically penalizes increasing distance of the surface to the sensor. Please note that there are different weighting schemes, e.g. using a uniform $w_k = 1$ would instead lead to a simple moving average update scheme.

To obtain the TSDF value $d_k(\boldsymbol{v})$, we compute the projective signed distance $\bar{z}_k(\boldsymbol{v}) - z_k(\boldsymbol{v})$ as a practical approximation of the true signed distance and truncate it using the truncation function $\Psi : \mathbb{R} \to \mathbb{R}$:

$$d_k(\boldsymbol{v}) = \Psi(\bar{z}_k(\boldsymbol{v}) - z_k(\boldsymbol{v})). \tag{3.46}$$

Ψ truncates the absolute SDF value to a specified maximum threshold μ, which represents a regularizing narrow band around the iso-surface:

$$\Psi(d) = \begin{cases} \min(1, \frac{d}{\mu}) & \text{if } d \geq -\mu, \\ 0 & \text{otherwise.} \end{cases} \tag{3.47}$$

Frame Integration With the sampled color $c_k(\boldsymbol{v})$, the TSDF value $d_k(\boldsymbol{v})$ and its integration weight $w_k(\boldsymbol{v})$ derived from input frame k, we update the per-voxel attributes stored in the TSDF voxel grid. When all M input RGB-D frames are given in advance,

the final per-voxel data is obtained using a weighted average of all observations:

$$\mathbf{D}(\boldsymbol{v}) = \frac{\sum_{k=1}^{M} w_k(\boldsymbol{v}) d_k(\boldsymbol{v})}{\mathbf{W}_k(\boldsymbol{v})} \tag{3.48}$$

$$\mathbf{C}(\boldsymbol{v}) = \frac{\sum_{k=1}^{M} w_k(\boldsymbol{v}) c_k(\boldsymbol{v})}{\mathbf{W}_k(\boldsymbol{v})} \tag{3.49}$$

$$\mathbf{W}(\boldsymbol{v}) = \sum_{k=1}^{M} w_k(\boldsymbol{v}). \tag{3.50}$$

However, in real-time RGB-D scanning scenarios the input frames are continuously acquired from the RGB-D sensor and immediately incorporated into the 3D model. In this case, the TSDF volume is incrementally updated with each frame:

$$\mathbf{D}_k(\boldsymbol{v}) = \frac{\mathbf{D}_{k-1}(\boldsymbol{v})\mathbf{W}_{k-1}(\boldsymbol{v}) + d_k(\boldsymbol{v}) w_k(\boldsymbol{v})}{\mathbf{W}_{k-1}(\boldsymbol{v}) + w_k(\boldsymbol{v})} \tag{3.51}$$

$$\mathbf{C}_k(\boldsymbol{v}) = \frac{\mathbf{C}_{k-1}(\boldsymbol{v})\mathbf{W}_{k-1}(\boldsymbol{v}) + c_k(\boldsymbol{v}) w_k(\boldsymbol{v})}{\mathbf{W}_{k-1}(\boldsymbol{v}) + w_k(\boldsymbol{v})} \tag{3.52}$$

$$\mathbf{W}_k(\boldsymbol{v}) = \mathbf{W}_{k-1}(\boldsymbol{v}) + w_k(\boldsymbol{v}). \tag{3.53}$$

Please note that this static surface integration method does not explicitly deal with dynamic scenes. Nevertheless, by saturating the accumulated integration weight $\mathbf{W}(\boldsymbol{v})$ to a maximum threshold after each update, we prioritize more recent measurements and can, to some degree, react to dynamic changes in the scene. But even without thresholding $\mathbf{W}(\boldsymbol{v})$, inconsistent observations are commonly averaged out over time by the TSDF fusion process, making it robust to small changes of the environment.

Frame de-integration When an input RGB-D frame k with $(\mathcal{C}_k, \mathcal{Z}_k, \mathcal{T}_k)$ is kept available during the 3D reconstruction process, we can also remove it again from the SDF volume to reverse the integration procedure. We therefore sample and compute $c_k(\boldsymbol{v})$, $d_k(\boldsymbol{v})$ and $w_k(\boldsymbol{v})$ analogous to the surface update steps and de-integrate color, TSDF and weight of the respective voxel in the volume:

$$\mathbf{D}_{k-1}(\boldsymbol{v}) = \frac{\mathbf{D}_k(\boldsymbol{v})\mathbf{W}_k(\boldsymbol{v}) - d_k(\boldsymbol{v}) w_k(\boldsymbol{v})}{\mathbf{W}_k(\boldsymbol{v}) - w_k(\boldsymbol{v})} \tag{3.54}$$

$$\mathbf{C}_{k-1}(\boldsymbol{v}) = \frac{\mathbf{C}_k(\boldsymbol{v})\mathbf{W}_k(\boldsymbol{v}) - c_k(\boldsymbol{v}) w_k(\boldsymbol{v})}{\mathbf{W}_k(\boldsymbol{v}) - w_k(\boldsymbol{v})} \tag{3.55}$$

$$\mathbf{W}_{k-1}(\boldsymbol{v}) = \mathbf{W}_k(\boldsymbol{v}) - w_k(\boldsymbol{v}). \tag{3.56}$$

Frame de-integration and integration are combined in Chapter 6 to correct the reconstructed surface in a voxel-hashed TSDF on-the-fly.

3.4.3 Surface Extraction

While TSDFs allow to conveniently fuse and update the surface, we require additional methods to visualize and extract the underlying geometry that is only encoded implicitly. To achieve this, *raycasting* enables visual previews of the TSDF online during runtime. The Marching Cubes algorithm [75] extracts an explicit geometry representation in the form of a *3D triangle mesh*, which can be analyzed further e.g. for path planning or collision detection.

Raycasting To generate synthetic views of the surface stored in the TSDF, a ray is shot through each pixel of a virtual camera, with known pose and intrinsics, into the voxel grid. We traverse the ray through the volume and determine all intersecting voxels using the Digital Differential Analyzer (DDA) algorithm [11]. When a first zero-crossing of the iso-surface is found along the ray, we calculate a refined depth value and store it at the respective pixel of the virtual depth map. Besides the depth, we also render a synthetic normal map, with the normals computed directly from the underlying TSDF using finite differences, as well as a synthetic color image with the interpolated voxel colors at the intersection point. The predicted depth map can be used for frame-to-model camera tracking against the fused 3D model, e.g. using the projective ICP algorithm [84].

Marching Cubes Besides generating virtual views of the implicit iso-surface in the TSDF, we can also extract an explicit 3D geometry representation embedded in the zero level set using the *Marching Cubes* algorithm [75]. The generated 3D triangle mesh $\mathcal{M} = (\mathcal{V}, \mathcal{F})$, with vertices $\mathcal{V}$ and faces/triangles $\mathcal{F}$, is directly usable in the standard 3D rendering pipeline of graphics cards. In the algorithm, we compute for each voxel cube in the grid an index that represents one of 256 possible configurations, based on the contained zero-crossing of the iso-surface. The corresponding configuration generates polygons to represent the internal iso-surface; the polygon vertex positions are linearly interpolated according to the neighboring TSDF values. The final complete mesh is then combined from the polygons of the individual voxel cubes.

Texture Mapping The 3D triangle mesh generated by Marching Cubes accurately represents the reconstructed geometry, but often consists of millions of vertices and faces even for smaller objects. As rendering in Computer Graphics becomes inefficient with such overly complex geometry, we approach this shortcoming through *texture mapping*. This method achieves high visual model appearance by mapping high-quality textures to the faces of a simplified 3D mesh with low geometric complexity. Texture mapping represents a transformation from 3D mesh vertices $v = (x, y, z)^\top \in \mathcal{V}$ in object space to texture coordinates $t = (u, v)^\top \in \Omega_\mathcal{T}$ on a 2D texture image $\mathcal{T} : \Omega_\mathcal{T} \rightarrow \mathbb{R}^3$. This

parametrization allows to store multiple texels for a single triangle/fragment (defined by the uv-coordinates of its vertices), where each texel $t \in \Omega_{\mathcal{T}}$ stores an RGB color value. The graphics cards hardware rasterizes the triangles and performs per-fragment color lookups by interpolating texture coordinates from adjacent vertices.

The texture mapping pipeline in Chapter 4 is based on [67] and applicable for meshes with arbitrary topology. First, the mesh geometry is decimated by reducing the number of faces and vertices. This simplified mesh contains larger triangles that are well-suited for texturing, while still preserving the original shape. To better deal with non-trivial mesh topologies, the mesh is then segmented into planar charts. Afterwards, the individual charts are uv-unwrapped, i.e. the surface for each chart is projected onto a 2D texture by assigning 2D uv-coordinates to 3D vertex coordinates. We compute this texture parametrization for example using Least Squares Conformal Maps [67], which minimizes the angle deformations of the mesh triangles during the projection. The generated 2D sub-textures are efficiently packed onto a single large global texture map that contains the surface textures for the entire mesh.

There is a unique mapping from 2D texels to the corresponding 3D world coordinates and vice versa. This *barycentric mapping* $\psi : \Omega_{\mathcal{T}} \rightarrow \mathbb{R}^3$ computes for a texel $t \in \Omega_{\mathcal{T}}$ a 3D position $v_t = \psi(t) \in \mathbb{R}^3$ from its surrounding face vertices using barycentric interpolation. This allows to sample the generated 3D surface points and consequently re-compute per-texel colors. In addition to texture maps, various other surface properties can be stored in maps with the same uv-coordinates; normal maps for example contain per-texel normals that allow to store highly resolved geometric details.

3.4.4 Voxel Hashing

The regular uniform voxel grids introduced in Section 3.4.1 are characterized by their inefficient memory utilization because they also allocate voxels for empty space far from the actual surface. Since the bounded size of GPU memory restricts the reconstruction of large-scale scenes, more memory-efficient hierarchical or sparse TSDF representations have been developed to remedy this problem.

Voxel Hashing [86] is a sparse SDF representation that is substantial for the work presented in Chapter 6. It encodes the TSDF in a sparse fashion around the actual surface in order to support larger spatial extents. We explain this 3D representation in more details in the following.

3.4.4.1 Data Structure

To avoid unnecessary memory allocations for empty space and enable virtually infinite scene sizes, Voxel Hashing employs a two-level data structure. Smaller voxel blocks of a predefined size are allocated only close to the surface and store the signed distance values. These blocks are managed in a hash table and efficiently addressed using a spatial hash function. The data structure is optimized towards GPUs, providing efficient data fusion and access with a runtime complexity of both $\mathcal{O}(1)$.

Voxel Blocks Conceptually, the whole scene is partitioned into small fixed-size voxel blocks that altogether constitute an infinite uniform grid. However, we exploit sparsity by only instantiating voxel blocks within a truncation region around the reconstructed surface geometry. These subvolumes are dynamically allocated and deallocated to store the geometry only where surface measurements are observed. Each block by itself is a regular voxel grid composed of $8\times8\times8$ voxels, where each voxel stores a TSDF value, color and weight. Implementation-wise, a voxel block array stores the blocks sequentially in memory, managed by an additional array for maintaining all (un-)allocated voxel blocks and a corresponding index variable for the rearmost array element.

Hash Map The voxel blocks are then managed in a hash table on the GPU with $c \in \mathbb{N}$ consecutive hash buckets through a spatial hash function that maps from the 3D world coordinates of a voxel block to its corresponding hash bucket with $m \in \mathbb{N}$ entries each. Therefore, the 3D position of a voxel block is first converted to a 3D integer index $(i, j, k)^\top \in \mathbb{Z}^3$ using the voxel size and block dimensions. The spatial hash function presented in [113] then maps these integer coordinates to a linearized hash value:

$$H(i, j, k) = (i \cdot p_1 \oplus j \cdot p_2 \oplus k \cdot p_3) \bmod c, \tag{3.57}$$

where $\oplus$ is the XOR operator and $p_1 = 73856093, p_2 = 19349669, p_3 = 83492791$ are large prime numbers. In addition to the m entries per bucket, a linked list is appended to a bucket in the case of an overflow. Each entry contains the 3D position (i, j, k) to uniquely identify blocks, a pointer to its allocated block and an offset to the next linked list member to resolve collisions.

The proposed data structure allows to efficiently *retrieve*, *insert* or *delete* voxel blocks and resolve hash collisions. All operations commonly find the hash entry for a specific voxel block by first calculating its spatial hash, then iterating over the entries in the respective bucket and finally traversing the appended linked list if necessary. We refer the reader to the original paper [86] for more in-depth details.

3.4.4.2 Integration

We require an extended surface fusion procedure in order to integrate new RGB-D frames into the voxel-hashed TSDF volume. Before fusing a frame, voxel blocks need to be streamed from GPU to host and vice versa in advance, which will be discussed in Section 3.4.4.3.

Surface Allocation As the fused surface model is extended dynamically, we first need to allocate non-existing voxel blocks that have surface measurements in the RGB-D frame. We therefore instantiate rays from the frame's camera center through all pixels and use the DDA algorithm to determine all intersected voxel blocks. We only consider blocks along the ray that are within the TSDF truncation interval of the corresponding depth sample. All traversed voxel blocks are allocated and inserted into the hash map, if not already allocated.

Surface Update To fuse the surface, we select voxel blocks within the current camera frustrum and then update the voxels with the observed surface information. This is done by iterating over all blocks and checking whether their centers approximately project into the (slightly extended) camera frustrum. Each voxel in the selected voxel blocks that is within the truncation region is then updated analogous to the standard fusion procedure explained in Section 3.4.2.

Garbage Collection Finally, a garbage collection step removes voxel blocks that were unnecessarily allocated due to erroneous, unreliable or noisy depth measurements. Ideally, the surface fusion smoothes out the surface information stored in these blocks over time. Obsolete blocks are detected from the maximum per-voxel weight or minimum TSDF within each block and finally removed from the Voxel Hashing data structure.

3.4.4.3 Streaming

The scarce amount of available GPU memory poses a practical limitation on the size and resolution of the reconstructed scene. To enable large-scale surface reconstruction at efficient real-time runtimes, we keep only voxel blocks that approximately reside within the current camera frustum in GPU memory. This frees up space in GPU memory to allocate new voxel blocks and map even large-scale environments. All inactive voxel blocks are streamed out-of-core to the host memory and swapped in when they are required again on the GPU. Instead of also using a hash map in the host memory, the world space is partitioned into *chunks* of uniform size. Each chunk contains a linked list with the streamed out voxel blocks and a linked list with the voxel block descriptors.

GPU-to-Host Streaming After estimating the camera pose for a new frame, blocks outside the active region are selected for streaming by checking whether they reside within a sphere of constant radius around the camera center (plus some constant offset). All selected blocks are condensed for efficiency, streamed from GPU to host and deleted from the hash map on the GPU. The voxel blocks on the host are then referenced in the respective chunks they lie in.

Host-to-GPU Streaming If a chunk falls entirely into the sphere around the camera center again, the whole chunk with all its voxel blocks is streamed to the GPU again all at once. To ensure constant time, only the chunk closest to the sphere center is streamed in. This is implemented by manually increasing the index pointer of the voxel block array for all blocks of the chunk and then uploading the entire chunk to intermediate GPU buffers. All uploaded blocks are finally inserted into the hash map using a modified insertion algorithm without block allocation steps.

Synchronization of Voxel Block Allocation As duplicate hash map entries are problematic for the continuous host-to-GPU streaming, a binary array on the GPU encodes which chunks are currently stored on the host. This array is then checked when a new frame is about to be integrated, in order to prohibit that a voxel block is allocated in case is already in a chunk on the host.

3.5 Shading-based Shape Refinement

The noisy depth maps captured from commodity RGB-D sensors and the over-smoothed 3D surface geometry encoded in a fused SDF characteristically lack fine-scale surface details. *Shape-from-Shading (SfS)* techniques provide an elegant solution to recover high-quality geometry using the color images in RGB-D frames, which usually have a higher resolution than the associated depth maps. The preliminaries presented in this section describe and complement the fundamental principles for the work in Chapter 5.

3.5.1 Shading Model

The *shading equation* (or *rendering equation*) describes the physical light transport model and relates the *reflected irradiance* of a surface with its geometry, material properties and incident illumination (Figure 3.1). For each 3D point $\boldsymbol{p}$ on the surface $\mathcal{S}$, the following model computes the reflected irradiance $\mathbf{B}$ as a function of surface normal $\mathbf{n}$, material albedo $\mathbf{a}$ and lighting coefficients ℓ_m:

$$\mathbf{B}(\boldsymbol{p}) = \mathbf{a}(\boldsymbol{p}) \sum_{m=1}^{b^2} \ell_m H_m(\mathbf{n}(\boldsymbol{p})), \tag{3.58}$$

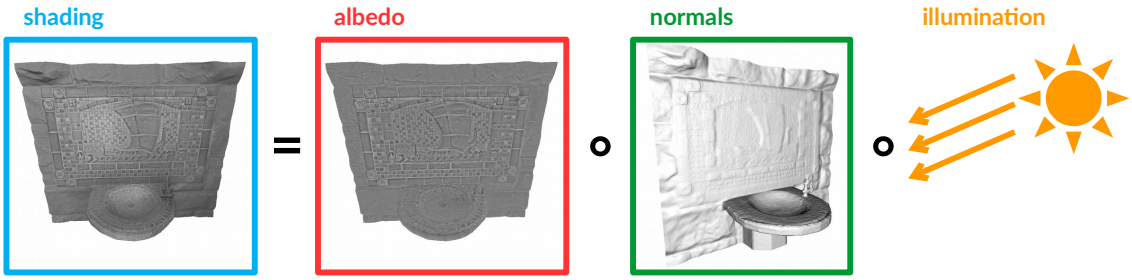


Figure 3.1: The shading model describes the physical light transport. It relates the emitted shading/irradiance (blue) with the underlying material properties (red), normals/geometry (green) and incident illumination (yellow).

where the emitted shading $\mathbf{B}$ is ultimately observable as intensities in input images. The global scene lighting is parametrized with *Spherical Harmonics (SH)* basis functions H_m with lighting coefficients ℓ_m (Section 3.5.4) that illuminate a surface with normals $\mathbf{n}(\boldsymbol{p})$; the spatially-varying surface albedo parameters $\mathbf{a}(\boldsymbol{p})$ describe the intrinsic color of the material. While the points $\boldsymbol{p} \in \mathcal{S}$ are independent of a specific surface representation, the underlying geometry is usually represented as a depth map or as a TSDF volume in practice. More expressive material reflectance models exist, e.g. *Bidirectional Reflectance Distribution Function (BRDF)*, but it is highly challenging to estimate their parameters in unconstrained real-world settings.

3.5.2 Shape-from-Shading

The challenging inverse problem of Shape-from-Shading tries to infer 3D shape from a single image (or multiple images) of a scene. The underlying intuition is that fine geometric features under illumination inherently result in shading cues in an image observing the scene. By inverting the image formation model, these cues can then be used to estimate 3D geometry and surface properties. A physical model of the light transport process, e.g. the shading model in Equation (3.58), is utilized to estimate illumination, intrinsic material properties (e.g. albedo) and surface geometry of a scene.

However, this problem is highly ill-posed and challenging due to its considerable ambiguity in uncontrolled environments with usually unknown lighting and material reflectance. To tackle the problem effectively, strong assumptions on both scene (e.g. Lambertian surface reflectance) as well as illumination are usually imposed on the scene. For more details on SfS, we recommend the interested reader to investigate the extensive survey presented in [131].

3.5.3 Shading-based Refinement

Shading-based Refinement (SBR) describes the method of applying SfS techniques to an initial surface geometry estimate, with the goal of recovering fine-scale geometry using the higher resolution color images. Generally, the coarse shape obtained from RGB-D data, i.e. either the captured depth maps or fused coarse SDFs, can be utilized as initial shape prior for both surface geometry and normals. This prior guides the SfS process and allows to resolve ambiguities among lighting, surface geometry and albedo. In particular, it is possible to overcome the well-known *Bas-Relief ambiguity* [17] and even deal with unknown material reflectance variations and uncalibrated lighting.

SBR approaches have converged to an often similar pipeline in recent years. Therefore, an inverse rendering optimization is usually coupled with an effective parameterization of a shading model similar to Section 3.5.1:

1. The refined surface is initialized with the coarse surface estimate obtained from RGB-D data and the albedo is set to a constant uniform value.
2. We estimate the scene lighting, given the current estimates of surface geometry and material albedo (Section 3.5.4).
3. We optimize the surface geometry and material albedo, given the estimated lighting (Section 3.5.5).

To achieve convergence, steps 2 and 3 are iterated and often embedded in a coarse-to-fine pyramid scheme.

Assumptions While the use of depth maps is highly beneficial for SBR, we still need some (less severe) assumptions about scene and illumination to better constrain the challenging problem in practice. Firstly, we assume uncalibrated natural illumination. Secondly, we represent input luminance, light color, albedo and reflected irradiance (shading) with gray-scale intensity values instead of three RGB color channels for simplicity. Thirdly, a common and in practice very important assumption is that surfaces in the scene are Lambertian, i.e. the reflected irradiance is independent of the viewing direction and not capable of representing specularities.

3.5.4 Lighting Estimation using Spherical Harmonics

One of the most relevant steps in every SfS approach consists of estimating the scene lighting. The coarse surface normals, which can be computed from a given depth map obtained from RGB-D sensors, aid to approximate the global scene illumination and tackle this problem more robustly.

Spherical Harmonics While there are various light representations in Computer Graphics pipelines (e.g. point light sources, etc.), Spherical Harmonics (SH) [88] have proven particularly successful in inverse rendering problems. They provide a good approximation for Lambertian surfaces under natural illumination, where the incident irradiance for surface points is known to be smooth.

For a point $\boldsymbol{p} \in \mathcal{S}$ on the surface, the SH basis functions $H_m : \mathbb{R}^3 \to \mathbb{R}$ encode the incident scene lighting as a function of the unit surface normal $\mathbf{n} = \mathbf{n}_x, \mathbf{n}_y, \mathbf{n}_z^\top$. With only nine 2nd order SH basis functions and their respective lighting coefficients $\ell = (\ell_1, \dots, \ell_{b^2})^\top$ (with $b = 3$ bands), we obtain a faithful approximation for the irradiance of diffuse objects under low-frequency illumination:

$$
\begin{aligned}
&H_0(\mathbf{n}) = 1.0 \qquad H_1(\mathbf{n}) = \mathbf{n}_y \qquad H_2(\mathbf{n}) = \mathbf{n}_z \qquad H_3(\mathbf{n}) = \mathbf{n}_x \\
&H_4(\mathbf{n}) = \mathbf{n}_x\mathbf{n}_y \qquad H_5(\mathbf{n}) = \mathbf{n}_y\mathbf{n}_z \qquad H_6(\mathbf{n}) = -\mathbf{n}_x\mathbf{n}_x - \mathbf{n}_y\mathbf{n}_y + 2\mathbf{n}_z\mathbf{n}_z \\
&H_7(\mathbf{n}) = \mathbf{n}_z\mathbf{n}_x \qquad H_8(\mathbf{n}) = \mathbf{n}_x\mathbf{n}_x - \mathbf{n}_y\mathbf{n}_y
\end{aligned}
$$

Since the lights are assumed to be infinitesimally far away and thus are purely directional, the coefficients ℓ_m are spatially invariant. While SH lighting represents diffuse and ambient lighting, neither specularities nor occlusions are considered.

Lighting Estimation The task of estimating the scene illumination from an input intensity image becomes significantly more tractable with additional knowledge about the scene geometry. With the presence of RGB-D data, we take advantage of rough surface normals from the (smoothed) input depth maps or the fused SDF volume. We initially set the material albedo to a constant uniform value; it will however be adjusted as the geometry and albedo optimization advances (Section 3.5.5).

We compute the scene lighting coefficients ℓ_m by minimizing the difference between the forward shading $\mathbf{B}$ in Equation (3.58) and the observed input intensity $\mathbf{I}$ over all surface points $\boldsymbol{p} \in \mathcal{S}$:

$$
\begin{aligned}
E_{\text{shading}} &= \sum_{\boldsymbol{p} \in \mathcal{S}} \left\| \mathbf{B}(\boldsymbol{p}) - \mathbf{I}(\boldsymbol{p}) \right\|^2 && (3.59) \\
&= \sum_{\boldsymbol{p} \in \mathcal{S}} \left\| \mathbf{a}(\boldsymbol{p}) \sum\nolimits_{m=1}^{b^2} \ell_m H_m(\mathbf{n}(\boldsymbol{p})) - \mathbf{I}(\boldsymbol{p}) \right\|^2, && (3.60)
\end{aligned}
$$

with given surface normals $\mathbf{n}(\boldsymbol{p})$ and albedos $\mathbf{a}(\boldsymbol{p})$. We find the lighting coefficients ℓ_m that minimize the energy E_{shading} by solving an overdetermined linear least-squares problem. This makes the method robust towards high-frequency illumination and geometry changes, such that we obtain an accurate global lighting model.

3.5.5 Geometry and Albedo Refinement

Once the lighting coefficients are determined, we use shading cues from the observed input intensity images to perform the actual Shape-from-Shading. We optimize for both the surface geometry and dense spatially-varying material albedo, because otherwise albedo variations could be interpreted as geometry changes and consequently lead to texture-copy artifacts. Even with prior knowledge on the shape, the problem of separating albedo and shading is particularly hard for high-frequency textures.

Starting with the coarse shape estimate obtained from RGB-D data and a uniform albedo, we tackle the same inverse rendering problem specified in Equation (3.60). We solve for the refined geometry through its surface normal $\mathbf{n}(\boldsymbol{p})$ and for the albedo $\mathbf{a}(\boldsymbol{p})$ by again minimizing the difference between estimated shading and its input luminance over all surface points. By formulating the surface normals as a function of the encoded surface geometry, we implicitly refine the underlying geometry.

The proposed least squares objective is highly non-linear and solved in an alternating or joint optimization. Since the specified SfS problem is highly ill-posed, additional regularization terms are required to reduce the effect of noise and to regularize the shading. This leads to the following overall minimization objective with multiple cost terms, similar to the formulation used in Chapter 5:

$$E_{\text{sfs}} = \sum_{\boldsymbol{p} \in \mathcal{S}} \lambda_{\text{shading}} E_{\text{shading}} + \lambda_v E_v + \lambda_s E_s + \lambda_a E_a, \tag{3.61}$$

with data term E_{shading}, geometric regularizers E_v and E_s, albedo regularizer E_a and their respective weighting hyperparameters λ_{shading}, λ_v, λ_s, λ_a.

Shading Constraint The shading constraint E_{shading} tries to maximize the consistency between the estimated forward shading of surface points and their sampled intensities in the input luminance images. To achieve this, geometry and albedo are gradually refined as the optimization progresses. While this data term follows the very same objective (Equation (3.60)) as the lighting estimation, a gradient-based shading constraint may be more robust in practice. Here, instead of directly comparing the forward shading with its input intensity, the gradients of the rendered shading are compared with the gradient of the respective intensities to improve robustness. The gradients are calculated using discrete (forward) differences from neighboring points on the surface.

Albedo Regularization To separate albedo from shading and to avoid overfitting or texture-copy artifacts, an albedo regularizer E_a is usually introduced to guide the albedo refinement. To effectively regularize the spatially-varying albedo, well-working priors

are often designed based on various heuristics and assumptions. Since the albedo is assumed to be piecewise smooth, a weighted Laplacian regularizer is often employed. Chromaticity changes, i.e. changes in colorfulness, often go along with changes of the intrinsic material. The weights of the anisotropic Laplacian regularizer can consequently be set based on the chromaticity differences of two neighboring surface points, or based on the spatial distance to geometry discontinuities.

Geometry Regularization As noise highly affects the robustness of SfS, we need regularization on the geometry to ensure a stable shape refinement. A *surface stabilization constraint* (or *depth constraint*) E_s enforces the refined geometry to stay close to the initial coarse reconstruction. This prior is also essential for resolving the related Bas-Relief ambiguity. A *volumetric regularizer* E_v enforces smoothness in the distance values between neighboring surface points, allowing only for subtle changes in the surface. This smoothness constraint is commonly implemented as an isotropic Laplacian regularizer.

Part II

Own Publications

Chapter 4

Super-Resolution Keyframe Fusion for 3D Modeling with High-Quality Textures

Authors:	Robert Maier[1]	robert.maier@tum.de
	Jörg Stückler[1]	joerg.stueckler@in.tum.de
	Daniel Cremers[1]	cremers@tum.de

[1] Technische Universität München, Germany

Conference: *IEEE International Conference on 3D Vision (3DV) 2015.*
DOI: 10.1109/3DV.2015.66.

Abstract We propose a novel fast and robust method for obtaining 3D models with high-quality appearance using commodity RGB-D sensors. Our method uses a direct keyframe-based SLAM frontend to consistently estimate the camera motion during the scan. The aligned images are fused into a volumetric truncated signed distance function representation, from which we extract a mesh. For obtaining a high-quality appearance model, we additionally deblur the low-resolution RGB-D frames using filtering techniques and fuse them into super-resolution keyframes. The meshes are textured from these sharp super-resolution keyframes employing a texture mapping approach. In experiments, we demonstrate that our method achieves superior quality in appearance compared to other state-of-the-art approaches.

Revised layout and minor adaptations. Accepted version of original publication [6] and detailed disclaimer are included in Chapter C.1 in the appendix.

(a) RGB input images

(b) Vertex colors

(c) Texture mapping using super-resolution keyframes

Figure 4.1: We propose an efficient method for generating high-quality textures from low-resolution RGB-D frames. Our approach significantly improves the visual quality of reconstructed 3D models while it is still fast enough for applicability in real-world 3D scanning scenarios.

4.1 Introduction

The wide availability of consumer RGB-D sensors has boosted research in 3D reconstruction in recent years. State-of-the-art methods in 3D model reconstruction yield impressively accurate geometric reconstruction in real-time [84, 126]. Such 3D reconstructions are well suitable for 3D printing [110]. Fast and robust estimation of high-quality visual appearance (i.e. texture) of the models has been given less attention. This plays, however, an equally important role for 3D modeling, for instance, of persons or objects.

Modern texture mapping approaches can obtain good-quality results, but are typically slow and impractical for instant 3D scanning applications. As scanning the 3D geometry with RGB-D sensors is possible in real-time, also the texture mapping process should be fast. We propose a method for fast and accurate reconstruction of geometry as well as appearance. Figure 4.1 shows a textured 3D model generated from low-resolution (LR) RGB-D input frames with our approach. For geometric reconstruction, we use a direct keyframe-based RGB-D SLAM method in order to estimate the camera trajectory consistently. Using these pose estimates, the individual frames are integrated into a volumetric truncated signed distance function (TSDF) representation, from which a 3D mesh is extracted. For this mesh we find a parametrization suitable for texture mapping. We significantly improve the quality of the generated texture maps through super-resolution (SR) fusion of RGB-D frames and deblurring. Simple weighted median filtering of projected color values onto the texture provides high-quality appearance results.

In experiments, we compare our method to standard pipelines that perform per-vertex coloring or texture mapping based on the original low-resolution frames. We demonstrate superior results of our method with respect to texture quality. We also evaluate the timing of our method and find that it yields high-quality results in reasonable and practical time for 3D scanning applications.

4.1.1 Related Work

Since the recent advent of low-cost commodity RGB-D sensors, there has been extensive research in the field of dense 3D reconstruction from RGB-D data. While generating highly accurate 3D models from RGB-D data has been investigated intensively, there seems to be a shortage of research in improving the visual appearance of such reconstructions.

To obtain geometrically accurate 3D reconstructions, Newcombe et al. [84] fuse RGB-D frames into a TSDF Volume and perform camera tracking against this model. Sturm et al. [110] developed a similar approach for reconstructing 3D printable models of persons, paired with direct TSDF tracking [21]. Other RGB-D SLAM methods [32, 44, 7, 109] are based on frame-to-(key)frame tracking with trajectory optimization and data fusion into a single model volume. Kerl et al. [57] developed a robust dense visual SLAM system that shows limited drift by combining dense robust visual odometry estimation with pose graph optimization. SLAM systems for reconstructing and mapping large-scale environments have also been developed [86, 106].

The systems presented above can produce models of high metric precision, however the state-of-the-art for representing the visual appearance in such 3D reconstruction systems is still volumetric averaging of per-vertex colors, such that the color resolution is limited to mesh resolution. Mostly, these vertex colors are computed as a weighted average of the observed colors for the respective vertices [84, 110, 119]. To improve the appearance, weights based on the normals computed from the depth image are employed; to remove further artifacts, pixels close to depth discontinuities are discarded.

However, to create photo-realistic 3D models of real-world objects, the challenging problem of generating and mapping high-quality textures from multiple input color images has been investigated intensively in the field of computer graphics for decades. Without increasing the geometric complexity, textures (usually of higher resolution than the mesh resolution) are mapped onto the mesh to enhance the visual quality, with camera poses assumed to be given. [83, 103] compute the texel colors using a weighted average of the observations in the input color images. Instead of using the weighted average, Coorg and Teller [24] use a color computation scheme based on weighted median to cope with color outliers in the observations. Eisemann et al. [30] correct for inac-

curacies in camera poses and calibration using optical flow for mapping images to the texture map. Lempitsky and Ivanov [66] and Gal et al. [36] select a single input view per face and minimize seams, however the approaches suffer from high runtimes because of the computationally expensive combinatorial optimization. Variational super-resolution methods, e.g. by Goldlücke et al. [39], produce compelling results, paired with impractical computation times of several hours in a controlled setup with only a limited number of input views. Waechter et al. [116] texture large-scale scenes reconstructed with Structure-from-Motion, however they rely on high-quality input images and have long computation times with up to 80 minutes per dataset.

The scenario of improving the visual appearance in RGB-D based 3D reconstruction has not been tackled extensively yet. Meilland and Comport [79] fuse low-resolution images into a single high-resolution keyframe by applying a super-resolution technique. The fused keyframes exhibit an impressive level of detail, but the approach does not create a globally consistent 3D model. Recently, Zhou and Koltun [133] have shown that the colors of 3D models obtained from handheld RGB-D cameras can be improved substantially; within several minutes, they alternatingly optimize camera poses and non-rigid correction to correct for imprecise camera localization and for complex distortions resulting from inaccurate geometric models. However, they use vertex colors of an upsampled mesh, leading to an increasingly complex geometry with a still limited resolution compared to texture maps.

To the best of our knowledge, we present the first method for combining keyframe fusion with texture mapping in an RGB-D based 3D reconstruction scenario. Our practical approach is efficient, with runtimes within a few minutes, and suitable for generating high-quality texture maps from low-quality color images obtained from consumer RGB-D sensors.

4.1.2 Contributions

In summary, we propose a novel fast and robust 3D modeling approach that provides accurate geometry and high-quality appearance. Our method uses direct keyframe-based RGB-D SLAM to find a consistent global image alignment, and extracts a high-quality mesh from a fused TSDF representation of the images.

- The mesh is parametrized in a texture map, which we fill from fused super-resolution RGB-D keyframes.
- The super-resolution RGB-D keyframes are sharpened using image deconvolution.
- Fast texture mapping is performed using the super-resolution keyframes. High quality of the texture is obtained through weighted median filtering of the keyframe projections.

4.2 3D Reconstruction System

In this section, we first describe the RGB-D sensor, the acquired data and the used camera model. We then introduce our 3D reconstruction system based on DVO-SLAM by Kerl et al. [57] and the data fusion into a TSDF volume as used by Newcombe et al. [84].

RGB-D Data Acquisition A calibrated Asus Xtion Pro Live RGB-D sensor provides us with RGB color and depth images at 30 fps at a resolution of $w \times h$ (in this case, 640×480 pixels). To limit automatic color correction during data acquisition, we fix exposure and white balance. We assume that depth and color images are registered. Since both color and depth images are utilized for real-time camera tracking, we cannot use the SXGA (1280×1024) color images provided at only 10 fps. We denote RGB images with $\mathcal{C} : \Omega_{\mathcal{C}} \to \mathbb{R}^3$ and depth images with $\mathcal{Z} : \Omega_{\mathcal{Z}} \to \mathbb{R}$.

Camera Model For the RGB-D sensor, we assume the pinhole camera model with focal length f_x, f_y and optical center c_x, c_y. The projection function π maps 3D points $\boldsymbol{p} = (X, Y, Z)^\top$ to 2D pixels $\boldsymbol{x} = (x, y)^\top$:

$$\boldsymbol{x} = \pi(\boldsymbol{p}) = \left(\frac{X}{Z} f_x + c_x, \frac{Y}{Z} f_y + c_y \right), \tag{4.1}$$

while 2D pixel locations $\boldsymbol{x}$ are mapped back to 3D points using their depth values $\mathcal{Z}(\boldsymbol{x})$ by the inverse projection π^{-1}:

$$\boldsymbol{p} = \pi^{-1}(\boldsymbol{x}, \mathcal{Z}(\boldsymbol{x})) = \left(\frac{x - c_x}{f_x}, \frac{y - c_y}{f_y}, 1 \right)^\top \mathcal{Z}(\boldsymbol{x}). \tag{4.2}$$

3D Reconstruction Framework DVO-SLAM performs dense camera tracking in real-time on the CPU and minimizes the photometric and geometric error between two RGB-D input frames to compute the relative pose. The use of color images significantly improves camera tracking and limits the drift of the SLAM system. Similarly, we perform an entropy-based loop closure detection and continuously optimize the pose graph in order to obtain a globally consistent camera trajectory.

To reconstruct a dense 3D model in a post-processing step, we fuse the N acquired RGB-D frames into a TSDF volume using their estimated absolute camera poses $\mathcal{T}_i = (\boldsymbol{R}, \boldsymbol{t}) \in \mathbb{SE}(3)$ (with $i \in 1 \dots N$, $\boldsymbol{t} \in \mathbb{R}^3$ and $\boldsymbol{R} \in \mathbb{SO}(3)$). We extract a 3D mesh $\mathcal{M} = (\mathcal{V}, \mathcal{F})$ with vertices $\mathcal{V}$ and faces $\mathcal{F}$ using the Marching Cubes algorithm. The camera poses exhibit only very limited drift due to the global pose graph optimization and hence the resulting 3D model is geometrically accurate.

4.3 Keyframe Fusion

Given an accurate geometric 3D model, reconstructed as described above, and the absolute camera poses for the input frames, we first fuse N_w neighboring frames into a common keyframe representation of higher resolution. We denote the color image of such a SR keyframe as $\mathcal{C}^* : \Omega_{\mathcal{C}} \to \mathbb{R}^3$ and the corresponding depth image as $\mathcal{Z}^* : \Omega_{\mathcal{Z}} \to \mathbb{R}$. To store the depth fusion weights, we introduce a depth weight image $\mathcal{W}^* : \Omega_{\mathcal{W}} \to \mathbb{R}$. These SR keyframes have the dimensions $sw \times sh$, where s is a scale factor that determines the amount of upsampling. We set the pose $\mathcal{T}^*$ of the SR keyframe to the first pose of the N_w LR frames to be fused. To integrate the LR images into the SR images, we additionally need to define the scale-dependent projection π_s and inverse projection π_s^{-1}, which use the upscaled intrinsic parameters sf_x, sf_y, sc_x, sc_y.

Depth Fusion We first fuse all N_w LR depth images into the corresponding SR depth image. Therefore, we compute the weights for the measured depth values, which is based on a theoretical random error model [59], as follows:

$$w_z(d) = \frac{fb}{\sigma_d} d^{-2}, \tag{4.3}$$

with the depth camera's focal length f, baseline b and disparity error standard deviation σ_d. Next, we transform the current depth image i into the keyframe's camera coordinate system using the relative transformation $\mathcal{T}^{*-1}\mathcal{T}_i$ between them:

$$\boldsymbol{p}^* = (X^*, Y^*, Z^*)^\top = \mathcal{T}^{*-1}\mathcal{T}_i \pi^{-1}(\boldsymbol{x}, \mathcal{Z}_i(\boldsymbol{x})). \tag{4.4}$$

We then use the image point $\boldsymbol{x}^* = \pi_s(\boldsymbol{p}^*)$ of the projection into the keyframe depth image to update the fused depth values and depth weights by weighted averaging:

$$\mathcal{Z}^*(\boldsymbol{x}^*) = \frac{\mathcal{W}^*(\boldsymbol{x}^*)\mathcal{Z}^*(\boldsymbol{x}^*) + w_z(\mathcal{Z}_i(\boldsymbol{x}))Z^*}{\mathcal{W}^*(\boldsymbol{x}^*) + w_z(\mathcal{Z}_i(\boldsymbol{x}))} \tag{4.5}$$

$$\mathcal{W}^*(\boldsymbol{x}^*) = \mathcal{W}^*(\boldsymbol{x}^*) + w_z(\mathcal{Z}_i(\boldsymbol{x})) \tag{4.6}$$

We achieve sub-pixel precision by updating all four neighboring depth values when transforming and projecting a depth value into the SR depth map. Occlusions are considered by fusing only the closest depth values within a given distance. After integrating all N_w LR depth images, we obtain the fused depth image $\mathcal{Z}^*$ for the SR keyframe.

Color Fusion We use the fused depth image $\mathcal{Z}^*$ to project the SR color image pixels into the LR color images. This allows us to directly look up the observed color values c_i

using bilinear interpolation:

$$c_i = \mathcal{C}_i(\pi(\mathcal{T}_i^{-1}\mathcal{T}^*\pi_s^{-1}(\boldsymbol{x}, \mathcal{Z}_i(\boldsymbol{x})))). \tag{4.7}$$

For every observation c_i, we also compute its weight

$$w_i^c = B_i w_z(\mathcal{Z}_i(\boldsymbol{x})), \tag{4.8}$$

where B_i is a measure of blurriness of the color image $\mathcal{C}_i$ according to Crete et al. [25], which downweighs views with strong motion blur. Integrating the depth into the color weights enforces that objects closer to the camera obtain higher weights. We store the observed colors and weights for pixel $\boldsymbol{x}$ in its set of color observations $\mathcal{O}_{\boldsymbol{x}} = \{(c_i, w_i^c)\}$. In order to increase color fidelity, we prune observations from $\mathcal{O}_{\boldsymbol{x}}$ with missing depth values or that are within a window of 7×7 pixels around depth discontinuities. Instead of calculating the weighted mean for averaging the color, we calculate the weighted median, for each color channel separately:

$$\mathcal{C}^*(\boldsymbol{x}) = \arg\min_c \sum_{(c_i, w_i^c) \in \mathcal{O}_{\boldsymbol{x}}} w_i^c \|c - c_i\|. \tag{4.9}$$

Since we usually have many observations per pixel, the use of weighted median is valid, which results in an overall sharper texture. The median selects the center probable value in the distribution of colors, while the mean would be heavily affected by outliers. Integrating weights into the median allows for incorporating a confidence or a prioritization of the individual color samples.

Before fusing the LR color images into the SR keyframe, we apply a Wiener filter on these LR color images as a pre-processing step. This removes motion blur and notably improves the sharpness of the visual appearance.

Note that we perform the keyframe fusion as a post-processing step; however, it is reasonable to perform this step online whenever a new keyframe is detected.

4.4 High-Quality Texture Mapping

In this section, we introduce our method for texture mapping from fused SR keyframes. First, we explain the computation of per-vertex colors based on a weighted median filtering scheme, applicable also for recomputing the vertex colors. We afterwards present our texture mapping approach, in which we compute the texel colors using the weighted median from SR keyframe color images.

4.4.1 Vertex Color Computation

In order to improve the colors of 3D meshes, a very common approach is to recompute the per-vertex colors of the 3D mesh vertices $v \in \mathcal{V}$. We therefore need to determine the views, in which a vertex is visible. To check if vertex $v \in \mathbb{R}^3$ is visible in view i, we render the mesh $\mathcal{M}$ into a virtual image using its pose $\mathcal{T}_i$ and the depth camera intrinsics. v is visible in the image, if its depth value is compatible with the depth in the depth buffer used for rendering. We then get the observed color c_i^v using bilinear interpolation:

$$c_i^v = \mathcal{C}_i(\pi(\mathcal{T}_i^{-1} v)). \tag{4.10}$$

The observation weights w_i^v of vertex v in its input views are computed as follows:

$$w_i^v = \frac{\cos(\theta) B_i}{d^2}, \tag{4.11}$$

where B_i is again the blurriness measure of color image $\mathcal{C}_i$ and d is the distance from v to the camera corresponding to $\mathcal{C}_i$; θ represents the angle between the vertex normal and the view vector at v for the camera. We store all color observations for vertex v and their respective weights in $\mathcal{O}_v = \{(c_i^v, w_i^v)\}$, observations close to depth discontinuities are discarded.

We can now compute the final vertex color c_v^* as the weighted mean of the observations:

$$c_v^* = \arg\min_{c_v} \sum_{(c_i^v, w_i^v) \in \mathcal{O}_v} w_i^v \| c_v - c_i^v \|^2. \tag{4.12}$$

Since we assume that each vertex has many observations, we can also compute the final color c_v^* (separately for each color channel) using a weighted median filtering scheme:

$$c_v^* = \arg\min_{c_v} \sum_{(c_i^v, w_i^v) \in \mathcal{O}_v} w_i^v \| c_v - c_i^v \|. \tag{4.13}$$

Given enough views that observe a vertex, this simple method already improves the mesh colors and results in a more detailed appearance, as demonstrated in Section 4.5.1.

4.4.2 Texture Mapping

Based on the introduced weighted median color computation scheme, we employ texture mapping to further improve the appearance of 3D models. In particular, we use the fused SR keyframes of Section 4.3 for texture mapping, leading to a significantly higher resolved visual appearance. We denote a texture as $\mathcal{T} : \Omega_\mathcal{T} \rightarrow \mathbb{R}^3$, which stores a color value at every texel $t \in \Omega_\mathcal{T}$.

Texture Parametrization For working with texture maps, a three-dimensional mesh needs to be projected onto a planar two-dimensional texture $\mathcal{T}$ first. We beforehand simplify the mesh geometry by decimating the number of mesh triangles. This usually results in larger triangles that can be textured more efficiently with larger patches, while the geometry is still preserved well. While different planar parametrization methods exist, our approach is in general independent of the chosen parametrization, as long as the mesh faces contain texture coordinates. In practice, we mostly use Least Squares Conformal Maps by Levy et al. [67], or a simple arrangement of the mesh triangles on the texture within a rectangular grid.

Since there is a unique mapping from a texel to its containing face, we can determine the respective surrounding vertices for each texel. The barycentric mapping $\psi : \Omega_{\mathcal{T}} \rightarrow \mathbb{R}^3$ performs a one-to-one mapping from 2D texel coordinates to 3D world coordinates. Using barycentric interpolation, we can compute interpolated 3D vertices v_t corresponding to 2D texels t and vice versa:

$$v_t = \psi(t). \tag{4.14}$$

Texel Color Computation To compute the texel color for every texel t in the texture map, we employ only the N^* SR keyframes $(\mathcal{C}_l^*, \mathcal{Z}_l^*)$ with camera poses $\mathcal{T}_l^*$ (with $l \in 1 \dots N^*$), generated as described in Section 4.3.

We collect the observations of the texel by first computing its 3D vertex position v_t according to Equation (4.14). We then determine the set of color observations $\mathcal{O}_t = \{(c_l^t, w_l^t)\}$ for v_t analogous to Equation (4.10) and Equation (4.11). From these observations, we compute the final texel colors by again applying a weighted median color computation scheme:

$$\mathcal{T}(t) = \arg\min_{c_t} \sum_{(c_l^t, w_l^t) \in \mathcal{O}_v} w_l^t \| c_t - c_l^t \|. \tag{4.15}$$

4.5 Experimental Results

In this section, we evaluated our approach on real-world datasets. Three evaluation sequences *face*, *phone* and *keyboard* were acquired using a handheld Asus Xtion Pro Live, details are given in Table 4.1. We captured RGB-D data at a low resolution of 640×480 pixels at 30 fps, with fixed exposure and white-balance.

The following experimental results demonstrate that (1) vertex recoloring using weighted median filtering improves the colors of 3D models compared to weighted mean, (2) fusing LR input frames into SR keyframes and using them for texture map-

	face	*phone*	*keyboard*
# RGB-D frames	512	1359	642
# vertices (original)	159583	82942	155842
# triangles (original)	319176	165888	311686
# triangles (decimated)	40000	40000	40000

Table 4.1: Details of the acquired real-world datasets and the corresponding reconstructed 3D meshes.

ping improves the visual quality substantially, and (3) the proposed method is efficient and practical for real-world 3D scanning applications. All experiments were performed on a standard desktop PC with Intel Core i7-2600 CPU with 3.40GHz and 8GB RAM.

4.5.1 Vertex Recoloring using Weighted Median

First, we demonstrate that the visual appearance of 3D models can already be improved by using a weighted median color integration scheme. Figure 4.2 shows that the weighted mean in combination with discontinuity checks already improves the vertex colors significantly compared to unweighted mean. The weighted median increases the sharpness and level of detail even further and leads to a more realistic model. Still, the texture resolution is limited by the number of vertices so far. Mesh subdivision increases the number of vertices, but the increasing geometric mesh complexity makes processing the mesh intractable.

4.5.2 Keyframe Fusion and Texture Mapping

After showing that a weighted median color computation scheme has advantages compared to weighted mean, we investigate how texture mapping with weighted median filtering further improves the appearance of 3D models. In the following, we show qualitative results of texture mapping from fused SR keyframes in comparison with per-vertex colors, which serves as currently most popular state-of-the-art.

By fusing several LR color images into a SR keyframe, we obtain high-quality frames from low-quality input data. Depending on the scale factor s, the SR images have a resolution of 1280×960 ($s = 2$) or 2560×1920 ($s = 4$). Figure 4.3 illustrates that both color and depth of the resulting fused SR keyframes exhibit more details compared to the LR input color images and depth maps.

An important aspect of the keyframe fusion is the deconvolution of the input images with a Wiener filter for deblurring. Figure 4.4 shows the results of generating a texture

(a) Unweighted mean (b) Weighted mean (c) Weighted median

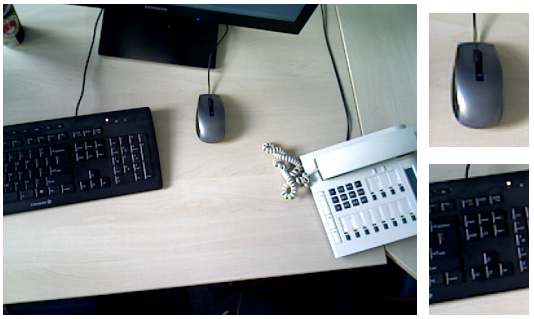
Figure 4.2: Improving the vertex colors of 3D models: (a) colors computed using the unweighted mean of the vertex can be improved by (b) using the weighted mean. (c) Applying the weighted median further improves the visual quality and preserves a higher level of detail.

map for a 3D model from SR keyframes with and without deconvolution. The texture computed from the deblurred SR keyframes (Figure 4.4(b)) exhibits a sharper texture with substantially more details compared to Figure 4.4(a). For deblurring, a Wiener filter is applied on the LR input images as a pre-processing step before fusing them into the keyframes.

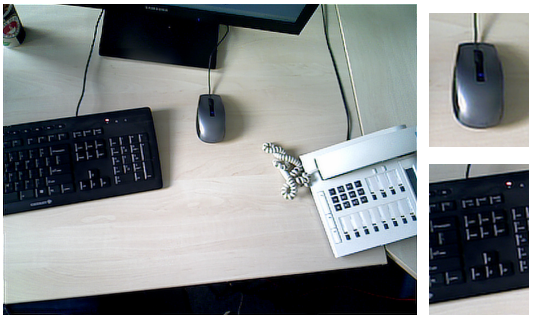
Next, we compare the reconstructed surface colors depending on the scale factor s for the SR keyframe dimensions. The textures shown in Figure 4.5 show that the level of detail can be slightly improved by using a higher keyframe resolution of 2560×1920 ($s = 4$) compared to a resolution of 1280×960 ($s = 2$).

For comparison, Figure 4.6 finally shows the improvements of texture mapping with SR keyframes compared to texture mapping with the LR input images only.

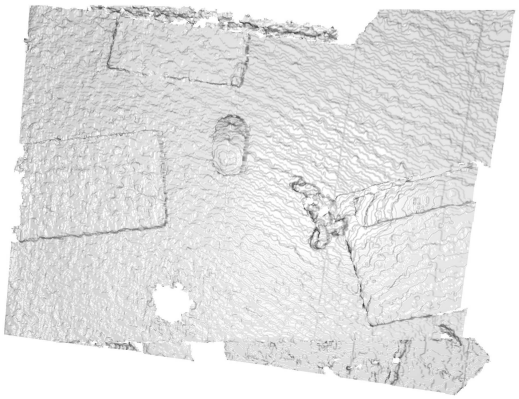
To demonstrate the practicability of our approach, we have reconstructed 3D models of the *face*, *phone* and *keyboard* datasets. All textures have been computed by fusion into SR keyframes of dimensions 2560×1920 and using weighted median filtering for



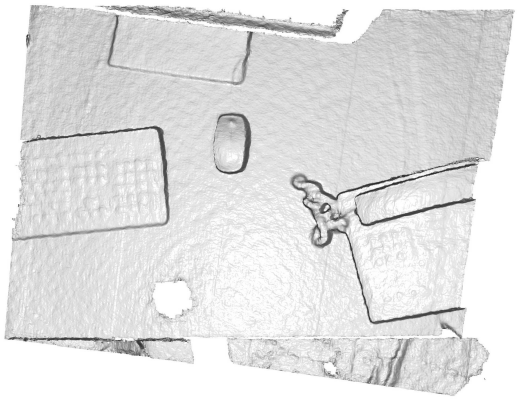
(a) LR input color image



(b) Fused SR color image



(c) LR input depth map (Phong shading)

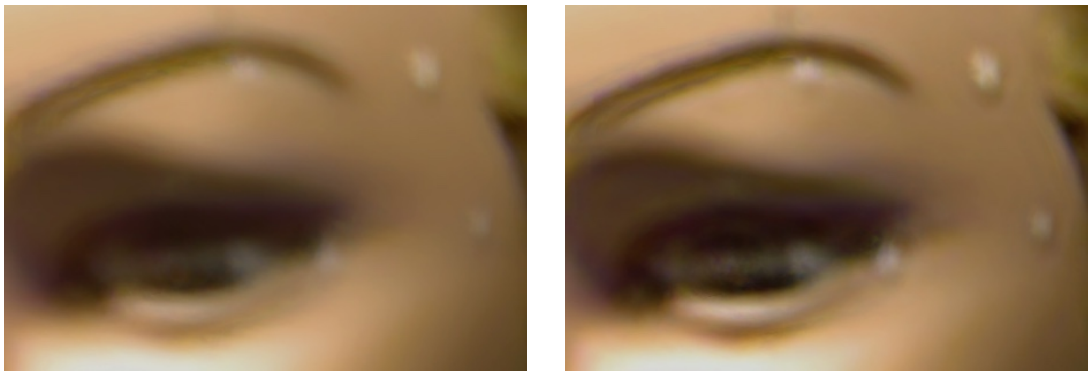


(d) Fused SR depth map (Phong shading)

Figure 4.3: Fusing several LR input color images into a single SR keyframe allows to directly obtain high-quality color images. Compared to the LR input color image (a), the fused SR color image (b) with a resolution of 2560×1920 (scale $s = 4$) exhibits more details. Similarly, the LR input depth map (c) shows significantly more noise than the fused SR depth map (d).

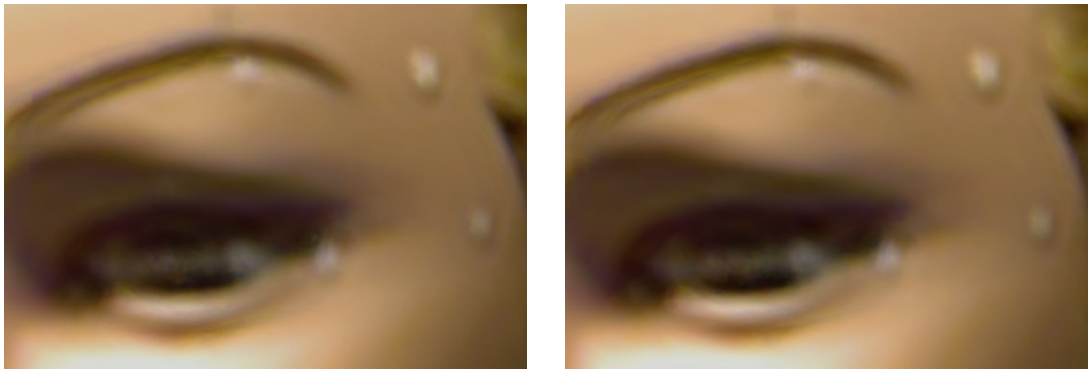
computing the texel colors. As a pre-processing step, a Wiener filter has been applied to the LR input RGB images. Figures 4.1, 4.7 and 4.8 show the results. The texture mapped 3D models provide a photo-realistic appearance and exhibit fine surface details that are not visible in the models with per-vertex colors only.

In Figure 4.8(c), the cable at the top of the keyboard is however not represented correctly in the texture. This may either be due to inaccuracies in the estimated camera trajectory or due to an inaccurate geometric model. To compensate for this, an approach similar to Zhou and Koltun [133] must be developed, which optimizes the camera poses as well as non-rigid image corrections.



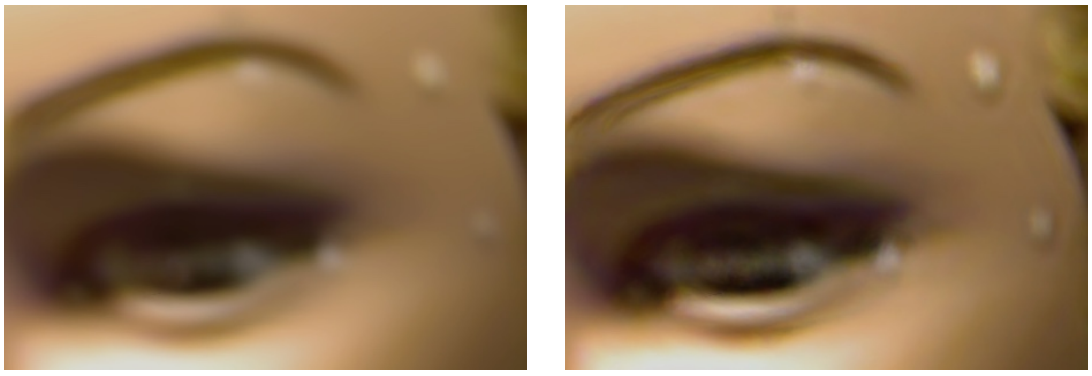
(a) Without deconvolution (b) With deconvolution

Figure 4.4: (a) The textures computed from SR keyframes are substantially improved by (b) applying deconvolution (e.g. using a Wiener filter) to the input images before the keyframe fusion.



(a) Keyframes of dimensions 1280×960 (b) Keyframes of dimensions 2560×1920

Figure 4.5: (a) The textures generated from keyframes of dimensions 1280×960 show slightly fewer details than (b) the ones generated from keyframes of dimensions 2560×1920.



(a) With LR input frames (b) With SR keyframes

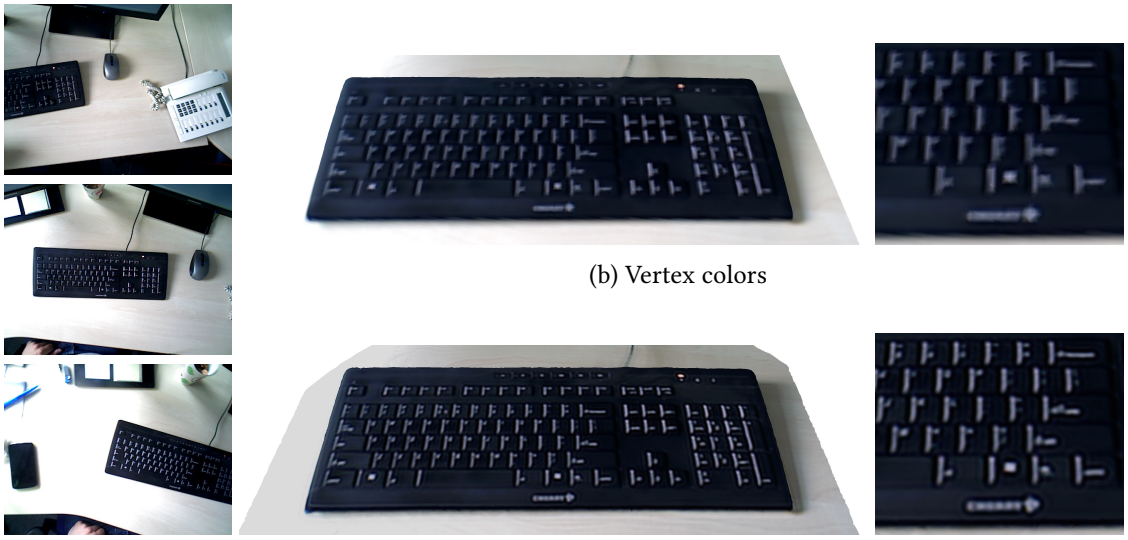
Figure 4.6: (a) Texture mapping with LR input frames only yields inferior results compared to (b) texture mapping with SR keyframe fusion.

(a) RGB input images

(b) Vertex colors

(c) Texture mapping using SR keyframes

Figure 4.7: 3D model of the *phone* dataset reconstructed and textured using our approach: We show (a) some input images and (b) the 3D model with vertex colors only. (c) The texture mapped reconstruction provides a significantly more detailed visual appearance.



(a) RGB input images

(b) Vertex colors

(c) Texture mapping using SR keyframes

Figure 4.8: 3D model of the *keyboard* dataset reconstructed and textured using our approach: We show (a) some input images and (b) the 3D model with vertex colors only. (c) The texture mapped reconstruction provides a significantly more detailed visual appearance.

		face		*phone*		*keyboard*	
	s	t [s]	fps	t [s]	fps	t [s]	fps
Texture Mapping		91.5	5.6	330.8	4.1	128.8	5.0
Keyframe Fusion	2	57.5	8.9	222.0	6.1	72.1	8.9
SR Texture Mapping	2	18.7	2.8	50.7	2.7	18.8	3.5
Keyframe Fusion	4	100.9	5.1	362.8	2.2	214.9	3.0
SR Texture Mapping	4	26.4	2.0	58.2	1.4	42.6	1.5

Table 4.2: Runtimes (in seconds) for texture mapping without SR keyframe fusion and texture mapping with SR keyframe fusion.

4.5.3 Runtime Evaluation

We finally evaluate the runtime and efficiency of the proposed texture mapping method, in particular the runtimes for keyframe fusion and texture mapping. Table 4.2 gives the results. With runtimes of between one and a few minutes, our approach is a very efficient method for generating high-quality texture maps. Since our implementation is based only on the CPU, a major speed-up can be achieved by porting the algorithm to the GPU. This holds in particular for the keyframe fusion, which has already been shown to work in real-time on a GPU [79].

4.6 Conclusion

We presented a novel efficient method for high-quality texture mapping in RGB-D-based 3D reconstruction approaches. Our method fuses low-quality color images from commodity depth sensors into super-resolution keyframes. These high-quality keyframes in turn are then mapped into a global texture for the 3D model, resulting in a significantly improved texture quality compared to simple volumetric blending. We deblur input images and use the weighted median for computing the texel colors from observations, which preserves a high level of detail. Using the weighted median already provides better results for vertex coloring compared to the weighted mean. The weights in our method consider criteria such as view-angle, motion blur, and distance to the surface.

We have shown in experimental results that our method produces high-quality textures that substantially increase the photo-realism of the reconstructed 3D models. At the same time, our method is a very efficient and practical post-processing step with runtimes within a few minutes, making it useful for real-world 3D scanning application scenarios.

Acknowledgements This work has been partially funded by the ERC Proof of Concept grant CopyMe3D (GA 632200) and the BMWi ZIM project 2Dzu3D (KF2080213CR4).

Chapter 5

Intrinsic3D: High-Quality 3D Reconstruction by Joint Appearance and Geometry Optimization with Spatially-Varying Lighting

Authors:	Robert Maier[1,2]	robert.maier@tum.de
	Kihwan Kim[1]	kihwank@nvidia.com
	Daniel Cremers[2]	cremers@tum.de
	Jan Kautz[1]	jkautz@nvidia.com
	Matthias Nießner[2,3]	niessner@tum.de

[1] NVIDIA, Santa Clara, CA, USA
[2] Technische Universität München, Germany
[3] Stanford University, Stanford, CA, USA

Conference: *IEEE International Conference on Computer Vision (ICCV) 2017.*
DOI: 10.1109/ICCV.2017.338.

Abstract We introduce a novel method to obtain high-quality 3D reconstructions from consumer RGB-D sensors. Our core idea is to simultaneously optimize for geometry encoded in a signed distance field (SDF), textures from automatically-selected keyframes, and their camera poses along with material and scene lighting. To this end, we propose a joint surface reconstruction approach that is based on Shape-from-Shading (SfS) techniques and utilizes the estimation of spatially-varying spherical harmonics (SVSH) from subvolumes of the reconstructed scene. Through extensive examples and evaluations, we demonstrate that our method dramatically increases the level of detail in the reconstructed scene geometry and contributes highly to consistent surface texture recovery.

Revised layout and minor adaptations. Accepted version of original publication [4] and detailed disclaimer are included in Chapter C.2 in the appendix.

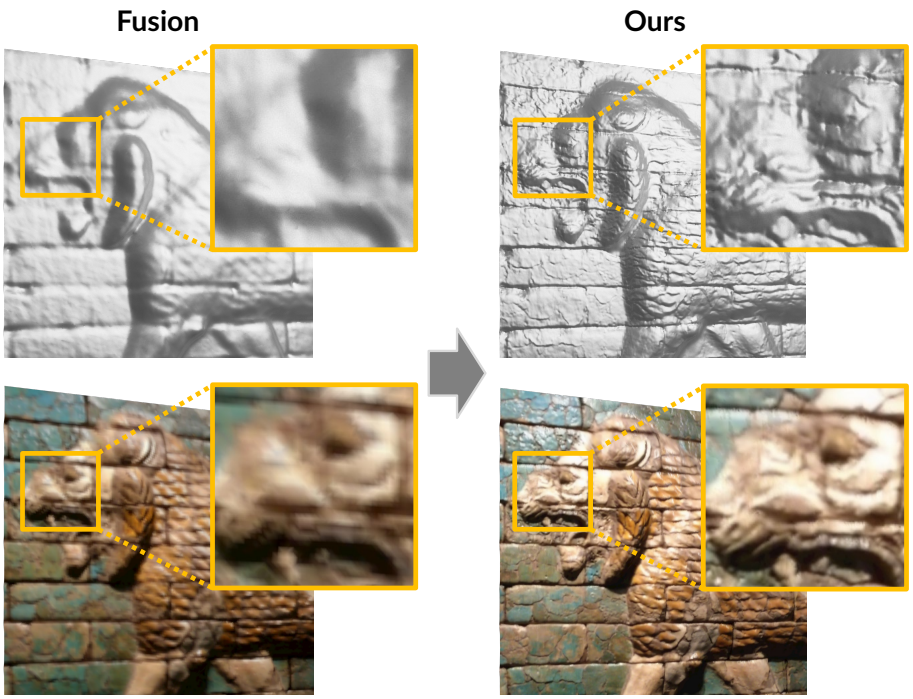


Figure 5.1: Our 3D reconstruction method jointly optimizes geometry and intrinsic material properties encoded in a Signed Distance Field (SDF), as well as the image formation model to produce high-quality models of fine-detail geometry (top) and compelling visual appearance (bottom).

5.1 Introduction

With the wide availability of commodity RGB-D sensors such as the Microsoft Kinect, Intel RealSense, or Google Tango, reconstruction of 3D scenes has gained significant attention. Along with new hardware, researchers have developed impressive approaches that are able to reconstruct 3D surfaces from the noisy depth measurements of these low-cost devices. A very popular strategy to handle strong noise characteristics is volumetric fusion of independent depth frames [26], which has become the core of many state-of-the-art RGB-D reconstruction frameworks [23, 28, 84, 86, 104].

Volumetric fusion is a fast and efficient solution for regularizing out sensor noise; however, due to its ℓ_2-regularization property, it tends to oversmooth the reconstruction, leaving little fine-scale surface detail in the result. The same problem also translates to reconstruction of surface textures. Most RGB-D reconstruction frameworks simply map RGB values of associated depth pixels onto the geometry by averaging all colors that have been observed for a given voxel. This typically leads to blurry textures, as wrong surface geometry and misaligned poses introduce re-projection errors where one voxel is associated with different color values that are then incorrectly averaged.

Very recent approaches address these two problems independently. For instance, Zhou and Koltun [133] optimize for consistent surface textures by iteratively solving for rigid pose alignment and color averages. To compensate for wrong surface geometry where re-projection consistency is infeasible, they non-rigidly warp RGB frames on top of the reconstructed mesh, thus obtaining a high-quality surface texture. On the other end of the spectrum, shading-based refinement techniques enhance depth frames [126] or surface geometry [135] by adding shading constraints from higher resolution color frames; i.e., they leverage RGB signal to refine the geometry. These reconstruction pipelines are sequential; for instance, Zollhöfer et al. [135] first compute the alignment between RGB-D frames, then fuse both RGB and depth data into a volumetric grid, and finally refine the 3D reconstruction. This results in visually promising reconstructions; however, the pipeline fundamentally cannot recover errors in its early stages; e.g., if pose alignment is off due to wrong depth measures, fused colors will be blurry, causing the following geometry refinement to fail.

In our work, we bring these two directions together by addressing these core problems simultaneously rather than separately. Our main idea is to compute accurate surface geometry such that color re-projections of the reconstructed texture are globally consistent. This leads to sharp surface colors, which can again provide constraints for correct 3D geometry. To achieve this goal, we introduce a novel joint optimization formulation that solves for all parameters of a global scene formation model: (1) surface geometry, represented by an implicit signed distance function, is constrained by input depth measures as well as a shading term from the RGB frames; (2) correct poses and intrinsic camera parameters are enforced by global photometric and geometric consistency; (3) surface texture inconsistency is minimized considering all inputs along with the 3D model; and (4) spatially-varying lighting as well as surface albedo values are constrained by RGB measures and surface geometry. The core contribution of our work is to provide a parametric model for all of these intrinsic 3D scene parameters and optimize them in a joint, continuous energy minimization for a given RGB-D sequence. As a result, we achieve both sharp color reconstruction, highly-detailed and physically-correct surface geometry (Figure 5.1), and an accurate representation of the scene lighting along with the surface albedo. In a series of thorough evaluations, we demonstrate that our method outperforms state-of-the-art approaches by a significant margin, both qualitatively and quantitatively.

To sum up, our technical contributions are as follows:

- We reconstruct a volumetric signed distance function by jointly optimizing for 3D geometry, surface material (albedo), camera poses, camera intrinsics (including lens distortion), as well as accurate scene lighting using spherical harmonics basis functions.

- Instead of estimating only a single, global scene illumination, we estimate spatially-varying spherical harmonics to retrieve accurate scene lighting.
- We utilize temporal view sampling and filtering techniques to mitigate the influence of motion blur, thus efficiently handling data from low-cost consumer-grade RGB-D sensor devices.

5.2 Related Work

3D Reconstruction using Signed Distance Functions Implicit surface representations have been widely used in 3D modeling and reconstruction algorithms. In particular, signed distance fields (SDF) [26] are often used to encode 3D surfaces in a voxel grid, and have become the basis of many successful RGB-D surface reconstruction algorithms [84, 86]. More recently, Choi et al. [23] propose a robust optimization for high-quality pose alignment using only geometry, and Dai et al. [28] present a global optimization for large-scale scenes in real time. While most SDF-based fusion methods efficiently regularize noisy depth input, they spend little focus on reconstructing consistent and sharp surface textures. In particular, in the context of wide baseline views and small surface misalignments, this leads to blurry voxel colors that are obtained by averaging the input RGB values of associated color images.

High-quality texture recovery In order to compute consistent colors on the reconstructed surface, Zhou and Koltun [133] introduce a method to optimize the mapping of colors onto the geometry (camera poses and 2D deformation grid), Klose et al. [62] propose to filter colors in scene space, and Jeon et al. [51] suggest a more efficient way of color optimization through texture coordinates. In addition to directly optimizing for consistent surface textures, refining texture quality also helps to improve the quality of reconstructed surface colors [39, 6]. While these methods achieve visually impressive RGB reconstructions (e.g., by warping RGB input), they do not address the core problem of color inconsistency, which is caused by wrong surface geometry that leads to inconsistent RGB-to-RGB and RGB-to-geometry re-projections.

Shading- and reflectance-based geometry refinement Shape-from-Shading [46, 131] aims to extract 3D geometry from a single RGB image, and forms the mathematical basis of shading-based refinement, targeted by our work. The theory behind Shape-from-Shading is well-studied, in particular when the surface reflectance, light source and camera locations are known. Unfortunately, the underlying optimizations are highly under-constrained, particularly in uncontrolled environments. Thus, one direction is to refine coarse image-based shape models based on incorporation of shading cues [16]. For instance, this can be achieved with images captured by multiple cameras [124, 125]

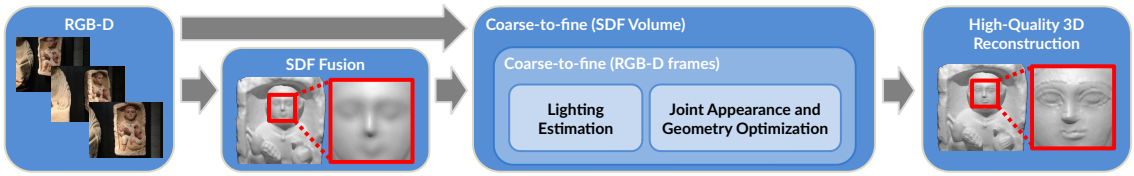


Figure 5.2: Overview of our method for joint appearance and geometry optimization. Our pipeline takes RGB-D data of a scene as input and fuses it into a Signed Distance Field (SDF). In a nested coarse-to-fine approach, spatially-varying lighting is estimated and used to jointly optimize for appearance and geometry of the scene, producing a high-quality 3D model.

or with RGB-D cameras that provide an initial depth estimate for every pixel [14, 41, 128].

Hence, shading and reflectance estimation has become an important contextual cue for refining geometry. Many methods leverage these cues to develop high-quality surface refinement approaches [15, 31, 126]. In particular, Zollhöfer et al. [135] motivates our direction of using volumetric signed distance fields to represent the 3D model. Unfortunately, the method has significant drawbacks; first, it only assumes a single global lighting setting based on spherical harmonics [88] that is constant over the entire scene; second, its pipeline is sequential, meaning that poses and surface colors are optimized only once in a pre-process, suffering from erroneous depth measures and small pose misalignments. In our approach, we systematically address these shortcomings with a joint optimization strategy, as well as a much more flexible spatially-varying lighting parametrization. Other related methods focus on specular surfaces with an alternating optimization strategy [127], represent lighting with illumination maps [73], or retrieve a box-like 3D representation with material parameters [130].

5.3 Overview

Our method first estimates a coarse sparse Signed Distance Field (SDF) similar to Nießner et al. [86] from an input RGB-D sequence with initial camera poses. To mitigate the influence of views with motion blur, we automatically select views based on a blurriness measure and constrain the optimization only based on color values from these keyframes.

Our joint optimization employs a nested hierarchical approach (see Figure 5.2): in an outer loop, we refine the SDF in a coarse-to-fine manner on multiple SDF grid pyramid levels in order to reconstruct fine detail. At the coarsest grid pyramid level, we use multiple RGB-D frame pyramid levels of all keyframes obtained through downsampling in order to improve the convergence and robustness of the joint camera pose estimation.

Within each inner iteration, we approximate complex scene lighting by partitioning the SDF volume into subvolumes of fixed size with separate spherical harmonics parameters. During estimation, we jointly solve for all SH parameters on a global scale with a Laplacian regularizer. The lighting at a given point is defined as the trilinear interpolation of the associated subvolumes.

In the main stage of our framework, we employ the estimated illumination to jointly refine surface and albedo of the SDF as well as the image formation model (camera poses of the input frames, camera intrinsics and lens distortion). As a consequence of this extensive set of optimized parameters, we implicitly obtain optimal colors. We re-compute the voxel colors from the keyframes using the refined parameters after each optimization. Finally, a 3D mesh is extracted from the refined SDF using Marching Cubes [75].

5.3.1 Signed Distance Field

At the core of our framework lies the reconstructed surface, which we implicitly store as a sparse Truncated Signed Distance Function (TSDF) [26], denoted by $\mathbf{D}$. Hereby, each voxel stores the raw (truncated) signed distance to the closest surface $\mathbf{D}(\boldsymbol{v})$, its color $\mathbf{C}(\boldsymbol{v})$, an integration weight $\mathbf{W}(\boldsymbol{v})$, an illumination albedo $\mathbf{a}(\boldsymbol{v})$, and an optimized signed distance $\tilde{\mathbf{D}}(\boldsymbol{v})$. We denote the current estimate of the iso-surface by $\mathbf{D}_0$ and the number of voxels in the SDF volume by N.

Following state-of-the-art reconstruction methods, we integrate depth maps into the SDF using a weighted running average scheme:

$$\mathbf{D}(\boldsymbol{v}) = \frac{\sum_{i=1}^{M} w_i(\boldsymbol{v}) d_i(\boldsymbol{v})}{\mathbf{W}(\boldsymbol{v})}, \quad \mathbf{W}(\boldsymbol{v}) = \sum_{i=1}^{M} w_i(\boldsymbol{v}), \tag{5.1}$$

with sample integration weight $w_i(\boldsymbol{v}) = \cos(\theta)$, based on the angle θ between the viewing direction and the normal computed from the input depth map. The truncated signed distance $d_i(\boldsymbol{v})$ between a voxel and a depth frame $\mathcal{Z}_i$ with pose $\mathcal{T}_i$ is computed as follows:

$$d_i(\boldsymbol{v}) = \Psi((\mathcal{T}_i^{-1}\boldsymbol{v})_z - \mathcal{Z}_i(\pi(\mathcal{T}_i^{-1}\boldsymbol{v})), \tag{5.2}$$

with truncation $\Psi(d) = \min(|d|, t_{\text{trunc}}) \cdot \text{sgn}(d)$. After integrating all frames of the RGB-D sequence in the implicit 3D model representation, we initialize the optimized SDF $\tilde{\mathbf{D}}$ with the integrated SDF $\mathbf{D}$. We directly compute the surface normal for each voxel from the gradient of the refined signed distance field using forward differences:

$$\mathbf{n}(\boldsymbol{v}) = (n_x, n_y, n_z)^\top = \frac{\nabla\tilde{\mathbf{D}}(\boldsymbol{v})}{||\nabla\tilde{\mathbf{D}}(\boldsymbol{v})||_2}, \tag{5.3}$$

with the gradient

$$\nabla\tilde{\mathbf{D}}(\boldsymbol{v}) = \nabla\tilde{\mathbf{D}}(i,j,k) = \begin{pmatrix} \tilde{\mathbf{D}}(i+1,j,k) - \tilde{\mathbf{D}}(i,j,k) \\ \tilde{\mathbf{D}}(i,j+1,k) - \tilde{\mathbf{D}}(i,j,k) \\ \tilde{\mathbf{D}}(i,j,k+1) - \tilde{\mathbf{D}}(i,j,k) \end{pmatrix} \tag{5.4}$$

where $\tilde{\mathbf{D}}(i,j,k)$ is the optimized distance value at the (discrete) voxel location (i,j,k). Since each voxel encodes the distance to its closest surface, it is possible to derive a corresponding 3D point on the iso-surface $\boldsymbol{v}_0$. Thus, the voxel center point $\boldsymbol{v}_c \in \mathbb{R}^3$ in world coordinates is projected onto the (nearest) iso-surface using the transformation ψ:

$$\boldsymbol{v}_0 = \psi(\boldsymbol{v}) = \boldsymbol{v}_c - \mathbf{n}(\boldsymbol{v})\tilde{\mathbf{D}}(\boldsymbol{v}). \tag{5.5}$$

5.3.2 Image Formation Model and Sampling

RGB-D Data As input, our framework takes M RGB-D frames with registered color images $\mathcal{C}_i$, derived intensity images $\mathcal{I}_i$, and depth maps $\mathcal{Z}_i$ (with $i \in 1 \dots M$). We assume exposure and white balance of the sensor to be fixed, which is a common setting in RGB-D sensors. Moreover, we are given an initial estimate of the absolute camera poses $\mathcal{T} = \{\mathcal{T}_i\}$ of the respective frames, with $\mathcal{T}_i = (\boldsymbol{R}_i, \boldsymbol{t}_i) \in \mathbb{SE}(3)$, $\boldsymbol{R}_i \in \mathbb{SO}(3)$ and $\boldsymbol{t}_i \in \mathbb{R}^3$. We denote the transformation of a point $\boldsymbol{p}$ using a pose $\mathcal{T}_i$ by $g(\mathcal{T}_i, \boldsymbol{p}) = \boldsymbol{R}_i\boldsymbol{p} + \boldsymbol{t}_i$. While our approach is based on the Voxel Hashing framework [86], the initial camera poses can in principle be computed using any state-of-the-art RGB-D based 3D reconstruction system; e.g., [23, 28].

Camera Model Our camera model is defined by the focal length f_x, f_y, the optical center c_x, c_y and three coefficients $\kappa_1, \kappa_2, \rho_1$ describing radial and tangential lens distortion respectively. 3D points $\boldsymbol{p} = (X, Y, Z)^\top$ are mapped to 2D image pixels $\boldsymbol{x} = (x, y)^\top$ with the projection function $\pi : \mathbb{R}^3 \to \mathbb{R}^2$.

Keyframe Selection In hand-held RGB-D scanning, input images often exhibit severe motion blur due to fast camera motion. To mitigate the effect of motion blur, we discard bad views by selecting views using the blurriness measure by Crete et al. [25]. More specifically, we choose the least blurred frame within a fixed size window of t_{KF} neighboring frames. We set $t_{\text{KF}} = 20$ for regular datasets that are captured with commodity RGB-D sensors, and $t_{\text{KF}} = 5$ for short sequences with less than 100 frames. Our method can also be applied to multi-view stereo datasets consisting of only few images; here, we use all frames (i.e., $t_{\text{KF}} = 1$).

Observations Sampling and Colorization After generating the SDF volume, we initially compute the voxel colors by sampling the selected keyframes. Given a frame $(\mathcal{C}_i, \mathcal{Z}_i)$ and its pose $\mathcal{T}_i$, we re-compute the color of a voxel $\boldsymbol{v}$ by sampling its 3D iso-surface point $\boldsymbol{v}_0$ in the input views. To check whether voxel $\boldsymbol{v}$ is visible in view i, we transform $\boldsymbol{v}_0$ back into the input view's coordinate system using the (refined) pose $\mathcal{T}_i$, project it into its depth map $\mathcal{Z}_i$ and look up the respective depth value. $\boldsymbol{v}$ is considered visible in the image if the voxel's z-coordinate in the camera coordinate system is compatible with the sampled depth value.

We collect all color observations of a voxel in its views and their respective weights in $\mathcal{O}_{\boldsymbol{v}} = \{(c_i^{\boldsymbol{v}}, w_i^{\boldsymbol{v}})\}$. The observed colors $c_i^{\boldsymbol{v}}$ are obtained by sampling from the input color image $\mathcal{C}_i$ using bilinear interpolation:

$$c_i^v = \mathcal{C}_i(\pi(\mathcal{T}_i^{-1}\boldsymbol{v}_0)). \tag{5.6}$$

The observation weight $w_i^{\boldsymbol{v}}$ is view-dependent on both normal and depth in the view:

$$w_i^{\boldsymbol{v}} = \frac{\cos(\theta)}{d^2}, \tag{5.7}$$

where d is the distance from $\boldsymbol{v}$ to the camera corresponding to $\mathcal{C}_i$. θ represents the angle between the voxel normal $\mathbf{n}(\boldsymbol{v})$ rotated into the camera coordinate system, and the view direction of the camera.

Colorization We sort the observations in $\mathcal{O}_{\boldsymbol{v}}$ by their weight and keep only the best t_{best} observations. The voxel color $c_{\boldsymbol{v}}^*$ is computed as the weighted mean of its observations $\mathcal{O}_{\boldsymbol{v}}$ (for each color channel independently):

$$c_{\boldsymbol{v}}^* = \arg\min_{c_{\boldsymbol{v}}} \sum_{(c_i^{\boldsymbol{v}}, w_i^{\boldsymbol{v}}) \in \mathcal{O}_{\boldsymbol{v}}} w_i^{\boldsymbol{v}} (c_{\boldsymbol{v}} - c_i^{\boldsymbol{v}})^2. \tag{5.8}$$

Note that the per-voxel colors are only used before each optimization step (for up-to-date chromaticity weights) and as a final postprocess during mesh extraction. The optimization itself directly constrains the input RGB images of the selected views and does not use the per-voxel color values.

5.4 Lighting Estimation using Spatially-varying Spherical Harmonics

Lighting Model In order to represent the lighting of the scene, we use a fully-parametric model that defines the shading at every surface point w.r.t. global scene lighting. To make the problem tractable, we follow previous methods and assume that the scene environment is Lambertian.

The shading $\mathbf{B}$ at a voxel $\boldsymbol{v}$ is then computed from the voxel surface normal $\mathbf{n}(\boldsymbol{v})$, the voxel albedo $\mathbf{a}(\boldsymbol{v})$ and scene lighting parameters l_m:

$$\mathbf{B}(\boldsymbol{v}) = \mathbf{a}(\boldsymbol{v}) \sum_{m=1}^{b^2} l_m H_m(\mathbf{n}(\boldsymbol{v})), \tag{5.9}$$

with shading basis H_m. As Equation (5.9) defines the forward shading computation, our aim is to tackle the inverse rendering problem by estimating the parameters of $\mathbf{B}$.

Spherical Harmonics In order to estimate the reflected irradiance $\mathbf{B}$ (cf. Equation 5.9) at a voxel $\boldsymbol{v}$, we parametrize the lighting with spherical harmonics (SH) basis functions [88], which is known to be a good approximation and smooth for Lambertian surface reflectance. The SH basis functions H_m are parametrized by a unit normal $\mathbf{n}$. In our implementation, we use SH coefficients up to the second order, which includes $b = 3$ SH bands and leaves us with nine unknown lighting coefficients $\ell = (l_1, \ldots, l_{b^2})$. For a given surface point, the SH basis encodes the incident lighting, parameterized as a spherical distribution. However, a single SH basis cannot faithfully represent scene lighting for all surface points simultaneously, as lights are assumed to be infinitesimally far away (i.e., purely directional), and neither visibility nor occlusion is taken into account.

Subvolume Partitioning To address the shortcoming of a single, global spherical harmonics basis that globally defines the scene lighting, we extend the traditional formulation. To this end, we partition the reconstruction volume into subvolumes $\mathcal{S} = \{s_1 \ldots, s_K\}$ of fixed size t_{sv}; the number of subvolumes is denoted as K. We now assign an SH basis – each with its own SH coefficients – to every subvolume. Thus, we substantially increase the number of lighting parameters per scene and allow for spatially-adaptive lighting changes. In order to avoid aliasing artifacts at subvolume boundaries, we define the global lighting function as a trilinear interpolation of local SH coefficients; i.e., for a voxel, we obtain a smooth function defining the actual SH coefficients as an interpolation of the lighting parameters of its eights adjacent subvolumes.

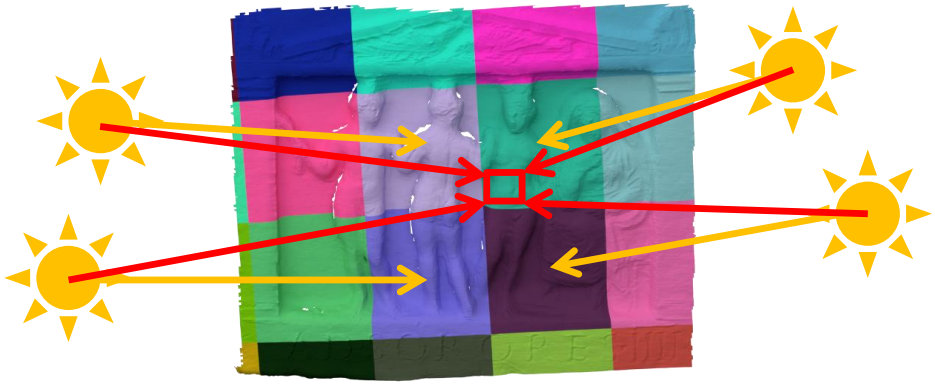


Figure 5.3: We partition the SDF volume into subvolumes of fixed size and estimate independent spherical harmonics (SH) coefficients for each subvolume (yellow). Per-voxel SH coefficients are obtained through tri-linear interpolation of the lighting of neighboring subvolumes (red).

Spatially-varying Spherical Harmonics The ability of subvolumes to define local spherical harmonics coefficients along with a global interpolant introduces the concept of spatially-varying spherical harmonics (SVSH). Instead of only representing lighting with a single set of SH coefficients, we have now $K \times b^2$ unknown parameters, that provide for significantly more expressibility in the scene lighting model. The lighting for subvolumes is estimated by minimizing the following objective:

$$E_{\text{lighting}}(\ell_1, \ldots, \ell_K) = E_{\text{appearance}} + \lambda_{\text{diffuse}} E_{\text{diffuse}}. \tag{5.10}$$

The intuition is that we try to approximate complex global illumination with varying local illumination models for smaller subvolumes. We estimate the spherical harmonics in a subvolume by minimizing the differences between the measured averaged voxel intensity and the estimated appearance:

$$E_{\text{appearance}} = \sum_{\boldsymbol{v} \in \tilde{\mathbf{D}}_0} (\mathbf{B}(\boldsymbol{v}) - \mathbf{I}(\boldsymbol{v}))^2, \tag{5.11}$$

where only voxels close to the current estimate of the iso-surface $\tilde{\mathbf{D}}_0$ are considered. Initially, we assume the albedo to be constant. However, the albedo is refined as the optimization commences. After the surface refinement on each level, we recompute the voxel colors (and hence voxel intensity). We further regularize the distribution of lighting coefficients with a Laplacian regularizer that considers the 1-ring neighborhood $\mathcal{N}_s$ of a subvolume s, thus effectively constraining global smoothness of the spherical harmonics:

$$E_{\text{diffuse}} = \sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{N}_s} (\ell_s - \ell_r)^2. \tag{5.12}$$

5.5 Joint Optimization of Geometry, Albedo, and Image Formation Model

One of the core ideas of our method is the joint optimization of the volumetric 3D reconstruction as well as the image formation model. In particular, we simultaneously optimize for the signed distance and albedo values of each voxel of the volumetric grid, as well as the camera poses and camera intrinsics such as focal length, center pixel, and (radial and tangential) lens distortion coefficients. We stack all parameters in the unknown vector $\mathcal{X} = (\mathcal{T}, \tilde{\mathbf{D}}, \mathbf{a}, f_x, f_y, c_x, c_y, \kappa_1, \kappa_2, \rho_1)$ and formulate our minimization objective as follows:

$$E_{\text{scene}}(\mathcal{X}) = \sum_{\boldsymbol{v} \in \tilde{\mathbf{D}}_0} \lambda_g E_g + \lambda_v E_v + \lambda_s E_s + \lambda_a E_a, \tag{5.13}$$

with λ_g, λ_v, λ_s, λ_a the weighting parameters that define the influence of each cost term. For efficiency, we only optimize voxels within a thin shell close to the current estimate of the iso-surface $\tilde{\mathbf{D}}_0$, i.e., $|\tilde{\mathbf{D}}| < t_{\text{shell}}$.

5.5.1 Camera Poses and Camera Intrinsics

For initial pose estimates, we use poses obtained by the frame-to-model tracking of Voxel Hashing [86]. However, this merely serves as an initialization of the non-convex energy landscape for our global pose optimization, which is performed jointly along with the scene reconstruction (see below). In order to define the underlying residuals of the energy term, we project each voxel into its associated input views by using the current state of the estimated camera parameters. These parameters involve not only the extrinsic poses, but also the pinhole camera settings defined by focal length, pixel center, and lens distortion parameters. During the coarse-to-fine pyramid optimization, we derive the camera intrinsics according to the resolution of the corresponding pyramid levels.

5.5.2 Shading-based SDF Optimization

In order to optimize for the 3D surface that best explains the re-projection and follows the RGB shading cues, we directly solve for the parameters of the refined signed distance field $\tilde{\mathbf{D}}$, which is directly coupled to the shading through its surface normals $\mathbf{n}(\boldsymbol{v})$. In addition to the distance values, the volumetric grid also contains per-voxel albedo parameters, which again is coupled with the lighting computation (cf. Equation 5.9); the surface albedo is initialized with a uniform constant value. Although this definition of solving for a distance field follows the direction of Zollhöfer et al. [135], it is different at

its core: here, we dynamically constrain the reconstruction with the RGB input images, which contrasts Zollhöfer et al. who simply rely on the initially pre-computed per-voxel colors. In the following, we introduce all terms of the shading-based SDF objective.

Gradient-based Shading Constraint In our data term, we want to maximize the consistency between the estimated shading of a voxel and its sampled observations in the corresponding intensity images. Our objective follows the intuition that high-frequency changes in the surface geometry result in shading cues in the input RGB images, while more accurate geometry and a more accurate scene formation model result in better sampling of input images.

We first collect all observations in which the iso-surface point $\psi(\boldsymbol{v})$ of a voxel $\boldsymbol{v}$ is visible; we therefore transform the voxel into each frame using the pose $\mathcal{T}_i$ and check whether the sampled depth value in the respective depth map $\mathcal{Z}_i$ is compatible. We collect all valid observations $\mathcal{O}_{\boldsymbol{v}}$, sort them according to their weights $w_i^{\boldsymbol{v}}$ (cf. Equation (5.7)), and keep only the best t_{best} views $\mathcal{V}_{\text{best}} = \{\mathcal{I}_i\}$. Our objective function is defined as follows:

$$E_g(\boldsymbol{v}) = \sum_{\mathcal{I}_i \in \mathcal{V}_{\text{best}}} w_i^{\boldsymbol{v}} \|\nabla \mathbf{B}(\boldsymbol{v}) - \nabla \mathcal{I}_i(\pi(v_i))\|_2^2, \tag{5.14}$$

where $v_i = g(\mathcal{T}_i, \psi(\boldsymbol{v}))$ is the 3D position of the voxel center transformed into the view's coordinate system. Observations are weighted with their view-dependent observation weights $w_i^{\boldsymbol{v}}$. By transforming and projecting a voxel $\boldsymbol{v}$ into its associated input intensity images $\mathcal{I}_i$, our joint optimization framework optimizes for all parameters of the scene formation model, including camera poses, camera intrinsics, and lens distortion parameters. The shading $\mathbf{B}(\boldsymbol{v})$ depends on both surface and material parameters and allows to optimize for signed distances, implicitly using the surface normals, and voxel albedo on-the-fly. Instead of comparing shading and intensities directly, we achieve improved robustness by comparing their gradients, which we obtain by discrete forward differences from its neighboring voxels.

To improve convergence, we compute an image pyramid of the input intensity images and run the optimization in a coarse-to-fine manner for all levels. This inner loop is embedded into a coarse-to-fine grid optimization strategy, that increases the resolution of the SDF with each level.

Regularization We add multiple cost terms to regularize our energy formulation required for the ill-posed problem of Shape-from-Shading and to mitigate the effect of noise.

First, we use a Laplacian smoothness term to regularize our signed distance field. This volumetric regularizer enforces smoothness in the distance values between neighboring voxels:

$$E_v(\boldsymbol{v}) = (\Delta\tilde{\mathbf{D}}(\boldsymbol{v}))^2. \tag{5.15}$$

To constrain the surface and keep the refined reconstruction close to the regularized original signed distances, we specify a surface stabilization constraint:

$$E_s(\boldsymbol{v}) = (\tilde{\mathbf{D}}(\boldsymbol{v}) - \mathbf{D}(\boldsymbol{v}))^2. \tag{5.16}$$

Given spherical harmonics coefficients, the shading computed at a voxel depends on both its albedo as well as its surface normal. We constrain to which degree the albedo or normal should be refined by introducing an additional term that regularizes the albedo. In particular, the 1-ring neighborhood $\mathcal{N}_{\boldsymbol{v}}$ of a voxel is used to constrain albedo changes based on the chromaticity differences of two neighboring voxels. This follows the idea that chromaticity changes often go along with changes of intrinsic material:

$$E_a(\boldsymbol{v}) = \sum_{\boldsymbol{u}\in\mathcal{N}_{\boldsymbol{v}}} \phi(\boldsymbol{\Gamma}(\boldsymbol{v}) - \boldsymbol{\Gamma}(\boldsymbol{u})) \cdot (\mathbf{a}(\boldsymbol{v}) - \mathbf{a}(\boldsymbol{u}))^2, \tag{5.17}$$

where the voxel chromaticity $\boldsymbol{\Gamma} = \mathbf{C}(\boldsymbol{v})/\mathbf{I}(\boldsymbol{v})$ is directly computed from the voxel colors and $\phi(x)$ is a robust kernel with $\phi(x) = 1/(1 + t_{\text{rob}} \cdot x)^3$.

5.5.3 Joint Optimization Problem

We jointly solve for all unknown scene parameters stacked in the unknown vector $\mathcal{X}$ by minimizing the proposed highly non-linear least squares objective:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} E_{\text{scene}}(\mathcal{X}) \tag{5.18}$$

We solve the optimization using the well-known *Ceres Solver* [10], which provides automatic differentiation and an efficient Levenberg-Marquardt implementation.

By jointly refining the SDF and image formation model, we implicitly obtain optimal colors for the reconstruction at minimal re-projection error. In the optimization, the color and shading constraints are directly expressed with respect to associated input images; however, for the final mesh generation, we recompute voxel colors in a postprocess after the optimization. Finally, we extract a mesh from the refined signed distance field using Marching Cubes [75].

Dataset	# frames	# keyframes	Resolution	
			color	depth
Fountain [133]	1086	55	1280x1024	640x480
Lucy [135]	100	20	640x480	640x480
Relief [135]	40	8	1280x1024	640x480
Lion	515	26	1296x968	640x480
Tomb Statuary	523	27	1296x968	640x480
Bricks	773	39	1296x968	640x480
Hieroglyphics	919	46	1296x968	640x480
Gate	1213	61	1296x968	640x480

Table 5.1: Test RGB-D datasets used for the evaluation.

5.6 Results

We evaluated our approach on publicly available RGB-D datasets as well as on own datasets acquired using a Structure Sensor; Table 5.1 gives an overview. For *Lucy* and *Relief* we used the camera poses provided with the datasets as initializations, while we estimated the poses using Voxel Hashing [86] for all other datasets. Our evaluations were performed on a workstation with Intel Core i7-5930 CPU with 3.50GHz and 32GB RAM.

We used $\lambda_{\text{diffuse}} = 0.01, \lambda_g = 0.2, \lambda_v = 160 \rightarrow 20, \lambda_s = 120 \rightarrow 10, \lambda_a = 0.1$ for our evaluations, with $a \rightarrow b$ indicating changing weights with every iteration. For objects with constant albedo, we fixed the albedo; i.e., we set $\lambda_a = \infty$. We used three RGB-D frame pyramid levels and three grid levels, such that the finest grid level has a resolution of 0.5mm (or 1.0mm, depending on object size). We set $t_{\text{best}} = 5$ to limit the number of data term residuals per voxel. To reduce the increase of the number of voxels close to the surface considered for optimization, we used an adaptive thin shell size t_{shell}, linearly decreasing from $2.0 \rightarrow 1.0$ times the voxel size with each grid pyramid level.

Appearance Using our method, we implicitly obtain optimal voxel colors as a consequence of the joint optimization of intrinsic material properties, surface geometry and image formation model. Figure 5.4 shows qualitative results from the *Fountain* dataset. While volumetric blending [84, 86] produces blurry colors, camera poses are corrected in advance by Zollhöfer et al. [135] using dense bundle adjustment to yield significantly better color and geometry. However, their static color integration cannot correct for small inaccuracies, resulting in slightly blurry colors. In contrast, our method adjusts the surface and image formation model jointly to produce highly detailed texture at the same voxel grid resolution of 1mm. Within our joint optimization, we also estimate varying albedo. Figure 5.7 shows the estimated albedo for the *Fountain* dataset.

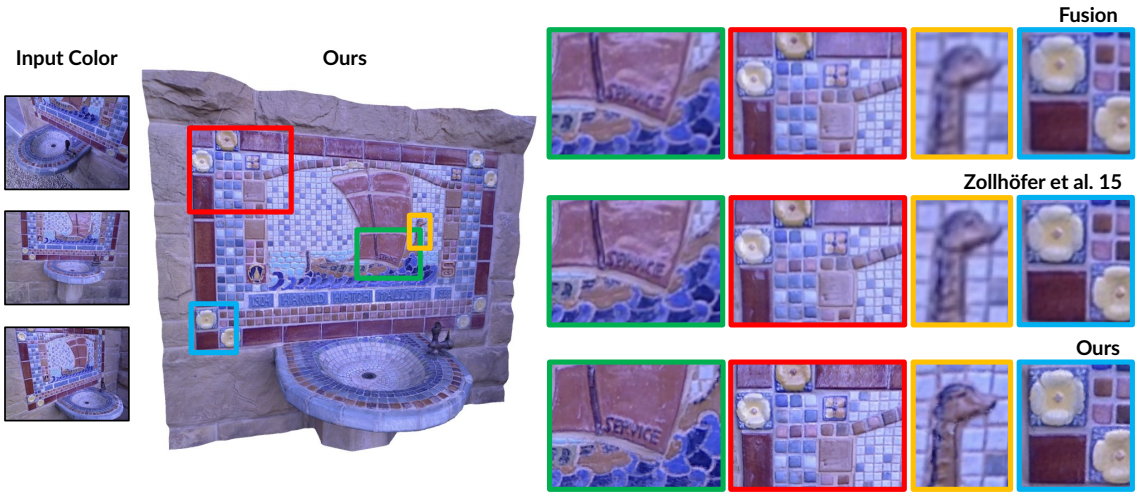


Figure 5.4: Appearance of the *Fountain* reconstruction. Our method shows a visually more appealing result compared to volumetric fusion and Zollhöfer et al. [135].

Surface Geometry We qualitatively compare the quality of refined surfaces using our method with the approach of Zollhöfer et al. [135] in Figure 5.5. The results of the *Relief* dataset visualize that our method reveals finer geometric details by directly sampling from high-resolution input color images instead of using averaged voxel colors. Moreover, we benefit from simultaneously optimizing for camera poses and camera intrinsics.

Additionally, we provide a quantitative ground truth evaluation of the geometry refinement on the synthetic *Frog* RGB-D dataset, which was generated by rendering a ground truth mesh with a high level of detail into synthetic color and depth images. Both depth and camera poses were perturbed with realistic noise. Figure 5.6 shows that, in contrast to fusion and [135], our method is able to reveal even smaller details. Quantitatively, the mean absolute deviation (MAD) between our reconstruction and the ground truth mesh is 0.222mm (with a standard deviation of 0.269mm), while the reconstruction generated using our implementation of [135] results in a higher error of 0.278mm (with a standard deviation of 0.299mm). This corresponds to an overall accuracy improvement of 20.14% of our method compared to [135]. We refer the reader to the supplementary material for a quantitative evaluation on real data and further results.

Lighting In the following, we evaluate lighting estimation via spatially-varying spherical harmonics, both qualitatively and quantitatively. In particular, we demonstrate that a single global set of SH coefficients cannot accurately reflect real-world environments with complex lighting. To analyze the effects of the illumination, we re-light the reconstruction using the surface normals and estimated voxel albedo according to Equation (5.9). The computed shading $\mathbf{B}(\boldsymbol{v})$ of a voxel is in the ideal case identical to the

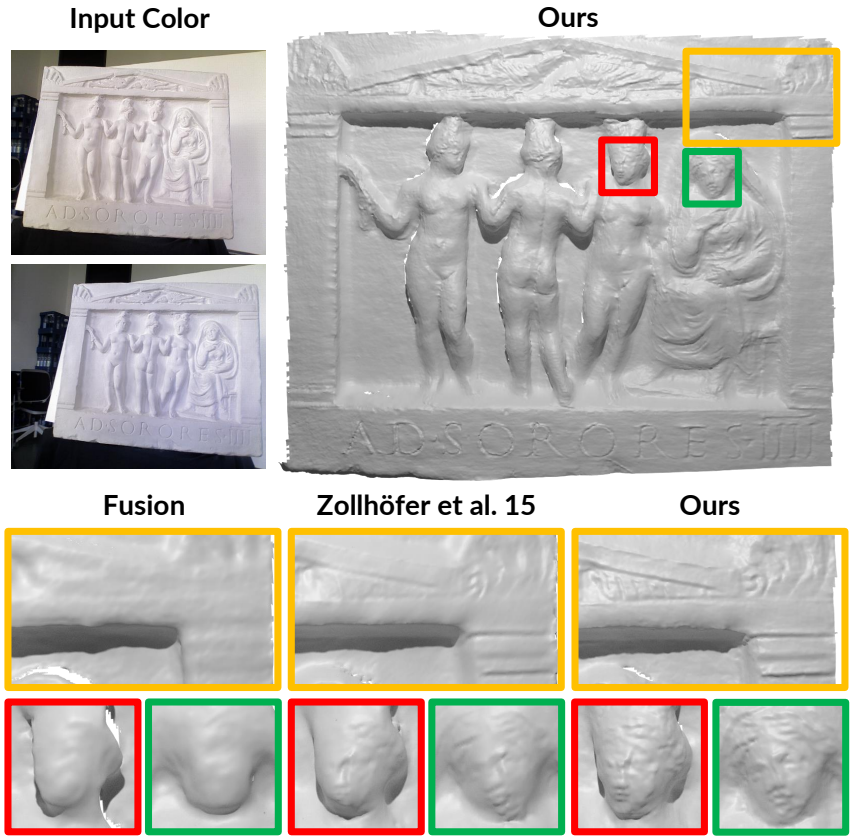


Figure 5.5: Comparison of the reconstructed geometry of the *Relief* dataset. Our method (right) reveals finer geometric details compared to volumetric fusion (left) and Zollhöfer et al. [135] (middle).

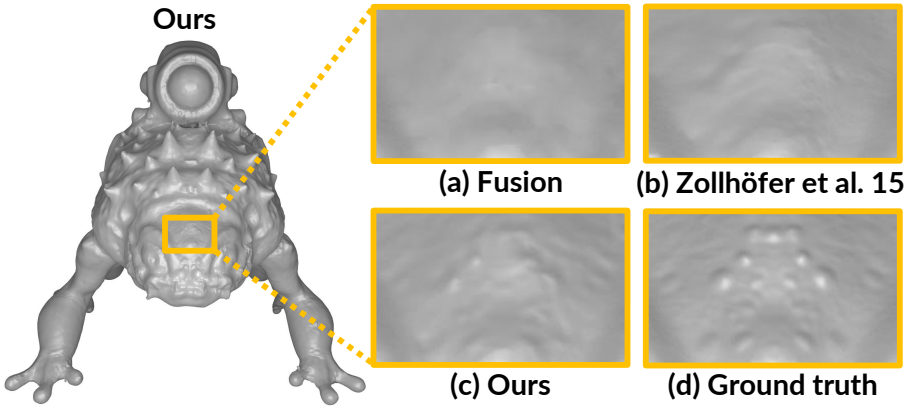


Figure 5.6: Refined geometry of the *Frog* dataset: while fusion (a) smooths out high-frequency details, Zollhöfer et al. [135] (b) can reconstruct some geometric details. Our method (c) recovers even smaller surface details present in the ground truth mesh (d).

Dataset	Global SH	SVSH (subvolume size)			
		0.5	0.2	0.1	0.05
Fountain	22.973	18.831	15.891	13.193	**10.263**
Lucy	22.190	19.408	16.564	14.141	**11.863**
Relief	13.818	12.432	11.121	9.454	**8.339**
Lion	30.895	25.775	20.811	16.243	**13.468**
Tomb Statuary	33.716	30.873	30.639	29.675	**26.433**
Bricks	29.327	27.110	25.318	22.850	**19.476**
Hieroglyphics	15.710	15.206	11.140	12.448	**9.998**
Gate	46.463	40.104	33.045	20.176	**12.947**

Table 5.2: Quantitative evaluation of spatially-varying spherical harmonics. The Mean Absolute Deviation (MAD) between averaged per-voxel intensity and estimated shading decreases with decreasing subvolume sizes.

measured voxel intensity $\mathbf{I}(\boldsymbol{v})$ computed from the voxel color.

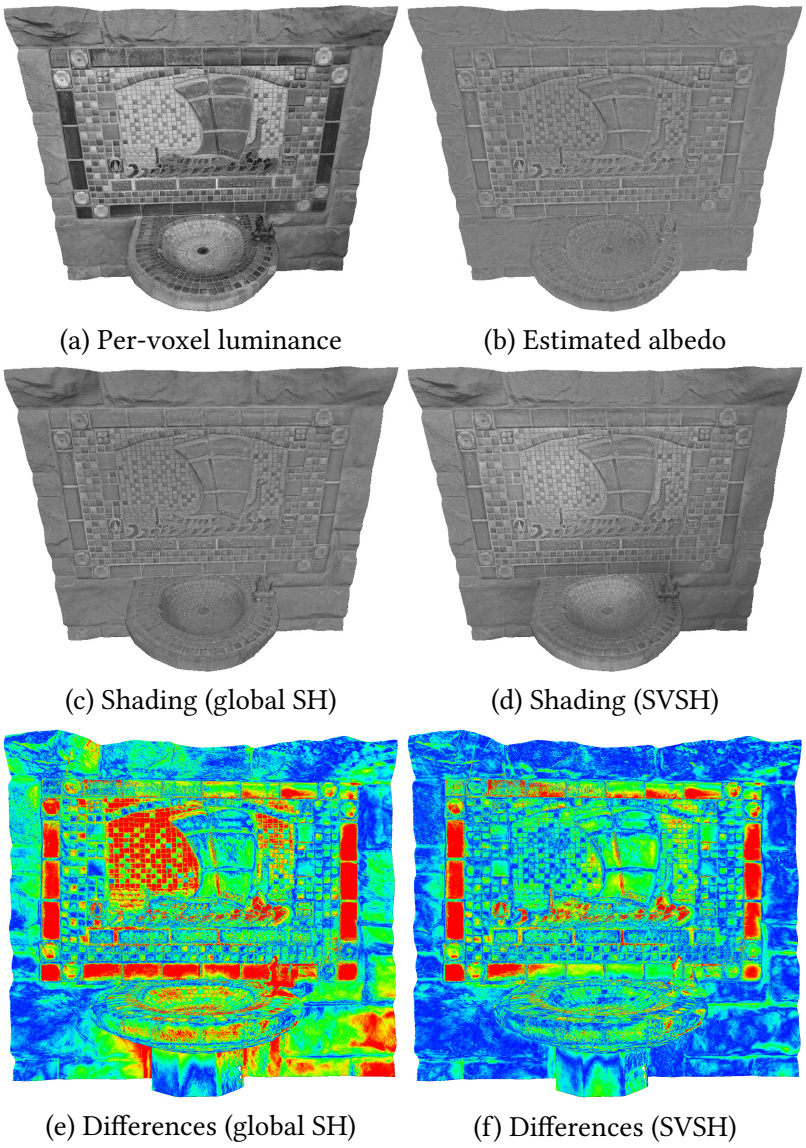
We exploit the absolute difference $|\mathbf{B}(\boldsymbol{v}) - \mathbf{I}(\boldsymbol{v})|$ as an error metric in order to quantitatively evaluate the quality of the illumination for given geometry and albedo. In particular, we measure the mean absolute deviation (MAD) for all N voxels of the SDF volume:

$$\epsilon_{\text{shading}} = \frac{1}{N} \sum_{\boldsymbol{v} \in \mathbf{D}} |\mathbf{B}(\boldsymbol{v}) - \mathbf{I}(\boldsymbol{v})| \tag{5.19}$$

Table 5.2 gives the results of global SH coefficients and SVSH with varying subvolume sizes for multiple datasets. In summary, the more the SDF volume is partitioned into subvolumes, the better the approximation to complex lighting scenarios. The illumination in the *Fountain* dataset is clearly spatially varying, violating the assumptions of distant and spatially invariant illumination for SH lighting coefficients. Figure 5.7 shows that the estimated shading is better approximated with SVSH coefficients compared to only with global SH coefficients, while the underlying surface and albedo are exactly the same for both shadings.

5.7 Conclusion

We have presented a novel method for simultaneous optimization of scene reconstruction along with the image formation model. This way, we obtain high-quality reconstructions along with well-aligned sharp surface textures using commodity RGB-D sensors by efficiently combining information from (potentially noisy) depth and (possibly) higher resolution RGB data. In comparison to existing Shape-from-Shading techniques (e.g., [126, 135]), we tackle the core problem of fixing wrong depth measurements jointly with pose alignment and intrinsic scene parameters. Hence, we minimize re-projection



(a) Per-voxel luminance (b) Estimated albedo

(c) Shading (global SH) (d) Shading (SVSH)

(e) Differences (global SH) (f) Differences (SVSH)

Figure 5.7: Quantitative evaluation of global SH vs. SVSH: the heatmaps in (e) and (f) represent the differences between the per-voxel input luminance (a) and the shadings with global SH (c) and with SVSH (d), both with underlying albedo (b).

errors, thus avoiding oversmoothed geometry and blurry surface textures. In addition, we introduce a significantly more flexible lighting model that is spatially-adaptive, thus allowing for a more precise estimation of the scene lighting.

Acknowledgment We would like to thank Qian-Yi Zhou and Vladlen Koltun for the *Fountain* data and Michael Zollhöfer for the *Socrates* laser scan. This work was partially funded by the ERC Consolidator grant *3D Reloaded*.

Chapter 6

Efficient Online Surface Correction for Real-time Large-Scale 3D Reconstruction

Authors:	Robert Maier*[1]	robert.maier@tum.de
	Raphael Schaller*[1]	schaller@in.tum.de
	Daniel Cremers[1]	cremers@tum.de

* Equal contribution
[1] Technische Universität München, Germany

Conference: *British Machine Vision Conference (BMVC) 2017.*
DOI: 10.5244/C.31.158.

Abstract State-of-the-art methods for large-scale 3D reconstruction from RGB-D sensors usually reduce drift in camera tracking by globally optimizing the estimated camera poses in real-time without simultaneously updating the reconstructed surface on pose changes. We propose an efficient on-the-fly surface correction method for globally consistent dense 3D reconstruction of large-scale scenes. Our approach uses a dense Visual RGB-D SLAM system that estimates the camera motion in real-time on a CPU and refines it in a global pose graph optimization. Consecutive RGB-D frames are locally fused into keyframes, which are incorporated into a sparse voxel hashed Signed Distance Field (SDF) on the GPU. On pose graph updates, the SDF volume is corrected on-the-fly using a novel keyframe re-integration strategy with reduced GPU-host streaming. We demonstrate in an extensive quantitative evaluation that our method is up to 93% more runtime efficient compared to the state-of-the-art and requires significantly less memory, with only negligible loss of surface quality. Overall, our system requires only a single GPU and allows for real-time surface correction of large environments.

Revised layout and minor adaptations. Original publication [5] and detailed disclaimer are included in Chapter C.3 in the appendix.

6.1 Introduction

In recent years, there has been a boost of research in the field of dense 3D reconstruction due to the wide availability of low-cost depth sensors such as the Microsoft Kinect. Most of the approaches fuse depth maps obtained from such sensors in real-time into a volumetric surface representation [26] to compensate for sensor noise and perform frame-to-model camera tracking against the fused volume. While researchers have shown the suitability of these methods for accurate geometric reconstruction of objects or scenes of limited size [84], global drift in camera tracking is not compensated, limiting the reconstruction of large-scale environments [53, 86, 120].

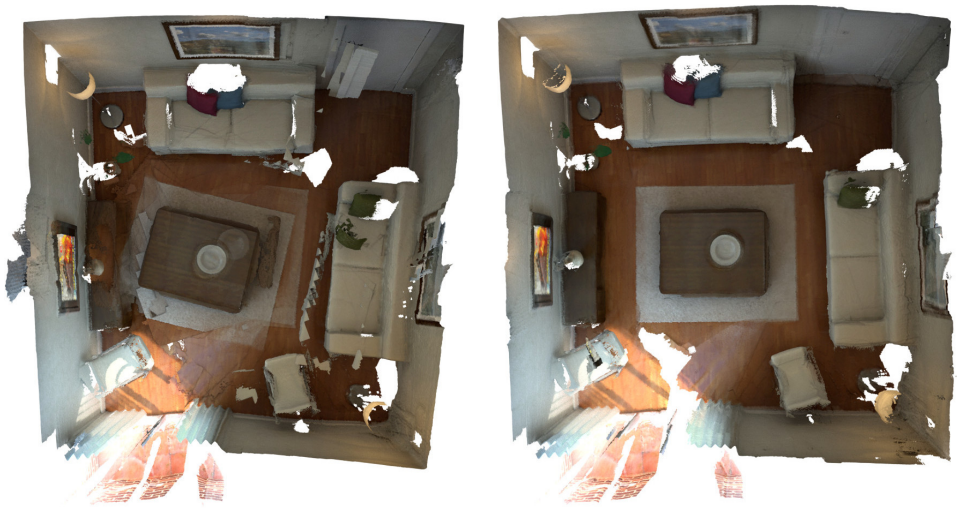
However, only few methods tackle the problem of globally optimizing the camera poses in real-time and simultaneously correcting the reconstructed surface on-the-fly. BundleFusion by Dai et al.[28] represents the state-of-the-art and estimates highly accurate camera poses on a high-end GPU. They require a second graphics card for integrating input RGB-D frames into a sparse Signed Distance Field (SDF) volume, making the entire framework computationally demanding. On pose graph updates, BundleFusion corrects the reconstructed surface on-the-fly by frame re-integration. However, all previous frames need to be held in memory to allow for a fast re-integration on pose updates; this limits its suitability for scanning large-scale environments with long sequences.

To enable state-of-the-art large-scale 3D reconstruction from RGB-D sensors, our SLAM framework is based on DVO-SLAM by Kerl et al. [57] for estimating a globally consistent camera motion. The system is computationally significantly less expensive than BundleFusion and works in real-time on a single CPU, with only slightly less accurate estimated camera poses. To obtain globally consistent and up-to-date reconstructions of large environments, we couple it with our novel 3D surface correction method. Figure 6.1 shows the result of our online surface re-integration method at the end of a 3D scanning session and indicates the effect of keyframe fusion on the completeness of the reconstruction.

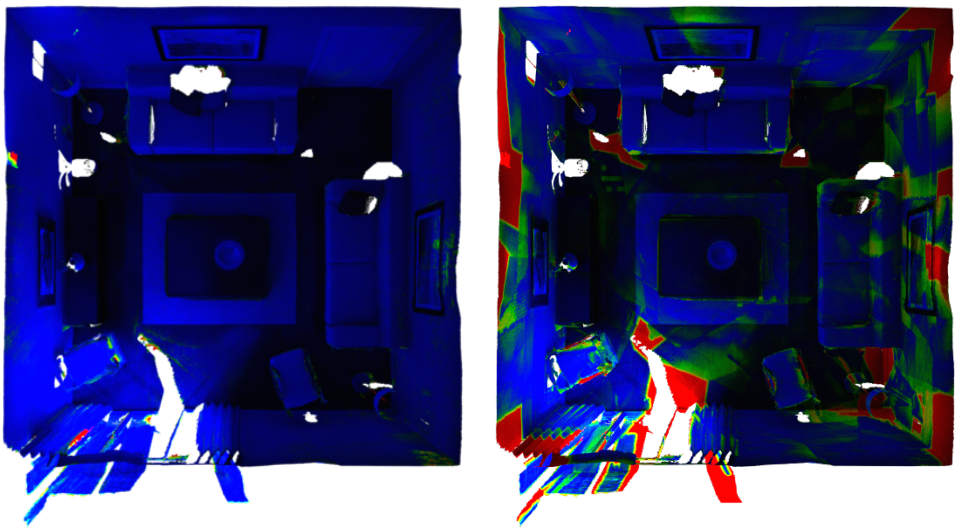
In summary, the main contributions of our work are:

- We integrate our 3D surface correction framework with a dense Visual SLAM system, such that our entire 3D reconstruction system runs in real-time with a single GPU only.
- We fuse consecutive RGB-D input frames in keyframes of high depth and color quality using different keyframe strategies.
- Our surface correction framework is highly efficient by only re-integrating fused keyframes into a sparse SDF volume on pose graph updates.

- Our strategy for selecting keyframes to be updated substantially reduces streaming between host and GPU.
- An extensive quantitative evaluation shows that our method is overall 93% more efficient compared to the state-of-the-art while maintaining surface quality.



a) Without and with on-the-fly surface correction



b) Surface completeness (5 vs. 60 frames per keyframe)

Figure 6.1: Our method efficiently corrects the surface during the 3D scanning process on-the-fly (a) using an efficient keyframe re-integration strategy. Fusing fewer frames into each keyframe allows to maintain the completeness of the reconstructed 3D model (b).

6.2 Related Work

The field of dense 3D reconstruction from RGB-D data has been investigated extensively in recent years. KinectFusion by Newcombe et al. [84] enabled dense 3D reconstruction in real-time through extensive use of GPU programming. Like most of the following approaches, it stores the 3D model in an SDF volume [26], which regularizes the noisy depth maps from RGB-D sensors, and performs ICP-based camera tracking against the raycasted 3D model. Voxel Hashing [86] better exploits scarce GPU memory and allocates only occupied voxel blocks of the SDF. A hash map flexibly maps 3D voxel block coordinates onto memory locations. Kähler et al. [53] designed an optimized version of Voxel Hashing for mobile devices. However, the frame-to-model camera tracking of the frameworks above is only of limited use for reconstructing larger scenes. To reduce drift explicitly, recent approaches [23, 7, 106, 134] rely on loop closure detection in combination with global pose optimization.

In order to efficiently estimate camera poses in real-time, DVO-SLAM by Kerl et al. [57] minimizes a photometric and geometric error to accurately align RGB-D frames. For global consistency, it continuously performs a pose graph optimization to reduce global drift. While there is no dense volumetric model representation, they exploit keyframes to reduce the influence of noise. The system provides an excellent trade-off between runtime and accuracy, making it highly suitable for our 3D reconstruction framework. Utilizing keyframes as intermediate representation for reducing noise has also been exploited for improving camera tracking [79] and reconstruction appearance [6]. Following this idea, we also employ keyframes in our work as memory efficient intermediate 2.5D representations of 3D surfaces.

There are only few works on real-time large-scale RGB-D based 3D reconstruction that incorporate online surface correction. Fioraio et al. [33] reconstruct overlapping subvolumes, register their poses globally and update the subvolumes using volume blending. However, the absence of loop closure detection avoids to cope with larger drift. Kähler et al. [52] perform real-time tracking against multiple submaps independently and globally optimize the estimated trajectories. Submaps are fused on-the-fly during raycasting.

Whelan et al. use a deformation graph for online update of a surfel-based model [123] and of an SDF model [121] with an as-rigid-as-possible surface deformation. In ElasticFusion [123] input frames are fused into surfels and then discarded. However, wrong camera poses (e.g. due to drift) result in inconsistent surfel observations and hence increase their uncertainties; surfels with high uncertainty are ultimately filtered out. When a loop closure is detected, only the existing surface can be corrected along with the deformation graph, while surface information lost through inconsistent fusion cannot be

recovered. In contrast, our method keeps all keyframes fused from input data and allows to re-integrate them at any pose graph update without a loss of surface information. Additionally, despite correcting the model online, the frame-to-model camera tracking may fail to compensate for drift due to delayed surface updates and undetected (or too late detected) loop closures.

BundleFusion et al. [28] represents the state-of-the-art both w.r.t. SLAM system accuracy as well as on-the-fly surface re-integration. The system first optimizes consecutive frame poses locally within chunks, which are then aligned globally in a hierarchical global optimization. New RGB-D input frames are matched brute-force against all previous chunk keyframes and subsequently aligned using a sparse-then-dense alignment. The global alignment regularly changes camera poses; to correct the reconstructed sparse SDF volume on-the-fly, the system first de-integrates frames with their former poses and then integrates them with their updated poses using a simple re-integration strategy. The 3D model is gradually adapted to the updated poses while still enabling real-time reconstruction. In contrast to BundleFusion, our method needs only a single GPU for surface modeling instead of two high-end graphics cards. Our online surface re-integration combines keyframe fusion with a more intelligent keyframe selection strategy, resulting in a significantly more efficient re-integration. Moreover, the use of keyframes requires substantially less memory and enables on-the-fly surface correction for large environments.

6.3 3D Reconstruction System

Our framework consists of a real-time RGB-D SLAM framework for globally consistent camera pose estimation and a sparse SDF volume for storing the reconstructed 3D model. While dense SDF-based 3D reconstruction methods usually integrate new RGB-D input frames directly into the volume, we first fuse them into keyframes as intermediate data representation. We integrate and re-integrate them online into the SDF on pose updates to efficiently correct the surface. This way, we can reduce the number of individual frames that we integrate into the SDF volume, which helps especially when we need to correct the 3D model due to a pose update. Figure 6.2 shows an overview of our approach.

Preliminaries We acquire RGB-D data from commodity depth sensors with 30 fps at a resolution of 640×480 pixels. The N captured RGB-D frames consist of registered color images $\mathcal{C}_i$, depth maps $\mathcal{Z}_i$ and camera poses $\mathcal{T}_i = (\boldsymbol{R}_i, \boldsymbol{t}_i) \in \mathbb{SE}(3)$ (with $\boldsymbol{R}_i \in \mathbb{SO}(3)$, $\boldsymbol{t}_i \in \mathbb{R}^3$ and $i \in 1 \dots N$). A 3D point $\boldsymbol{p} = (X, Y, Z)^\top$ is transformed using a pose $\mathcal{T}_i$ through $g(\mathcal{T}_i, \boldsymbol{p}) = \boldsymbol{R}_i \boldsymbol{p} + \boldsymbol{t}_i$. We use the pinhole camera model, which projects 3D

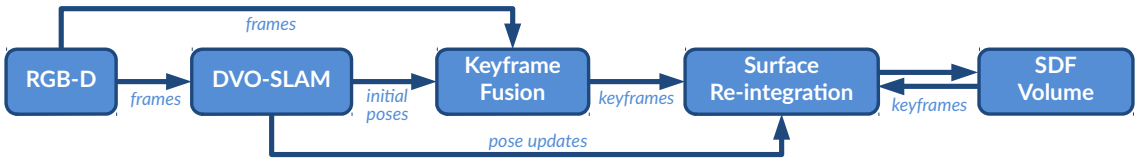


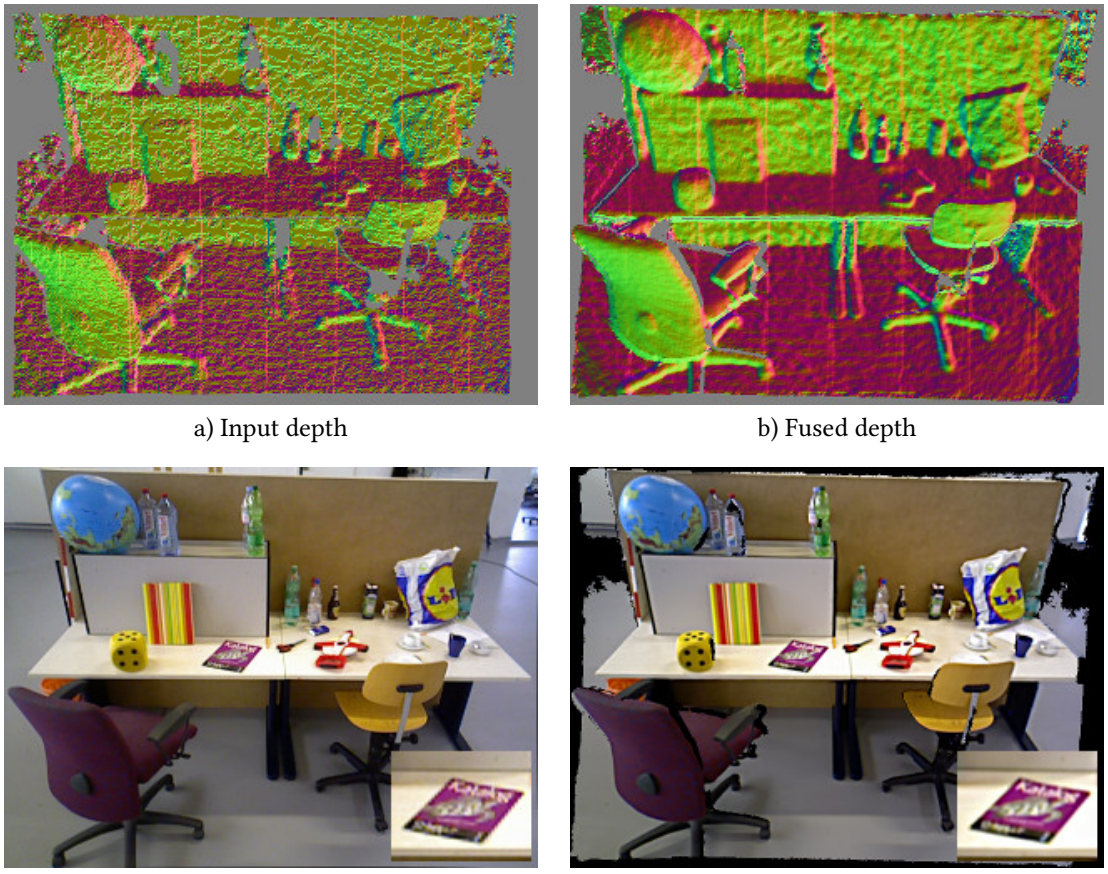
Figure 6.2: Overview of our online surface correction method. RGB-D frames are fused into keyframes, which are (re-)integrated into the SDF on-the-fly on DVO-SLAM pose updates.

points $\boldsymbol{p}$ to 2D pixels $\boldsymbol{x} = (u, v)^\top = \pi(\boldsymbol{p})$ using the projection $\pi : \mathbb{R}^3 \to \mathbb{R}^2$. The inverse projection $\pi^{-1} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$ maps a 2D pixel location $\boldsymbol{x}$ back to the respective 3D point $\boldsymbol{p} = \pi^{-1}(\boldsymbol{x}, \mathcal{Z}(\boldsymbol{x}))$ using its depth.

Dense Visual RGB-D SLAM To estimate globally consistent camera poses $\mathcal{T}_i$, we utilize the DVO-SLAM system by Kerl et al. [57]. It runs in real-time on a CPU and employs a robust dense visual odometry approach that minimizes the photometric and geometric error of all pixels to estimate the rigid body motion between two RGB-D frames. To reduce drift in camera pose estimation, input frames are aligned against the preceding keyframe. Keyframes are selected using the differential entropy of the motion estimate and a pose distance threshold. In the following, we refer to the keyframes selected by DVO-SLAM as *DVO keyframes*. DVO-SLAM detects loop closures by aligning keyframes against candidates of previous keyframes within a sphere of predefined radius and validates them using their entropy ratio. Estimated frame-to-(key)frame camera motions and successful loop closures are integrated as constraints into a graph based map representation. This keyframe pose graph is steadily optimized in the background during runtime, yielding a globally consistent camera trajectory with continuously updated keyframe poses $\mathcal{T}_i$. Please note that our surface correction method works in principle with any SLAM system that incorporates loop closures.

Keyframe Fusion Our keyframe fusion builds up on [6] and consists of separate steps for depth and color fusion (cf. Figure 6.3). While new depth maps are immediately fused into the keyframe depth, color fusion relies on the more complete fused keyframe depth.

For depth fusion, we first compute for each pixel $\boldsymbol{x}$ of $\mathcal{Z}_i$ its respective view- and distance-dependent weight $w_z(\boldsymbol{x}) = \cos(\theta_i(\boldsymbol{x})) \cdot \mathcal{Z}_i(\boldsymbol{x})^{-2}$, where $\theta_i(\boldsymbol{x})$ is the angle between the depth normal at $\boldsymbol{x}$ and the camera axis. Furthermore, we discard error-prone depth values close to depth discontinuities. We then warp each pixel with the frame pose $\mathcal{T}_i$ into the keyframe with pose $\mathcal{T}^*$ and obtain $\boldsymbol{p}^* = (X^*, Y^*, Z^*)^\top = g(\mathcal{T}^{*-1}, g(\mathcal{T}_i, \pi^{-1}(\boldsymbol{x}, \mathcal{Z}_i(\boldsymbol{x}))))$. The keyframe depth $\mathcal{Z}^*$ and the depth fusion weights $\mathcal{W}^*$



a) Input depth b) Fused depth

c) Input color d) Fused color

Figure 6.3: Keyframe fusion: several consecutive input depth maps (a) are fused into the keyframe depth (b). Our color fusion creates sharp color keyframes (d) from input color (c).

at the projected 2D image point $\boldsymbol{x}^* = \pi(\boldsymbol{p}^*)$ are then updated as follows:

$$\mathcal{W}^*(\boldsymbol{x}^*) = \frac{\mathcal{W}^*(\boldsymbol{x}^*)\mathcal{Z}^*(\boldsymbol{x}^*) + w_z(\boldsymbol{x})Z^*}{\mathcal{W}^*(\boldsymbol{x}^*) + w_z(\boldsymbol{x})}, \qquad \mathcal{W}^*(\boldsymbol{x}^*) = \mathcal{W}^*(\boldsymbol{x}^*) + w_z(\boldsymbol{x}). \tag{6.1}$$

For color fusion, we first deblur the input color images using *Unsharp Masking* and compute a per-frame blurriness measure w_b [25] from $\mathcal{C}_i$ to alleviate frames with strong motion blur. In contrast to depth fusion, the fused keyframes are warped back into each input frame i and the observed color values $c_i(\boldsymbol{x}^*) = \mathcal{C}_i(\pi(g(\mathcal{T}_i^{-1}, g(\mathcal{T}^*, \pi^{-1}(\boldsymbol{x}^*, \mathcal{Z}^*(\boldsymbol{x}^*))))))$ are sampled using bilinear interpolation. For a pixel $\boldsymbol{x}^*$ in the keyframe, we collect all valid color observations in the input views and their color weights $w_{c_i}(\boldsymbol{x}^*) = w_b \cdot w_z$. We compute the final keyframe color $\mathcal{C}^*(\boldsymbol{x}^*)$ as the weighted median of the collected observations.

SDF volume At the core of our method, we store a memory efficient sparse SDF volume based on Voxel Hashing [86] as volumetric 3D model representation for large-scale 3D reconstructions. The implemented data structure is tailored to GPUs and only occupied space is allocated in voxel blocks, which are efficiently addressed using spatial hashing. For each voxel $\boldsymbol{v}$, we store its signed distance $\mathbf{D}(\boldsymbol{v})$, its color $\mathbf{C}(\boldsymbol{v})$ and its integration weight $\mathbf{W}(\boldsymbol{v})$. We extract the iso-surface from the SDF using Marching Cubes [75]. To overcome the limitations of scarce GPU memory for large-scale environments, voxel blocks are streamed from GPU to host (and vice versa) before integration of a new frame. In particular, only voxel blocks within a sphere of constant radius around the current camera position are kept in GPU memory, while all other voxel blocks are streamed to the host. When an RGB-D frame is integrated into the SDF volume, voxel blocks are first allocated and the voxels are then updated using a running weighted average.

6.4 Efficient Online Surface Re-Integration

In the following, we introduce our online surface correction method that combines keyframe fusion with our sparse SDF volume implementation. Firstly, we incorporate on-the-fly keyframe re-integration into the 3D reconstruction pipeline; secondly, we show different strategies for starting new keyframes; thirdly, we propose an efficient surface correction procedure that is based on a re-integration strategy that reduces GPU-host transfer.

6.4.1 System Pipeline

While DVO-SLAM selects *DVO keyframes* for camera tracking based on an entropy criteria, we introduce *KF keyframes* (Keyframe Fusion keyframes) as intermediate representation for surface (re-)integration. When a new frame arrives, DVO-SLAM provides an initial pose estimate which is used to fuse the input frame into the current KF keyframe. Depending on the chosen keyframe selection strategy, a new keyframe will be started if some criteria are met and the previous KF keyframe is integrated into the SDF volume. The KF keyframe is also stored in memory for later re-integration on pose updates. Since DVO-SLAM issues only pose updates for DVO keyframes, we convert by expressing KF keyframe poses relative to DVO keyframe poses. The KF keyframes are then de-integrated from the SDF volume with their former camera poses and re-integrated on-the-fly with their updated poses.

6.4.2 Keyframe Strategies

In the following, we investigate keyframe selection strategies w.r.t. obtaining optimal surface quality. With only few input frames fused into KF keyframes, many KF keyframes need to be (re-)integrated into the SDF volume on pose updates. On the other hand, fusing many input frames into KF keyframes leads to a degradation in 3D reconstruction quality, since the 2.5D keyframes cannot fully represent the incorporated 3D information. We present keyframe strategies to find the optimal trade-off between re-integration performance and reconstruction quality.

The KF_CONST strategy is a simple but effective strategy and fuses a constant number κ of frames into each KF keyframe. KF_DVO uses the frames selected as DVO keyframes also as KF keyframes. The distance based strategy KF_DIST issues a new KF keyframe whenever the rotational distance Δ_r or translational distance Δ_t of the relative pose $\mathcal{T}_{ij}$ between the current frame and the current KF keyframe exceeds a certain threshold, similar to [61]. The overlap strategy KF_OVRLP is derived from [45] and generates a new KF keyframe when the ratio of the pixels visible in both current frame and keyframe drops below a threshold.

6.4.3 On-the-fly Surface Correction

Our surface correction method follows the frame re-integration approach of [28]. However, we substantially improve it at critical points w.r.t. runtime efficiency by implementing a more intelligent strategy for selecting the KF keyframes to be re-integrated. Since we only need to correct KF keyframes instead of all frames, our surface correction is highly efficient w.r.t. runtime and memory consumption.

Frame de-integration For de-integrating an RGB-D frame i from the SDF volume, we simply reverse the integration procedure. We therefore retrieve the KF keyframe from the memory and compute the projective distance d_i (along the z axis) of $\boldsymbol{v}$ in depth map $\mathcal{Z}_i$ (with sampling weight w_i) and its sampled color c_i in the input color image $\mathcal{C}_i$. The de-integration steps for updating signed distance, color and weight of a voxel are denoted as follows:

$$\mathbf{D}'(\boldsymbol{v}) = \frac{\mathbf{D}(\boldsymbol{v})\mathbf{W}(\boldsymbol{v}) - d_i w_i}{\mathbf{W}(\boldsymbol{v}) - w_i}, \mathbf{C}'(\boldsymbol{v}) = \frac{\mathbf{C}(\boldsymbol{v})\mathbf{W}(\boldsymbol{v}) - c_i w_i}{\mathbf{W}(\boldsymbol{v}) - w_i}, \mathbf{W}'(\boldsymbol{v}) = \mathbf{W}(\boldsymbol{v}) - w_i \quad (6.2)$$

Re-integration strategy While the poses of all keyframes are updated when DVO-SLAM issues a pose update, it is computationally too expensive to correct them all immediately. Instead, we re-integrate only m changed frames whenever we receive a pose update. Poses that were not corrected on-the-fly are re-integrated in a final pass

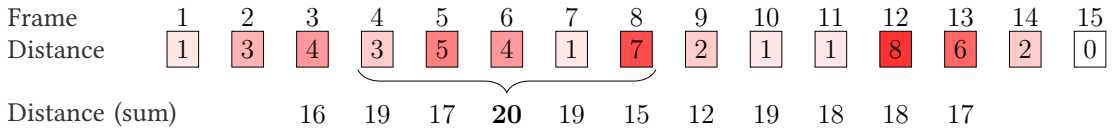


Figure 6.4: Selection of frames for re-integration: BundleFusion [28] chooses the frames with highest distances between integrated pose and new pose. However, selecting frames $12, 8, 13, 5, 3$ results here in disadvantageous shifts of the streaming sphere. Our method selects the group of most-moved m consecutive frames, which results in frames 4 to 8 ($j^* = 4$).

after the reconstruction. We denote the SDF integration pose of a frame by $\mathcal{T}_i$ and the updated pose by $\mathcal{T}_i'$.

To select the m frames for re-integration, BundleFusion orders all frames by descending distance between $\mathcal{T}_i$ and $\mathcal{T}_i'$ $\|s\boldsymbol{t}_i - s\boldsymbol{t}_i'\|$ and selects the m most-moved frames. The vectors $\boldsymbol{t}_i$ and $\boldsymbol{t}_i'$ contain the Euler rotation angles and the translation of the poses $\mathcal{T}_i$ and $\mathcal{T}_i'$, with a constant scale vector $s = (2, 2, 2, 1, 1, 1)^\top$. However, since the corrected frames (and the respective SDF voxel blocks) may be spatially distant, suboptimal expensive GPU-host-streaming of voxel blocks may be required. To limit the streaming overhead, it is beneficial to correct close frames within the same re-integration procedure. We therefore keep the original temporal ordering of frames and select the group of most-moved m *consecutive* frames:

$$j^* = \underset{j \in [1, K-m+1]}{\arg\max} \sum_{i=j}^{j+m-1} \|s\boldsymbol{t}_i - s\boldsymbol{t}_i'\|\,, \tag{6.3}$$

where K is the total number of frames integrated so far. The resulting j^* represents the first frame of our m consecutive frames thats need to be re-integrated. Figure 6.4 exemplifies the advantages of this procedure. Additionally, we adjust the streaming procedure of [86] to the re-integration process: We first stream in all voxel blocks inside the sphere around pose $\mathcal{T}_{j^*}$ to safely access them. Then, we successively de-integrate frames $[j^*, j^* + m - 1]$ with regular streaming. After de-integration, we stream in the sphere around the updated pose $\mathcal{T}_{j^*}'$ und successively re-integrate frames $[j^*, j^* + m - 1]$ using their updated poses with regular streaming. We finally stream the sphere back to the next integration pose.

6.5 Evaluation and Experimental Results

To demonstrate the effectiveness of our surface reconstruction algorithm, we provide a thorough quantitative evaluation w.r.t. runtime efficiency and surface accuracy. In particular, we analyze the effects of combining keyframe fusion with our surface re-integration method.

Datasets We use publicly available RGB-D datasets of large-scale scenes with loop closures that provide registered depth and color images as well as the respective camera poses. *AUG_ICL/Liv1* (noisy) [23] is a synthetic RGB-D sequence that is rendered from a modeled scene of a living room with realistic sensor noise. In addition to ground truth poses it also provides the ground truth 3D scene model that allows for a quantitative comparison of surface quality of reconstructed 3D models. *BundleFusion/apt0* [28] features a long camera trajectory of 8560 frames with poses estimated from BundleFusion.

Surface evaluation methods and metrics The evaluation procedure for comparing our reconstructed 3D models with synthetic ground truth is adapted from [42] and first extracts a 3D mesh $\mathcal{M}$ from the reconstructed SDF volume. We use CloudCompare [1] to uniformly sample a reference point cloud $\mathcal{R}$ with 50 million points from the ground truth mesh of *AUG_ICL/Liv1*. We measure the distance of each vertex of $\mathcal{M}$ to its closest vertex in $\mathcal{R}$ with SurfReg [2] and compute the mean absolute deviation MAD. This technique assesses the *correctness* CORR of the model, i.e. the accuracy of the successfully reconstructed surfaces. However, we also want to measure the *completeness* COMPL of reconstructions to determine the information loss from keyframe fusion. For measuring COMPL, we inversely compare every vertex of $\mathcal{R}$ to the nearest neighbor in $\mathcal{M}$. For a fair comparison and to only compare surfaces visible in the synthetic frames, we re-generate the reference $\mathcal{R}$ by fusing all input frames into the SDF with ground truth poses. We rely on the poses from the datasets for assessing the surface quality to eliminate a substantial source of error. We used a workstation with Intel Core i7-3770 CPU, 32GB RAM and an NVIDIA GeForce GTX 1070 GPU.

6.5.1 Keyframe Fusion

We quantitatively investigate the effect of keyframe fusion on the reconstruction quality, i.e. surface completeness and correctness, of the noisy *AUG_ICL/Liv1* dataset.

Keyframe strategies Figure 6.5 shows the results of different keyframe selection strategies and their average keyframe sizes on the reconstructed surface quality. Each mark represents a separate evaluation run of a given strategy with a different set of specified parameters. For KF_CONST, we vary the number of consecutive frames κ that are fused into each keyframe. In KF_DIST we adjusted the pose distance threshold Δ_t and Δ_r, while we varied the overlap ratio parameters in KF_OVRLP. In KF_DVO we use the same keyframes as DVO-SLAM. In summary, more relaxed parameters result in a

[1] http://www.danielgm.net/cc/
[2] https://github.com/mp3guy/SurfReg

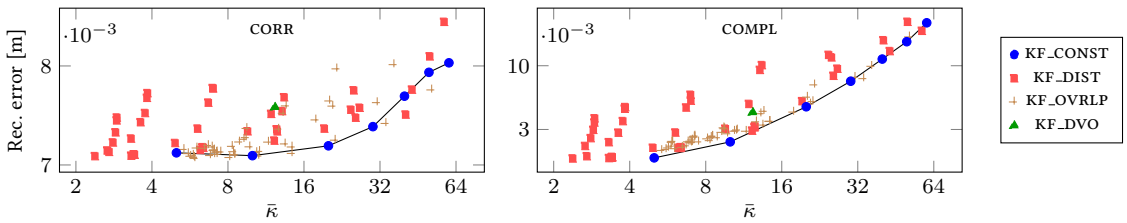


Figure 6.5: Quantitative evaluation of reconstruction correctness (left) and completeness (right) w.r.t. different keyframe strategies on *AUG_ICL/Liv1*. The x-axis shows the average keyframe size $\bar{\kappa}$ produced in each run, the y-axis shows the MAD error (axes are logarithmic). The KF_CONST strategy achieves the best reconstruction results for both CORR and COMPL.

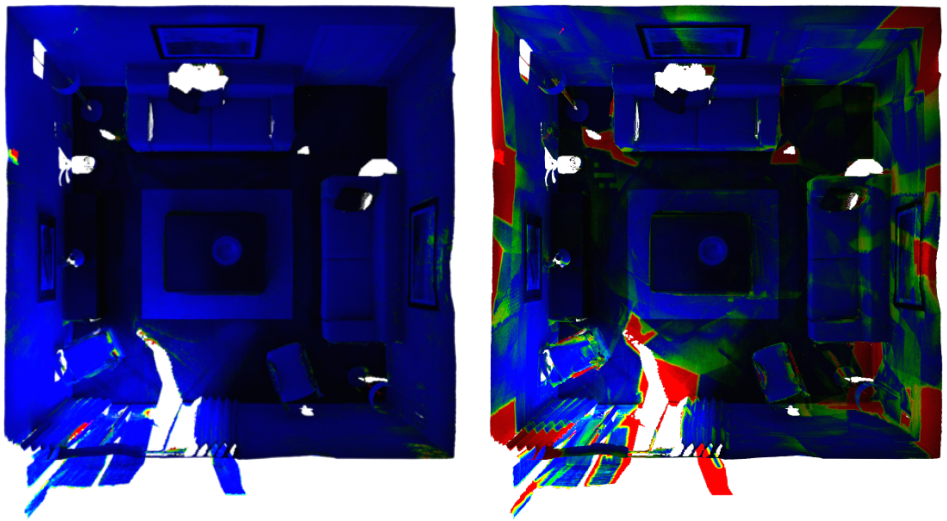
higher average number of fused frames per keyframe $\bar{\kappa}$ for all strategies; different parameter combinations for the same strategy may result in a similar $\bar{\kappa}$. With an increasing $\bar{\kappa}$, the completeness of reconstructions decreases rapidly, since 3D surface information gets lost in 2.5D keyframe fusion. The effect on surface correctness is less significant, since the deviation for the remaining surfaces is still reasonably close to the ground truth 3D model. Compared to KF_CONST, the strategies KF_DIST, KF_OVRLP and KF_DVO result mostly in worse quantitative results, a hardly predictable number of keyframes and barely tunable interdependent parameters. We found KF_CONST to give good quantitative results, while it is also highly predictable w.r.t. fusion and re-integration runtime as well as memory consumption ($\sim 1/\bar{\kappa}$) due to its priorly known number of frames per keyframe. We refer the reader to the supplementary material for more details.

Completeness Figure 6.6 shows color coded distance renderings for KF_CONST keyframe fusion with $\kappa = 5$ and $\kappa = 60$ on *AUG_ICL/Liv1*. The colors represent errors from 0mm (blue) to 50mm (red). Again, the completeness COMPL of reconstructions decreases with more fused frames per keyframe because of the loss of surface information with 2.5D keyframes. The reconstructed surfaces are still accurate w.r.t. ground truth (CORR).

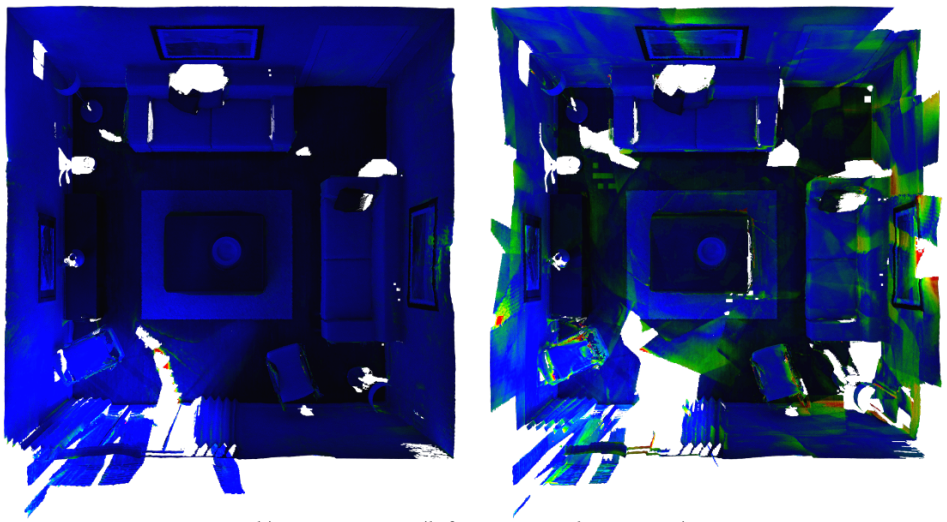
6.5.2 Surface Re-integration

We finally assess our surface correction w.r.t. real-time performance and show results of on-the-fly surface re-integration on real-world data.

Runtime While BundleFusion requires two high-end GPUs to operate in real-time, our system requires only a single GPU. Figure 6.7 gives the average amortized runtimes of our system, specifically for (re-)integration of frames w.r.t. κ (red), for DVO-SLAM (green) and the total runtime of both combined (blue). Generally, the higher κ is, the



a) Completeness (left $\kappa = 5$, right $\kappa = 60$)



b) Correctness (left $\kappa = 5$, right $\kappa = 60$)

Figure 6.6: Completeness and correctness after integration of *AUG_ICL/Liv1* with KF_CONST keyframe strategy (with keyframe sizes 5 and 60).

fewer keyframes are generated and thus need to be updated. We accomplish real-time performance with $\kappa = 20, m = 5$. Here, the re-integration strategy of updating the m most-moved *consecutive* keyframes (solid lines) saves already 47% of (re-)integration runtime compared to BundleFusion's simple strategy (dashed). This is further accelerated through the use of keyframes only: Overall, the reconstruction with $\kappa = 20, m = 5$ and our re-integration strategy takes 93% less time than BundleFusion's re-integration strategy without keyframe fusion ($\kappa = 1, m = 100$). We found $m \in [10, 20]$ to be a good trade-off between reconstruction quality and model correction speed for most data sets.

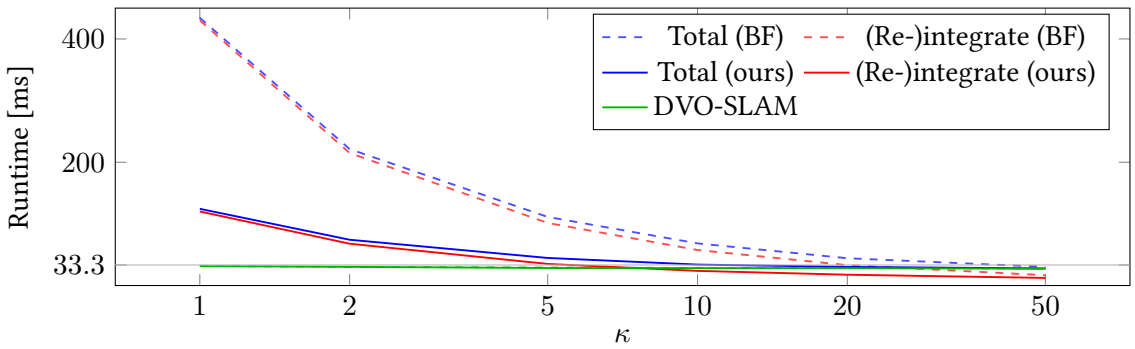


Figure 6.7: Average runtime per frame for reconstruction of *AUG_ICL/Liv1* with KF_CONST w.r.t. κ. Our re-integration strategy (solid) is substantially faster than BundleFusion's (dashed). m was set to $100/\kappa$, yielding a constant effective re-integration rate.

On-the-fly surface re-integration As DVO-SLAM steadily optimizes a pose graph and issues pose updates, our surface correction method gradually improves the reconstructed 3D model on-the-fly by re-integrating the most-moved consecutive m keyframes into the SDF. While updating all changed keyframes at once is too expensive, we can control the speed of incorporating pose updates into the 3D model by adjusting m. Also, with decreasing $\bar{\kappa}$ more keyframes are generated and need to be updated. Figure 6.8 shows an example of how a 3D reconstruction is corrected on-the-fly during the reconstruction to be as globally consistent as possible (with $m = 5$, $\kappa = 20$ and KF_CONST strategy).

An isolated comparison of the surface correction of ElasticFusion [123] with our method is not applicable since the respective SLAM systems may result in different camera trajectories. Nevertheless, Figure 6.9 shows a qualitative comparison of the generated models, which is in accordance with the findings in [28]. While ElasticFusion might benefit from camera tracking against the corrected model, the point cloud reconstructed with ElasticFusion with default parameters exhibits double walls and artifacts due to potentially undetected loop closures and surface warping artifacts. These effects are mitigated in the continuous surface mesh reconstructed from our method, which successfully corrects the model on-the-fly.

6.6 Conclusion

We presented an efficient online surface re-integration method for globally consistent 3D reconstruction of large-scale scenes from RGB-D sensors in real-time on a single GPU only. Our SLAM system based on DVO-SLAM estimates the camera motion in real-time on a CPU and employs pose graph optimization for obtaining globally optimized camera poses. Multiple RGB-D frames are first fused into keyframes, which are then integrated

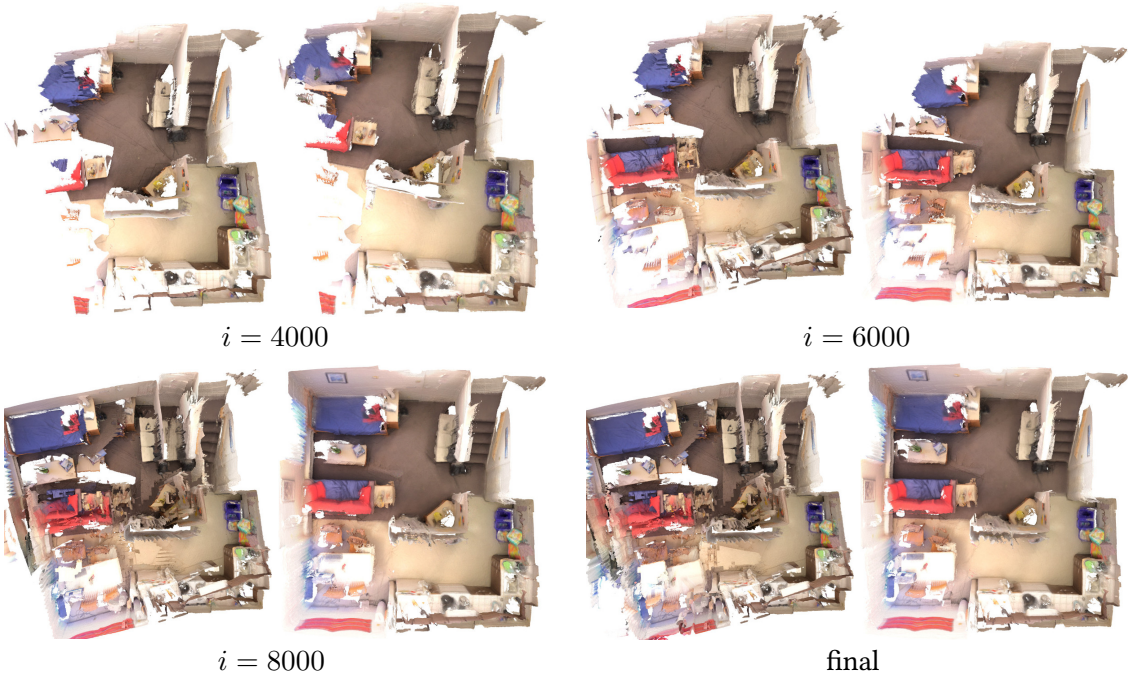


Figure 6.8: Reconstruction of *BundleFusion/apt0*. Every 2000 frames, a model was generated without (left) and with (right) on-the-fly surface correction (KF_CONST keyframe strategy with $m = 5$, $\kappa = 20$).

Figure 6.9: Qualitative comparison of ElasticFusion [123] (left) with our method (right) on *BundleFusion/apt0*. The point cloud reconstructed with ElasticFusion exhibits artifacts due to potentially undetected loop closures and surface warping artifacts, whereas our method successfully corrects the model.

into a sparse voxel hashed SDF model representation. Continuous keyframe pose updates are gradually incorporated into the SDF volume by on-the-fly re-integration of changed keyframes. Our improved re-integration strategy with correction of keyframes and significantly reduced host-GPU-streaming saves about 93% of runtime compared to the state-of-the-art. By re-integrating keyframes (instead of all frames), we substantially reduce the number of frames to be re-integrated with only a slight degradation of reconstruction quality.

Acknowledgement This work was funded by the ERC Consolidator grant *3D Reloaded*.

Part III

Conclusion and Outlook

Chapter 7

Summary

In this thesis, we introduced novel algorithms for reconstructing high-quality 3D models of real scenes using low-cost commodity RGB-D sensors. To this end, we first proposed techniques for enhancing the level of detail in the reconstructed scene geometry and for recovering sharp model textures in order to improve typically oversmooth and blurry SDF-based 3D reconstructions. Additionally, we developed an efficient surface correction method for updating dense SDF volumes on-the-fly, which enables high-quality model previews in real-time whenever the SLAM system updates camera poses on loop closures. In the following, we summarize the key contributions presented in the individual chapters in Part II of this dissertation.

Texture Mapping using Super-Resolution Keyframes The novel texture mapping method demonstrated in Chapter 4 employs a robust super-resolution keyframe fusion to achieve high-quality model appearance. This dramatically improves the photorealism of 3D models generated by fusing RGB-D frames in an SDF with smoothing ℓ_2-regularization. Firstly, the low-quality input frames are fused into super-resolution keyframes of higher resolution. In particular, we utilize advanced filtering techniques to preserve a high level of detail: a blurriness measure mitigates frames with motion blur and the color observations are finally fused using a weighted median scheme. Secondly, we texture map the produced super-resolution keyframes onto the reconstructed 3D triangle mesh, again using a weighted median per-texel color computation. The experimental results show that the devised method is able to generate consistent and sharp model textures, with a significantly better quality compared to simple volumetric blending. Its robustness and efficient runtime make the approach highly practical for real-world 3D scanning applications.

Joint Geometry and Appearance Optimization In Chapter 5, a joint 3D reconstruction approach that simultaneously optimizes scene geometry, material albedo and image formation model was proposed. Our method recovers fine-scale geometric details from a coarse initial SDF volume as well as sharp surface textures. High-resolution RGB keyframes are selected using a blurriness measure and utilized in a joint optimization based on Shape-from-Shading. A flexible lighting model based on Spatially-Varying Spherical Harmonics is introduced, allowing to faithfully approximate complex scene illumination. The estimated lighting is then employed in a shading-based refinement method that jointly optimizes for surface geometry and material albedo encoded in a voxel-hashed SDF volume along with the image formation model including keyframe camera poses. We show in extensive qualitative and quantitative evaluations that the developed technique recovers impressive geometry details and an improved reconstruction of spatially-varying scene lighting. Through joint optimization of both geometry and camera poses, the color sampling in the keyframes continuously improves, which implicitly leads to optimal high-quality surface colors as a by-product.

Efficient Online Surface Correction In Chapter 6, we addressed the challenging task of efficient on-the-fly surface re-integration for large-scale 3D reconstruction using SDFs. In combination with a real-time SLAM system [57] that continuously optimizes a global pose graph and reduces drift on loop closures, our method manages to gradually correct a sparse voxel-hashed SDF [86] on pose updates. Consecutive RGB-D input frames are first fused in keyframes with more complete depth and reduced noise. These keyframes are then directly incorporated into the global 3D model using their initial poses. As the scanning advances, the SLAM system potentially finds loop closures of previously visited places and consequently updates its pose graph to maximize global consistency. To accordingly update the SDF volume, we de-integrate and re-integrate the updated keyframes with their new poses on-the-fly. Our advanced keyframe selection and re-integration strategy considerably accelerates GPU-host-streaming for real-time performance. Our experiments demonstrate that the runtime is improved by up to 93% compared to the state-of-the-art. Our method benefits highly from re-integrating keyframes only (instead of all frames), requiring significantly fewer frames to be corrected. This comes with only a slight degradation of the reconstructed surface geometry, which renders up-to-date model previews with high global consistency and enhanced visual quality possible. The presented method requires only a single GPU and is a first step towards real-time online surface correction in large-scale dense 3D reconstruction.

Chapter 8

Future Research

The techniques presented in this thesis achieve compelling state-of-the-art results in the context of 3D reconstruction from consumer RGB-D cameras. Despite all recent advances, there are still many open challenges and possible extensions to existing approaches. In the following, we cover interesting directions that can potentially be addressed in future research.

RGB-D Sensors and Calibration Since the Microsoft Kinect, RGB-D sensors have had a significant impact on solving practical 3D reconstruction problems. However, slow progress in sensor hardware has led to a stagnation of depth and color image quality. The noisy depth maps have limited image resolutions, while the associated RGB images contain noticeable visual artifacts due to low-quality lenses and cheap image sensors without global shutter. Moreover, the inexact intrinsic and extrinsic factory calibration and the lack of hardware synchronization between depth and color often lead to uncompensated inaccuracies in the 3D reconstruction. However, the framework in [118] has proved that a thoroughly assembled and calibrated camera rig is crucial for impressive 3D reconstructions. A further miniaturization of depth sensors with lower power consumption is required for a more wide-spread use in smartphones or HMDs.

To remedy the limitations above, there are some new RGB-D sensors on the horizon such as the new *Microsoft Azure Kinect*[1]. In contrast to using depth sensing hardware, there is a lot of potential in combining learning-based monocular or stereo depth estimation methods [69, 132] with existing RGB-D based 3D reconstruction approaches.

Real-time Large-Scale Geometry Refinement The optimization method in Chapter 5 reconstructs high-quality 3D models of only limited-size objects due to its high memory requirements. We consider partitioned optimization techniques as a possible direction for refining even large-scale reconstructions, in which two steps are iterated:

[1] `https://azure.microsoft.com/en-in/services/kinect-dk/`

(1) fixed-size subvolumes of the reconstruction are optimized independently; (2) we slide the subvolumes about half of their size in order to afterwards refine the unoptimized boundaries. To speed up the slow optimization runtimes and aim for real-time performance, a data-parallel NLS solver on the GPU can be employed, either implemented as a custom solver or using the generic *Opt Solver* library [29].

Geometry Simplification and Completion While fusing many RGB-D frames in a SDF volume effectively regularizes the depth noise, the geometry may still contain noise or holes in regions with few or no observations. By analyzing the geometry and incorporating shape priors such as planes [2, 49, 70], CAD models [12] or other geometric primitives [100], the surface geometry can be completed and denoised. These shape priors have the potential to enable compact scene representations with a very low memory footprint, which is crucial for large-scale 3D reconstruction of almost unlimited scale.

Volumetric 3D deep learning approaches have recently demonstrated that semantic scene analysis helps to complete geometry [27, 48, 102], but are still suffering from their high memory requirements and limited resolution. Geometric deep learning [20, 81] and graph convolutional neural networks [38] provide an interesting research direction for analyzing and simplifying 3D model geometry.

Scene Decomposition Despite the recent progress, 3D reconstructions obtained in unconstrained environments commonly still lack photorealism. For example, uncaptured material reflectance properties and baked-in shadows or specularities are often hindering the use of such models in VR applications, where realistic re-lighting of objects is essential. Our approach in Chapter 5 tackles the complex problem of separating geometry and albedo by heavily relying on a heuristic but often insufficient albedo regularization term. Additionally, more advanced surface reflectance models like BRDFs with material roughness parameters for specularities are required in practice.

Recent works using deep neural networks have shown their potential to effectively estimate material reflectance [13, 60, 80], outperforming handcrafted albedo regularizers. Likewise, deep learning based methods can approximate the scene illumination more realistically than SVSH lighting; this allows to faithfully re-light virtual objects in reconstructed 3D models [37, 65, 101]. The differentiable rendering [54, 74] and raytracing [68] techniques utilized in these works enable semi- or unsupervised learning by incorporating a differentiable physical light transport model already during training.

Online Surface Correction On-the-fly 3D surface correction, in particular of SDFs, is still a research topic that is not fully explored. Most state-of-the-art SLAM methods have lightweight internal scene maps for enabling up-to-date localization and incorporating

loop closures. These maps are however independent of a dense 3D surface representation for model visualization and analysis.

Using fused keyframes for surface correction in Chapter 6 significantly improves performance and memory consumption compared to [28]. To scale even better w.r.t. memory usage as the scanning progresses, revisiting of keyframes is appealing. This involves fusion of input frames into existing keyframes that observe the same scene parts, instead of creating new ones. By avoiding to redundantly store 3D scene information, the method becomes suitable for robotic scenarios with limited hardware capabilities. Moreover, adopting the revised hash map structure of the voxel-hashed SDF in [53] further speeds up the re-integration times of our presented method and allows to ultimately correct more keyframes per pose update.

While we have utilized dense SDF volumes for reconstructing and updating the geometry, fundamental research on suitable 3D representations for surface correction is necessary. Memory efficiency, real-time performance for fusion and correction, and online 3D mesh extraction are aspects of special interest. Recent methods employ deformation graphs that have local SDF subvolumes [52] or dense surfel maps [94, 95, 123] attached. These graphs are corrected on loop closures and allow the attached local 3D reconstructions to move along; while this obviates the need for explicit de-/re-integration, it also involves a more sophisticated real-time mesh extraction.

Part IV

Appendix

Appendix A

Supplementary Material for Publication

Intrinsic3D: High-Quality 3D Reconstruction by Joint Appearance and Geometry Optimization with Spatially-Varying Lighting

In this supplementary material, we provide additional experiments and details. Specifically, we give an overview of the mathematical symbols in Sec. A.1, and in Sec. A.2 we provide a thorough quantitative evaluation regarding the geometric reconstruction quality on ground truth data (both real and synthetic). We further show qualitative results of the reconstructed models on several own and publicly-available datasets, with a focus on both reconstruction geometry and appearance; see Sec. A.3. Finally, in Sec. A.4, we detail additional experiments on spatially-varying lighting under both qualitative and quantitative standpoints.

A.1 List of Mathematical Symbols

Symbol	Description
$\boldsymbol{p}$	continuous 3D point in $\mathbb{R}^3$
$\boldsymbol{x}$	continuous 2D image point in $\mathbb{R}^2$
$\boldsymbol{v}$	position of voxel in $\mathbb{R}^3$
$\boldsymbol{v}_c$	position of voxel center of $\boldsymbol{v}$ in $\mathbb{R}^3$
$\boldsymbol{v}_0$	position of $\boldsymbol{v}$ transformed onto iso-surface in $\mathbb{R}^3$
$\mathbf{n}(\boldsymbol{v})$	surface normal at $\boldsymbol{v}$ in $\mathbb{R}^3$
$\mathbf{D}(\boldsymbol{v})$	signed distance value at $\boldsymbol{v}$
$\mathbf{C}(\boldsymbol{v}), \mathbf{I}(\boldsymbol{v})$	color (RGB) and intensity at $\boldsymbol{v}$
$\mathbf{W}(\boldsymbol{v})$	integration weight at $\boldsymbol{v}$
$\mathbf{a}(\boldsymbol{v})$	albedo at $\boldsymbol{v}$
$\tilde{\mathbf{D}}(\boldsymbol{v})$	refined signed distance value at $\boldsymbol{v}$
$\mathbf{D}_0$	iso-surface of the refined SDF
$\mathbf{B}(\boldsymbol{v})$	estimated reflected shading at $\boldsymbol{v}$
$\mathbf{\Gamma}(\boldsymbol{v})$	chromaticity at $\boldsymbol{v}$
t_{shell}	thin shell size
N	number of voxels inside the thin shell region
K, t_{sv}	number of subvolumes and subvolume size in $\mathbb{R}^3$
$\mathcal{S}$	set of subvolumes s_k
ℓ	vector of all lighting coefficients l_m
H_m	m-th spherical harmonics basis
b	number of spherical harmonics bands
M	number of input frames
$\mathcal{C}_i, \mathcal{I}_i, \mathcal{Z}_i$	color, intensity and depth image of frame i
$\mathcal{T}_i$	transformation from frame i to the base frame
t_{KF}	keyframe selection window size
$t_{\text{best}}, \mathcal{V}_{\text{best}}$	number of best views for $\boldsymbol{v}$ and corresponding set
$d_i(\boldsymbol{v})$	projective distance to voxel center in frame i
$w_i(\boldsymbol{v})$	sample integration weight of frame i
$\mathcal{O}_{\boldsymbol{v}}$	set of color observations of $\boldsymbol{v}$
$c_i^{\boldsymbol{v}}$	observed color of $\boldsymbol{v}$ in frame i
$w_i^{\boldsymbol{v}}$	observation weight of $\boldsymbol{v}$ in frame i
f_x, f_y, c_x, c_y	camera intrinsics (focal length, optical center)
$\kappa_1, \kappa_2, \rho_1$	radial and tangential lens distortion parameters
$\mathcal{X}$	stacked vector of optimization variables

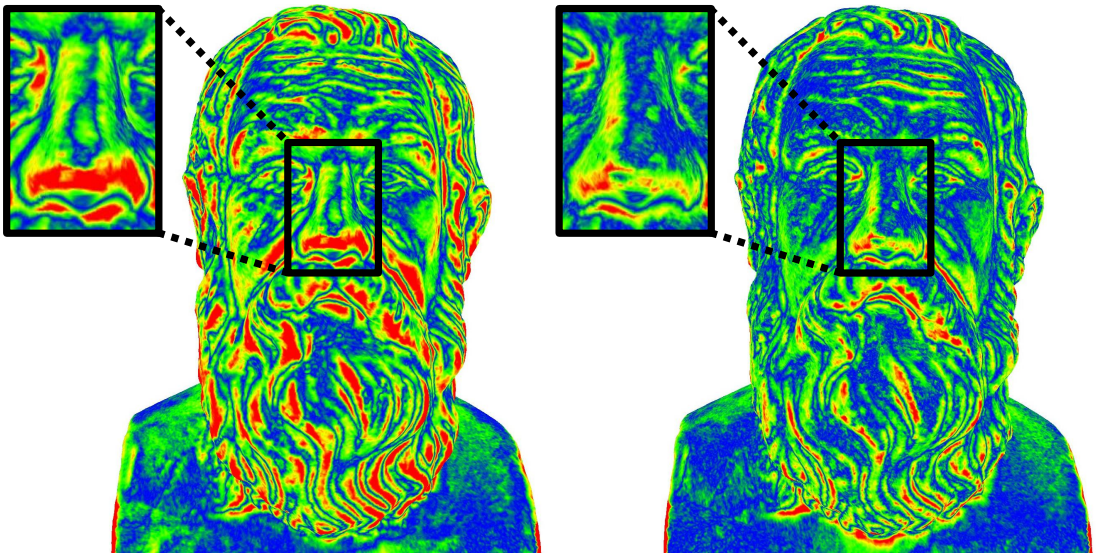


Figure A.1: Surface accuracy comparison with a ground truth laser scan of the *Socrates* dataset: the approach of Zollhöfer et al. [135] (left) exhibits a higher mean absolute deviation from the ground truth compared to our method (right).

A.2 Quantitative Geometry Evaluation

In the following, we show a quantitative surface accuracy evaluation of our geometry refinement on the *Socrates* and *Frog* datasets.

A.2.1 Socrates

In order to measure the surface accuracy of our method quantitatively, we first compare our method with a ground truth laser scan of the *Socrates* Multi-View Stereo dataset from [135]. The mean absolute deviation (MAD) between our reconstruction and the laser scan is 1.09mm (with a standard deviation of 2.55mm), while the publicly-available refined 3D model of Zollhöfer et al. [135] has a significantly higher mean absolute deviation of 1.80mm (with a standard deviation of 3.35mm). This corresponds to an accuracy improvement of 39.44% of our method. Figure A.1 visualizes the color-coded mean absolute deviation on the surface.

A.2.2 Frog

Besides a quantitative comparison with a laser scan, we also evaluate the surface accuracy of a 3D model reconstructed from synthetic RGB-D data. We therefore generated the synthetic *Frog* dataset by rendering a ground truth mesh with a high level of detail

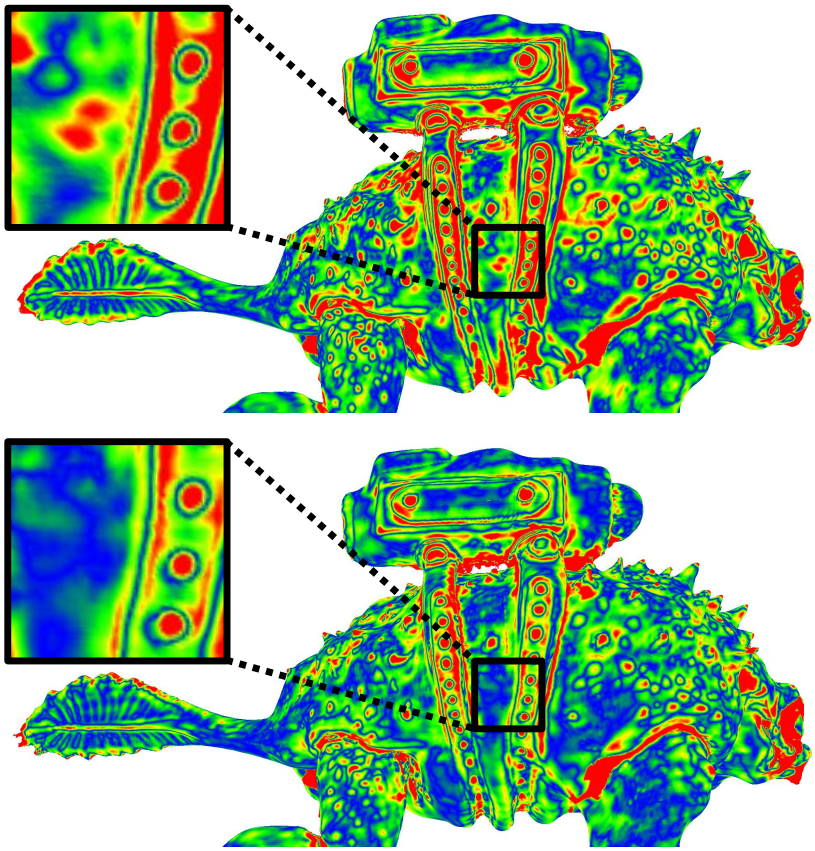


Figure A.2: Surface accuracy comparison on synthetic data with a ground truth mesh of the *Frog* dataset: our method (bottom) generates more accurate results compared to Zollhöfer et al. [135] (top).

into synthetic color and depth images. We smooth the depth maps using a bilateral filter and add Gaussian noise to both the depth values and to the camera poses.

Instead of comparing the reconstructed 3D models directly with the original mesh, we instead fuse the generated noise-free RGB-D frames into a Signed Distance Field and extract a 3D mesh with Marching Cubes [75]. This extracted mesh is then used as ground truth reference and represents the best possible reconstruction given the raycasted input data in combination with an SDF volume representation.

The mean absolute deviation between our reconstruction and the ground truth mesh is 0.222mm (with a standard deviation of 0.269mm). With the reconstruction generated using our implementation of [135], we obtain a substantially higher mean absolute deviation of 0.278mm (with a standard deviation of 0.299mm). Compared to [135], our method improves the reconstruction accuracy by 20.14% and is able to reveal geometric details lost with [135]. Figure A.2 visualizes the color-coded mean absolute deviation on the surface.

A.3 Examples of 3D Reconstructions

In addition to providing a thorough quantitative ground truth evaluation, we show qualitative results of 3D models reconstructed from several RGB-D datasets. In particular, we present 3D reconstructions of the publicly-available *Relief* and *Lucy* datasets from Zollhöfer et al. [135] as well as 3D models of the *Gate*, *Lion*, *Hieroglyphics*, *Tomb Statuary* and *Bricks* datasets that we acquired with a Structure Sensor.

Apart from showing the fine detailed geometry, we also demonstrate the improved appearance of the reconstructions, which we implicitly obtain by jointly optimizing for surface, albedo, and image formation model parameters within our approach.

A.3.1 Relief

In Figure A.3, we show a comparison of the appearance generated using our method with simple volumetric fusion (e.g., Voxel Hashing [86]) and the shading-based surface refinement approach by Zollhöfer et al. [135]. The results in (a) and (b) are visualizations from the meshes that are publicly-available on the project website of [135]. The close-ups successfully visualize that our method results in significantly sharper textures.

A.3.2 Lucy

In Figure A.4, we present a visual comparison of the reconstructed surface geometry of the *Lucy* dataset. Note how volumetric fusion (a) and Zollhöfer et al. [135] (b) cannot reveal fine-scale details due to the use of averaged per-voxel colors for the refinement, while our method gives the best results and provides geometric consistency (c).

Regarding appearance, we can observe in Figure A.5 that our method (c) provides a more detailed texture compared to fusion (a) and Zollhöfer et al. [135] (b).

A.3.3 Additional Datasets

While the *Relief* and *Lucy* datasets provided by [135] consist of rather small objects with only few input RGB-D frames and short camera trajectories, we acquired more advanced RGB-D datasets using a Structure Sensor.

Figure A.6 shows the reconstruction of the *Gate* dataset, while the 3D model of the *Lion* dataset is visualized in Figure A.7. The 3D reconstructions of *Hieroglyphics*, *Tomb Statuary* and *Bricks* are presented in Figure A.8, Figure A.9 and Figure A.10 respectively. For all of these datasets, our method generates high-quality 3D reconstructions with fine-scale surface details and and compelling visual appearance with sharp texture details. In contrast, the models obtained from volumetric fusion lack fine details in both geometry and appearance.

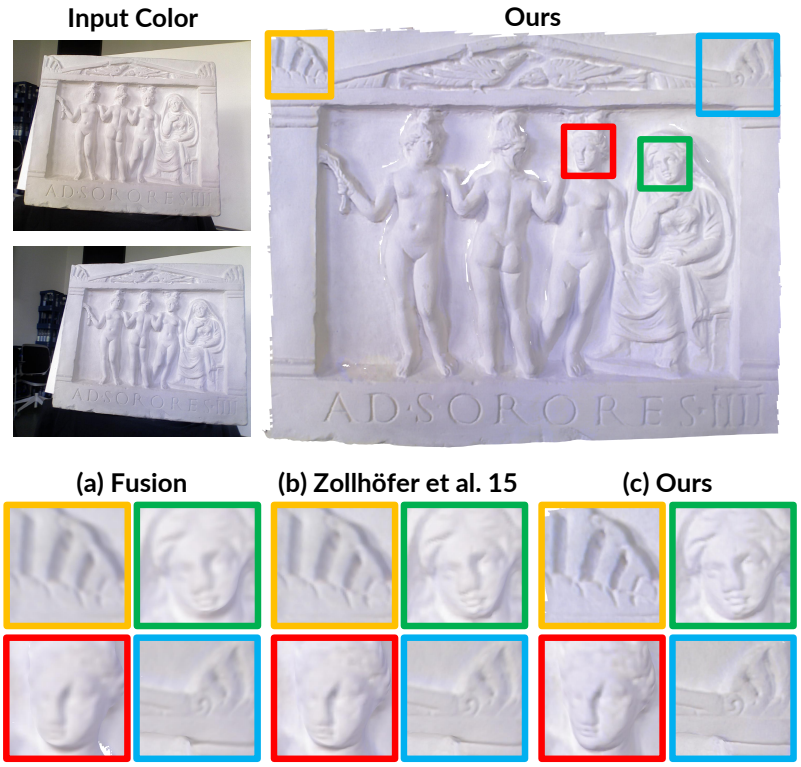


Figure A.3: Refined appearance of *Relief* dataset: our method (c) reconstructs significantly sharper textures compared to (a) and (b). Close-ups of ornaments (yellow, blue) and figures (green, red) exhibit more visual details.

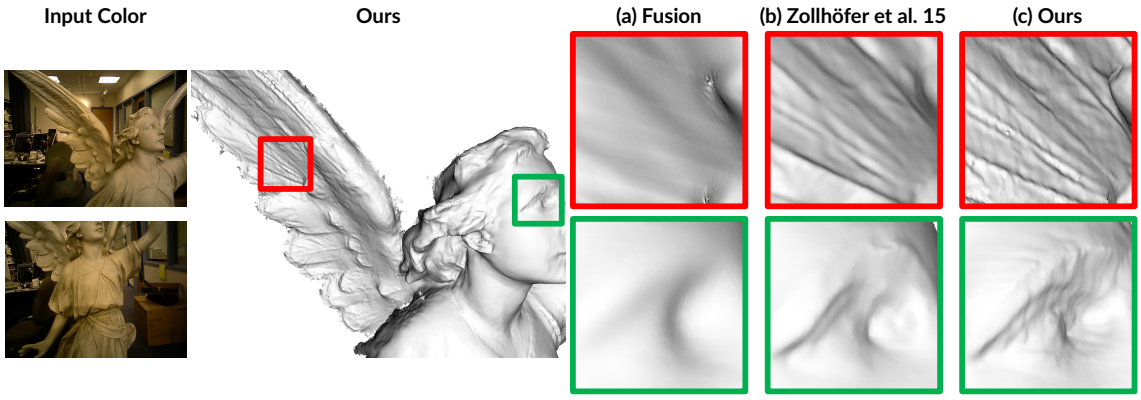


Figure A.4: Refined geometry of *Lucy* dataset: volumetric fusion (a) with its strong regularization gives only coarse models. Zollhöfer et al. [135] (b) generate more details; however, limited by using averaged per-voxel colors for the refinement. Our approach that jointly optimizes for all involved parameters (c) reconstructs fine-detailed high-quality geometry.

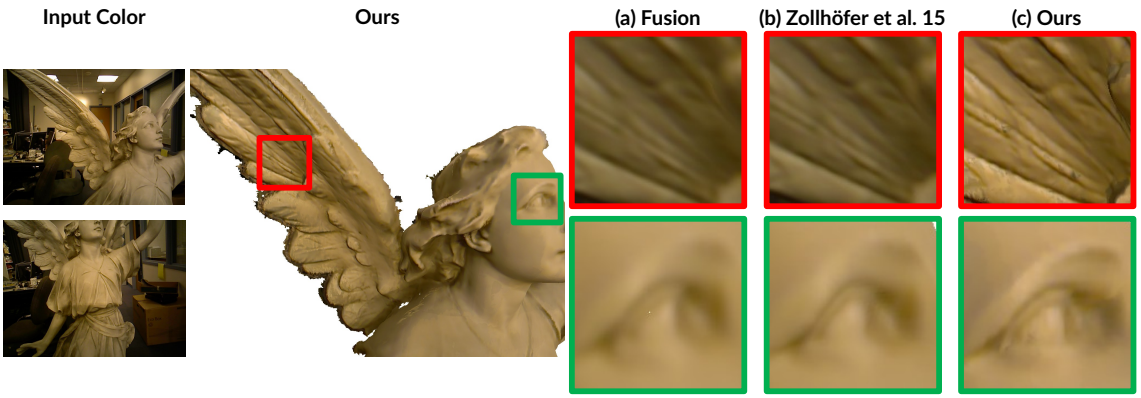


Figure A.5: Refined appearance of *Lucy* dataset: in addition to precise geometry our method (c) also produces high-quality colors compared to (a) and (b).

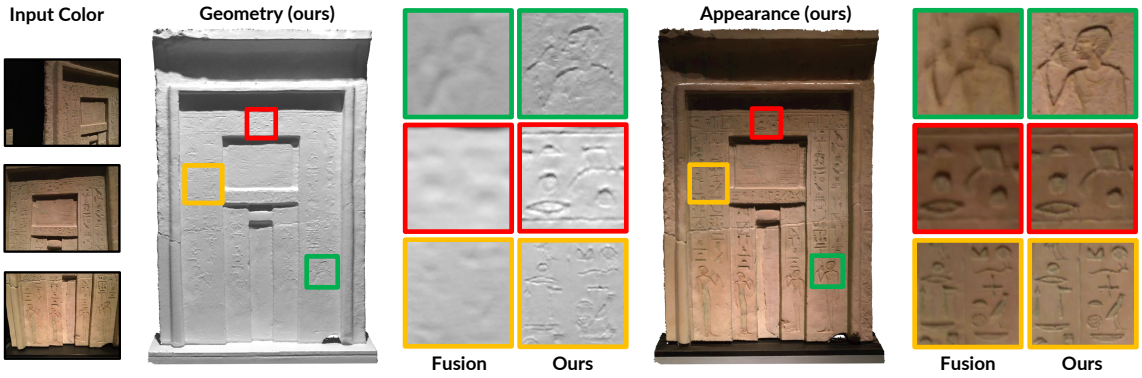


Figure A.6: Reconstruction of the *Gate* dataset.

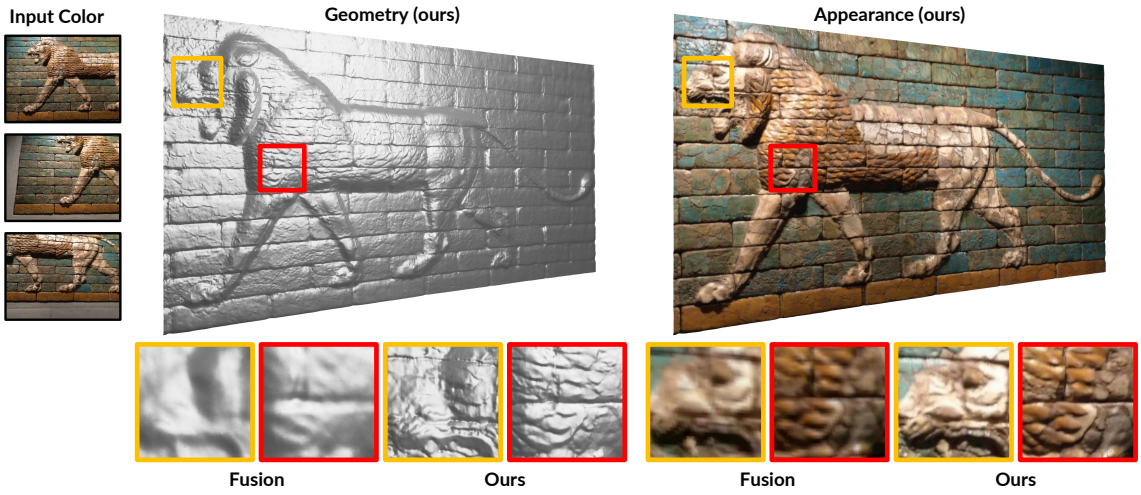


Figure A.7: Reconstruction of the *Lion* dataset.

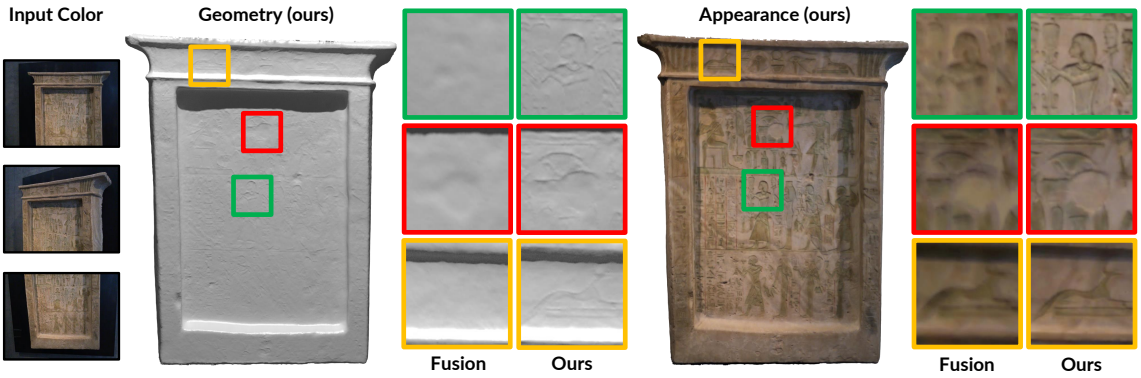


Figure A.8: Reconstruction of the *Hieroglyphics* dataset.

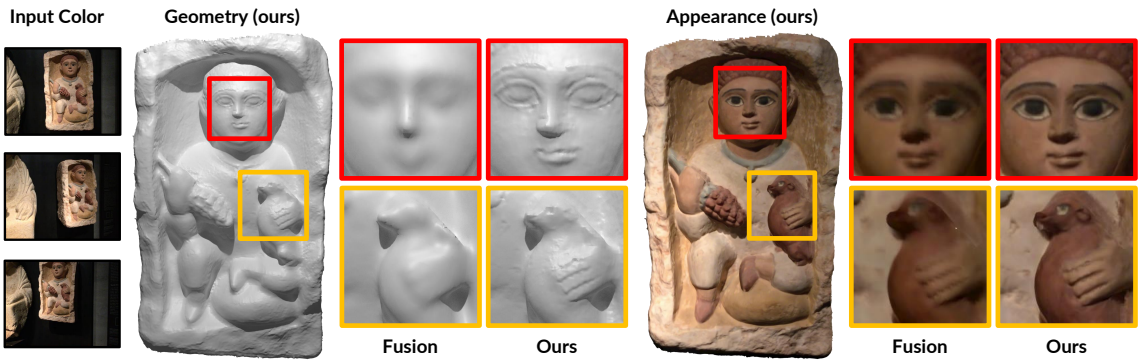


Figure A.9: Reconstruction of the *Tomb Statuary* dataset.

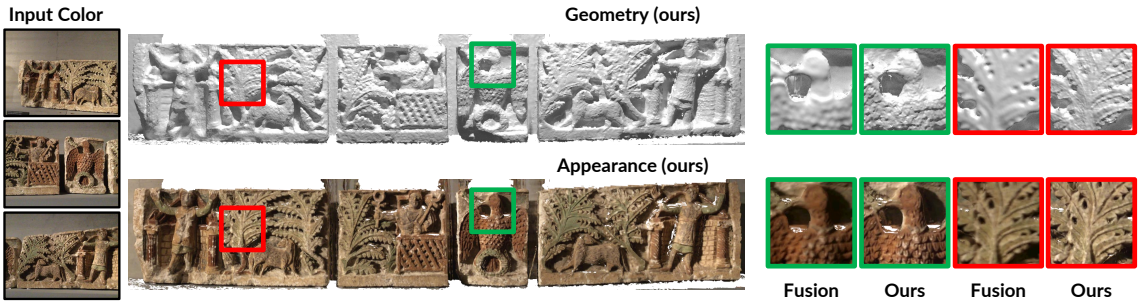
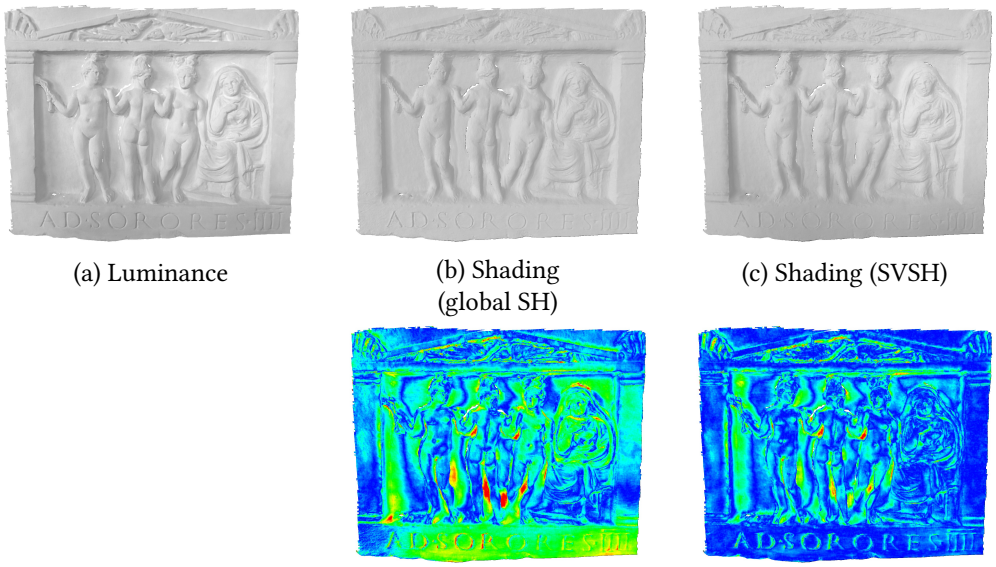


Figure A.10: Reconstruction of the *Bricks* dataset.



(a) Luminance

(b) Shading (global SH)

(c) Shading (SVSH)

(d) Difference (global SH)

(e) Difference (SVSH)

Figure A.11: Estimated illumination of *Relief* dataset: the differences between input luminance (a) and estimated shading (b) and (c) are less for SVSH (e) than for global SH (d), meaning a better approximation of the illumination.

A.4 Evaluation of Spatially-Varying Lighting

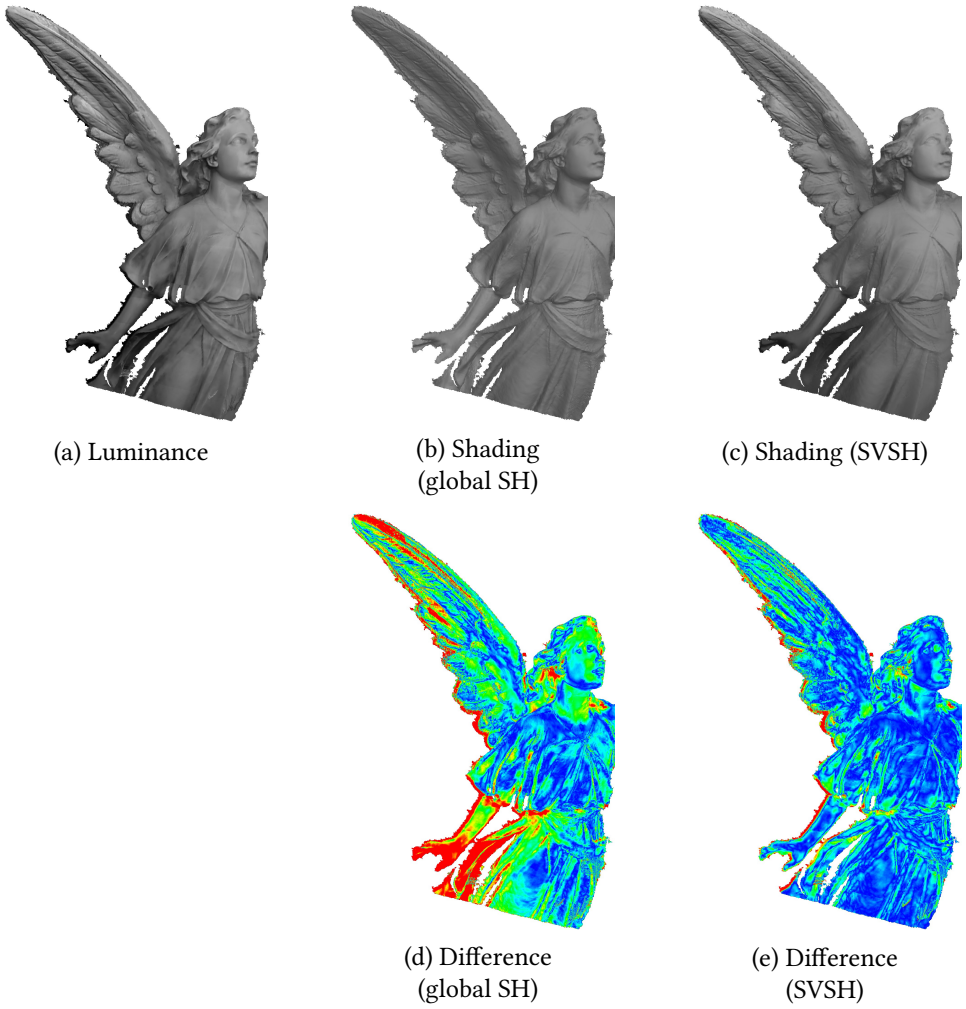
In this section, we present further qualitative results for lighting estimation via spatially-varying spherical harmonics (SVSH) compared to global spherical harmonics (global SH) on various datasets. We use the same underlying geometry for both variants of lighting estimation for each dataset.

Error Metric As a metric, we use the absolute difference between estimated shading and observed input luminance of a voxel $\boldsymbol{v}$; i.e.,

$$\mathbf{B}_{\text{diff}} = |\mathbf{B}(\boldsymbol{v}) - \mathbf{I}(\boldsymbol{v})|, \tag{A.1}$$

to determine the quality of the illumination for given geometry and albedo. Ideally, this difference should be as small as possible.

Relief For the *Relief* dataset, the differences between lighting estimation with global SH and SVSH (with a subvolume size of 0.05m) are shown in Figure A.11. It becomes obvious that even for seemingly simple scenes, a single global set of Spherical Harmonics coefficients cannot accurately reflect real-world environments with complex lighting.



(a) Luminance

(b) Shading (global SH)

(c) Shading (SVSH)

(d) Difference (global SH)

(e) Difference (SVSH)

Figure A.12: Estimated illumination of *Lucy* dataset: illumination with SVSH (c) explains the illumination better than global SH only (b), resulting in less differences (e) compared to (d) between input luminance (a) and shading.

Lucy Similar to the *Relief*, SVSH (with a subvolume size of 0.05m) can better approximate the complex illumination in the *Lucy* dataset than global SH. Figure A.12 visualizes the differences in the estimated shadings.

Appendix B

Supplementary Material for Publication

Efficient Online Surface Correction for Real-time Large-Scale 3D Reconstruction

In this chapter, we provide additional experiments and details of our work "Efficient Online Surface Correction for Real-time Large-Scale 3D Reconstruction". Specifically, we demonstrate the effect of the different keyframe strategies on the completeness of reconstructions; see Section B.1. Moreover, we provide a detailed evaluation of the runtime (Section B.2) and memory consumption (Section B.3) of our surface correction method. Finally, Section B.4 shows some more qualitative results for on-the-fly surface re-integration on additional datasets. Our evaluation was performed on a workstation with Intel Core i7-3770 CPU, 32GB RAM and an NVIDIA GeForce GTX 1070 GPU.

Datasets For our evaluation, we use publicly available RGB-D datasets, an overview is given in Table B.1. All used sequences depict larger scenes and provide registered depth and color images (with a resolution of 640×480 at 30 fps) as well as respective camera poses. For assessing the surface quality and to eliminate a substantial source of error, we rely on on the (ground truth) camera poses provided directly with the datasets.

The real-world sequence *TUM/long_office_household* [111] shows a long sequence with a loop closure and provides ground truth camera poses obtained from a high-speed motion capture system.

BundleFusion/apt0 [28] features a very long camera trajectory estimated using a highly accurate 3D reconstruction framework.

AUG_ICL/Liv1 [23] is a synthetic RGB-D sequence generated from manually modeled scenes of a living room. In addition to providing ground truth camera poses, the dataset also provides exact ground truth 3D scene models which allow for a quantitative comparison of surface quality of reconstructed 3D models. The *AUG_ICL* sequences

Sequence	# frames	Synthetic	GT trajectory	GT 3D model
TUM/long_office_household [111]	2486	No	Yes	No
AUG_ICL/Liv1 [23]	2870	Yes	Yes	Yes
BundleFusion/apt0 [28]	8560	No	No	No

Table B.1: RGB-D datasets used for our evaluation. (GT stands for ground truth)

consist of separate sequences with clean and noisy depth maps, which exhibit a more realistic camera noise model.

Surface evaluation methods and metrics In order to perform a quantitative evaluation of our reconstructed 3D models with ground truth models, we first introduce the employed methods and metrics. The evaluation procedure for a comparison with synthetic ground truth is adapted from [42] and first extracts a 3D mesh $\mathcal{M}$ from the Signed Distance Field (SDF) volume using the Marching Cubes algorithm [75].

Then, using CloudCompare [1], we sample a ground truth reference model $\mathcal{R}$ from the mesh provided for *AUG_ICL/Liv1*, giving a point cloud with 50 million vertices distributed uniformly on the models. Note that we generate distinct models for clean and noisy sequences. By using the poses from the sequences, our models are already aligned with the reference. We use SurfReg [2] to measure the distance of each vertex of our reconstruction $\mathcal{M}$ to its closest vertex in the reference point cloud $\mathcal{R}$ and compute the values for the mean absolute deviation MAD. This technique assesses the *correctness* (CORR) of the model and basically compares the accuracy of the reconstructed surfaces w.r.t. the ground truth model.

However, this method fails to evaluate the *completeness* (COMPL) of the reconstruction, which is especially important for determining the information loss when applying keyframe fusion. For measuring COMPL, we use the inverse procedure and compare every vertex of the reference $\mathcal{R}$ to the nearest neighbor in $\mathcal{M}$. Since the ground truth model contains more surface than actually covered in the synthetic RGB-D frames, we re-generate the reference models by fusing all input frames (without keyframe fusion) into the SDF volume using the ground truth trajectory. This yields reference models that are as complete as possible based on the synthetic frames.

[1] `http://www.danielgm.net/cc/`
[2] `https://github.com/mp3guy/SurfReg`

B.1 Keyframe Strategies

In our work, we have introduced four different strategies for creating Keyframe Fusion keyframes (KF keyframes), namely KF_CONST, KF_DIST, KF_OVRLP and KF_DVO. The number of input RGB-D frames fused into a keyframe has an important effect on both re-integration efficiency and reconstruction surface quality. The more keyframes are generated, the less frames need to be re-integrated on DVO-SLAM pose graph updates, resulting in a more efficient surface correction. At the same time there is a loss in reconstruction quality (completeness COMPL in particular) that goes along with less keyframes, since 3D information cannot be fully represented in the 2.5D keyframe depth maps. Our keyframe fusion generally follows [6] and consists of separate steps for depth and color fusion.

The different keyframe strategies result in a different number of keyframes and consequently directly influence the reconstruction quality. Each keyframe strategy can be configured with specific parameters; the more relaxed these parameters are, the more keyframes are created in general. Figure B.1 shows the effect of the different strategies on the completeness COMPL of noisy *AUG_ICL/Liv1*, with parameters chosen in a way that all strategies fuse roughly the same average number $\bar{\kappa}$ of input frames in each keyframe. We additionally show the result of omitting intermediate frames, i.e. only integrating the (unfused) depth map of the input frame corresponding to the keyframe into the SDF volume.

B.2 Surface Correction Runtime Evaluation

In the following, we show a quantitative runtime evaluation of our method. We rely on the KF_CONST keyframe strategy, since it provides the best trade-off between efficiency, surface quality and predictability (runtime and memory consumption).

Figure B.2 gives the average runtimes per frame of our full 3D reconstruction framework w.r.t. the number of input frames fused into a keyframe. Our system is split into it's processing steps:

Integrate Adding of a new frame, i.e., either fusion of a frame into the current keyframe or integration of a keyframe into the 3D model.

Update Correcting the model, i.e., the re-integration procedure.

Wait for DVO-SLAM DVO-SLAM runs in parallel to our system. This component represents the time required for synchronization with DVO-SLAM .

Miscellaneous All other required processing.

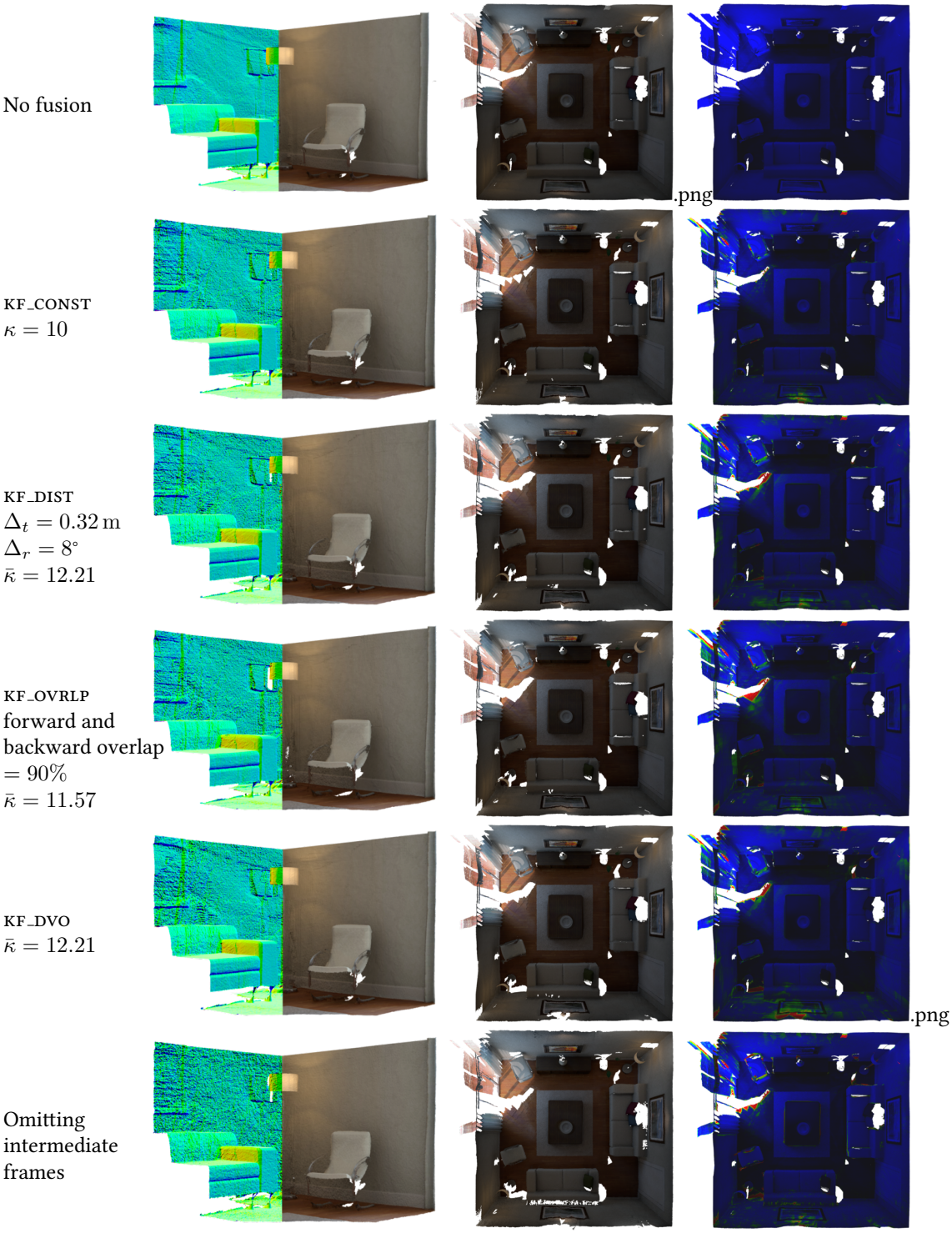


Figure B.1: Reconstruction of noisy *AUG_ICL/Liv1* with different keyframe strategies: Keyframe fusion is generally more efficient, but also results in more noisy reconstructions (left) and less complete 3D models (middle and right).

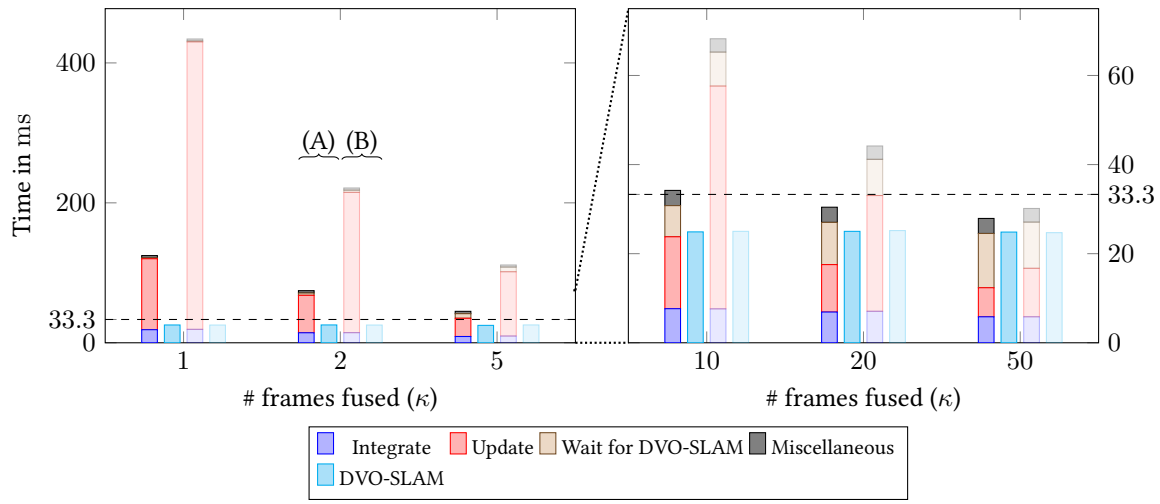


Figure B.2: Average runtime required per frame for reconstruction of *AUG_ICL/Liv1* with KF_CONST keyframe strategy w.r.t. the number of frames fused per keyframe κ. For each pose update, $k = 100/\kappa$ frames were re-integrated. Our system (consisting of the steps Integrate, Update, Wait for DVO-SLAM and Miscellaneous, see text) and DVO-SLAM are executed in parallel, hence resulting in the two bars next to each other. While the strong colored bars (A) represent our system run with our improved re-integration strategy, the washed out bars (B) stand for our system run with BundleFusion's strategy. Note that for better visibility of short runtimes, the scale of the figure's x-axis is adapted between $\kappa = 5$ and $\kappa = 10$. The dashed line shows our goal of 30 fps. Table B.2 contains the raw data of this figure.

For a keyframe size of $\kappa \geq 20$ we achieve real-time performance. DVO-SLAM runs in real-time on a single CPU, asynchronously to our system in a separate thread. Figure B.2 also compares our re-integration strategy to BundleFusion's by Dai et al. [28]. Especially for low κ we obtain a significant speed-up stemming from a reduction of runtime required for *Update*.

The raw data of Figure B.2 is displayed in Table B.2.

B.3 Surface Correction Memory Evaluation

In addition to major runtime improvements during re-integration, keyframe fusion also leads to decreased memory consumption on the host.

Figure B.3 shows the memory consumption of our system for different numbers of frames per keyframe κ. The memory consumption was measured every 10^{th} frame and refers to *RSS* (resident set size); we excluded the memory usage of DVO-SLAM , since our surface reconstruction method is independent of the used SLAM system.

As demonstrated in Figure B.3, the *RSS* increases linearly with progressing surface

Our re-integration strategy					
κ	Integrate	Update	Wait for DVO-SLAM	Miscellaneous	DVO-SLAM
1	18.70	101.40	1.28	3.24	25.45
2	14.31	53.55	3.24	3.39	25.50
5	8.97	26.07	6.55	3.35	24.89
10	7.65	16.14	6.99	3.42	24.89
20	6.90	10.65	9.47	3.41	25.01
50	5.83	6.52	12.18	3.38	24.85
BundleFusion's re-integration strategy					
κ	Integrate	Update	Wait for DVO-SLAM	Miscellaneous	DVO-SLAM
1	19.22	410.83	1.25	2.85	25.32
2	14.42	200.57	3.14	3.06	25.31
5	9.65	91.87	6.68	2.94	25.47
10	7.61	50.01	7.62	2.99	25.01
20	7.07	25.98	8.13	2.98	25.18
50	5.83	10.89	10.37	3.06	24.71

Table B.2: Raw data of Figure B.2. Our re-integration method requires substantially less time for updating, i.e., correcting, the surface on-the-fly compared to BundleFusion's strategy, while all other steps of the pipeline remain essentially the same.

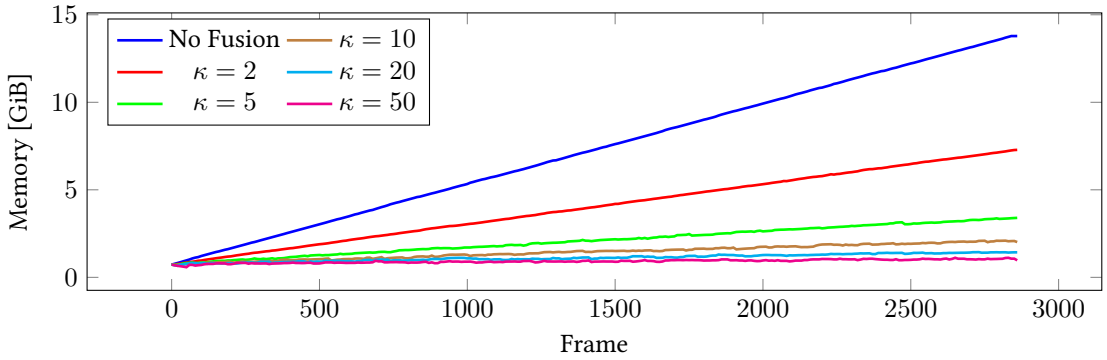


Figure B.3: Host memory consumption of our system (excluding DVO-SLAM). We measured the *RSS* (resident set size) after every 10^{th} frame during reconstruction of *AUG_ICL/Liv1* with KF_CONST.

reconstruction (i.e. with the number of integrated frames). Inversely, the memory consumption decreases linearly with increasing κ. In our framework, most of the memory is used for storing the keyframes for potential later re-integration. When comparing *no keyframe fusion* with KF_CONST with $\kappa = 20$, we save about 90% of host memory (14.5 GiB vs. 1.5 GiB).

B.4 On-the-fly Surface Re-integration

Finally, we present some qualitative examples for on-the-fly surface re-integration on various large-scale datasets.

The following sequences of models were created by reconstructing the RGB-D sequences *TUM/long_office_household*, *BundleFusion/apt0* and *AUG_ICL/Liv1*; the underlying camera poses were provided by DVO-SLAM . With a certain frequency (specified in each figure's caption), a polygon 3D model was generated and later rendered using Blender. In order to compare the outcome with and without re-integration, pairs of renderings are shown, with the left and right image stemming from reconstruction without and with re-integration, respectively. After integration of all frames, the final 3D model was generated, independently of the above frequency. For the run with re-integration, before generation of the final 3D model, *all* frames were re-integrated once to ensure that all pose updates were incorporated in the 3D model.

In particular, Figure B.4 shows the results for *AUG_ICL/Liv1*, while Figure B.5 shows the model for *TUM/long_office_household* and Figure B.6 the reconstruction of *BundleFusion/apt0*.

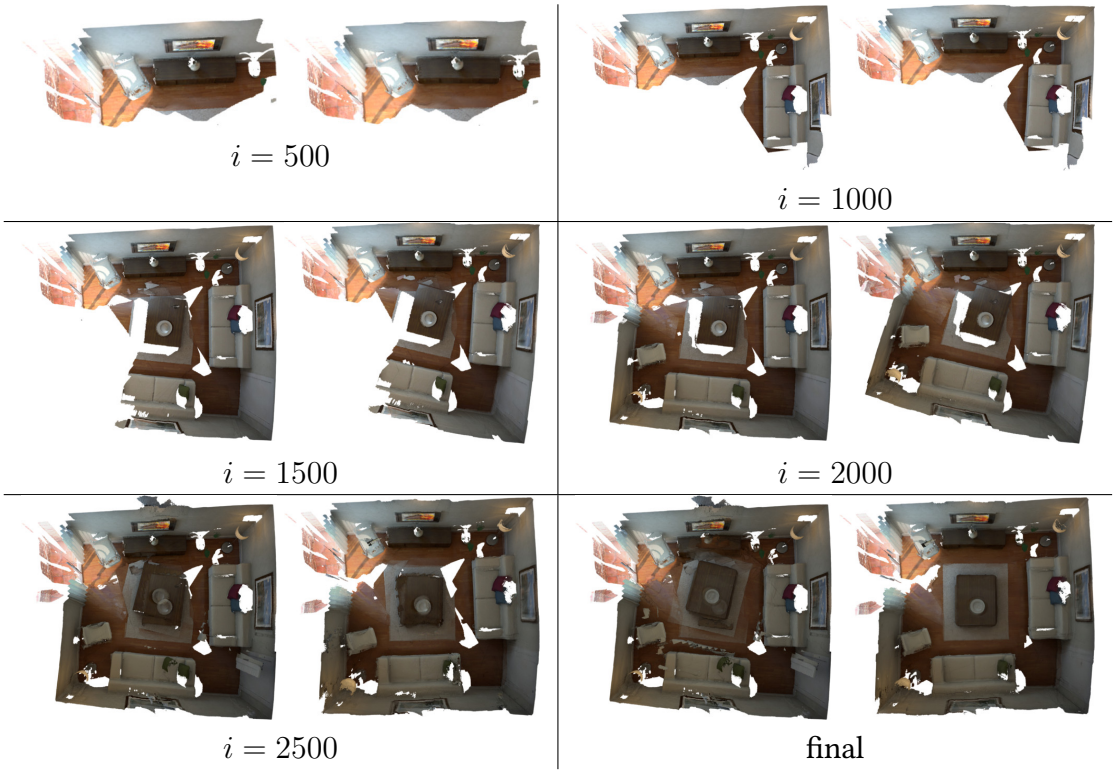


Figure B.4: On-the-fly surface re-integration of *AUG_ICL/Liv1*. Every 500 frames, a model was generated.

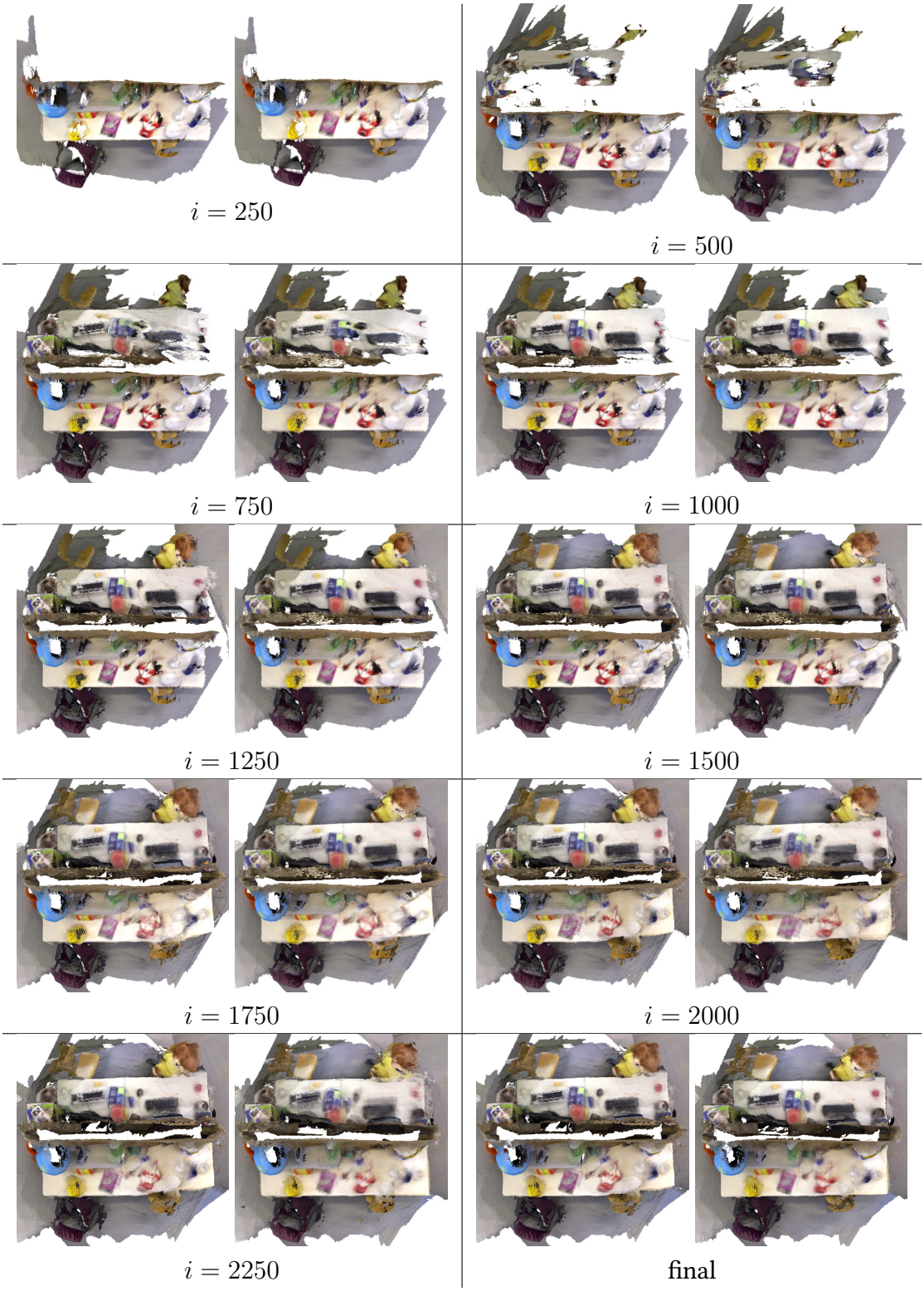


Figure B.5: On-the-fly surface re-integration of *TUM/long_office_household*. Every 250 frames, a model was generated.

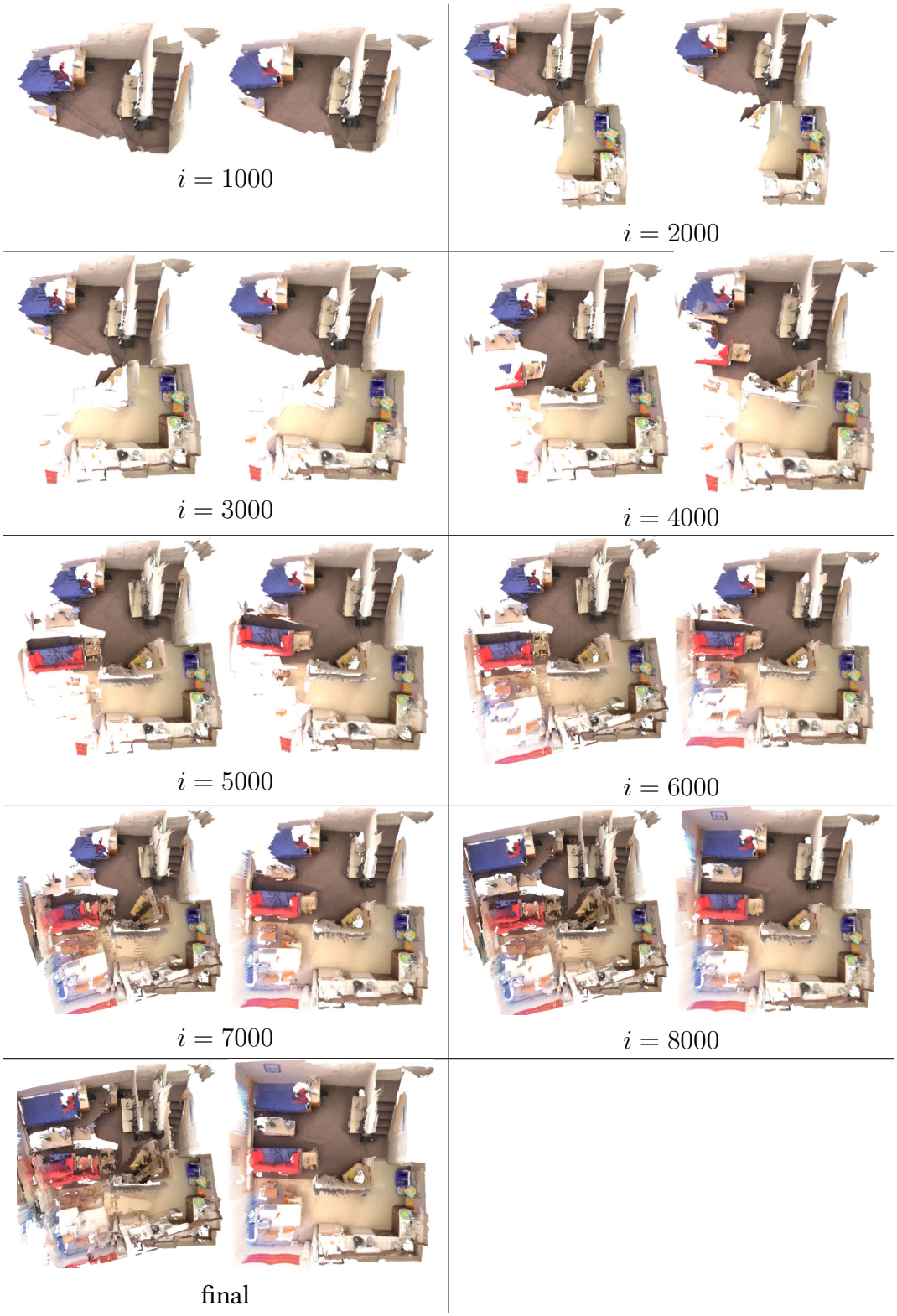


Figure B.6: On-the-fly surface re-integration of *BundleFusion/apt0*. Every 1000 frames, a model was generated.

Appendix C

Original Publications

This chapter includes the original publications of the peer-reviewed papers [4, 5, 6] that contribute to this cumulative dissertation. The works in Chapters 4, 5 and 6 have revised layouts as well as minor content adaptations compared to the original publications. Additionally, we provide a detailed disclaimer for each paper, indicating a copyright notice, a publication summary and the specific individual contributions of the author of this thesis. More precisely, the individual contributions are divided in problem definition, literature survey, implementation, experimental evaluation and manuscript preparation.

C.1 Super-Resolution Keyframe Fusion for 3D Modeling with High-Quality Textures

Copyright

Robert Maier, Jörg Stückler, and Daniel Cremers
Super-Resolution Keyframe Fusion for 3D Modeling with High-Quality Textures
IEEE International Conference on 3D Vision (3DV) 2015
DOI: 10.1109/3DV.2015.66

Summary

We propose a novel fast and robust method for obtaining 3D models with high-quality appearance using commodity RGB-D sensors. Our method uses a direct keyframe-based SLAM frontend to consistently estimate the camera motion during the scan. The aligned images are fused into a volumetric truncated signed distance function representation, from which we extract a mesh. For obtaining a high-quality appearance model, we additionally deblur the low-resolution RGB-D frames using filtering techniques and fuse them into super-resolution keyframes. The meshes are textured from these sharp super-resolution keyframes employing a texture mapping approach. In experiments, we demonstrate that our method achieves superior quality in appearance compared to other state-of-the-art approaches.

Individual contributions

Leading role in realizing the scientific project.

Problem definition	*significantly contributed*
Literature survey	*significantly contributed*
Implementation	*significantly contributed*
Experimental evaluation	*significantly contributed*
Preparation of the manuscript	*significantly contributed*

In accordance with the *IEEE Thesis / Dissertation Reuse Permissions*, we include the accepted version of the original publication [6] in the following.

Super-Resolution Keyframe Fusion for 3D Modeling with High-Quality Textures

Robert Maier, Jörg Stückler, Daniel Cremers
Computer Vision Group, Technische Universität München, Germany
{maierr,stueckle,cremers}@in.tum.de

Abstract

We propose a novel fast and robust method for obtaining 3D models with high-quality appearance using commodity RGB-D sensors. Our method uses a direct keyframe-based SLAM frontend to consistently estimate the camera motion during the scan. The aligned images are fused into a volumetric truncated signed distance function representation, from which we extract a mesh. For obtaining a high-quality appearance model, we additionally deblur the low-resolution RGB-D frames using filtering techniques and fuse them into super-resolution keyframes. The meshes are textured from these sharp super-resolution keyframes employing a texture mapping approach. In experiments, we demonstrate that our method achieves superior quality in appearance compared to other state-of-the-art approaches.

1. Introduction

The wide availability of consumer RGB-D sensors has boosted research in 3D reconstruction in recent years. State-of-the-art methods in 3D model reconstruction yield impressively accurate geometric reconstruction in real-time [16, 24]. Such 3D reconstructions are well suitable for 3D printing [21]. Fast and robust estimation of high-quality visual appearance (i.e. texture) of the models has been given less attention. This plays, however, an equally important role for 3D modeling, for instance, of persons or objects.

Modern texture mapping approaches can obtain good-quality results, but are typically slow and impractical for instant 3D scanning applications. As scanning the 3D geometry with RGB-D sensors is possible in real-time, also the texture mapping process should be fast. We propose a method for fast and accurate reconstruction of geometry as well as appearance. Figure 1 shows a textured 3D model generated from low-resolution (LR) RGB-D input frames with our approach. For geometric reconstruction, we use a direct keyframe-based RGB-D SLAM method in order to estimate the camera trajectory consistently. Using these pose estimates, the individual frames are integrated into a volumetric truncated signed distance function (TSDF) representation, from which a 3D mesh is extracted. For this mesh we find a parametrization suitable for texture mapping. We significantly improve the quality of the generated texture maps through super-resolution (SR) fusion of RGB-D frames and deblurring. Simple weighted median filtering of projected color values onto the texture provides high-quality appearance results.

In experiments, we compare our method to standard pipelines that perform per-vertex coloring or texture mapping based on the original low-resolution frames. We demonstrate superior results of our method with respect to texture quality. We also evaluate the timing of our method and find that it yields high-quality results in reasonable and practical time for 3D scanning applications.

1.1. Related Work

Since the recent advent of low-cost commodity RGB-D sensors, there has been extensive research in the field of dense 3D reconstruction from RGB-D data. While generating highly accurate 3D models from RGB-D data has been investigated intensively, there seems to be a shortage of research in improving the visual appearance of such reconstructions.

To obtain geometrically accurate 3D reconstructions, Newcombe et al. [16] fuse RGB-D frames into a TSDF Volume and perform camera tracking against this model. Sturm et al. [21] developed a similar approach for reconstructing 3D printable models of persons, paired with direct TSDF tracking [1]. Other RGB-D SLAM methods [5, 20, 13, 8] are based on frame-to-(key)frame tracking with trajectory optimization and data fusion into a single model volume. Kerl et al. [9] developed a robust dense visual SLAM system that shows limited drift by combining dense robust visual odometry estimation with pose graph optimization. SLAM systems for reconstructing and mapping large-scale environments have also been developed [17, 19].

The systems presented above can produce models of

(a) RGB input images (b) Vertex colors (c) Texture mapping using super-resolution keyframes

Figure 1: We propose an efficient method for generating high-quality textures from low-resolution RGB-D frames. Our approach significantly improves the visual quality of reconstructed 3D models while it is still fast enough for applicability in real-world 3D scanning scenarios.

high metric precision, however the state-of-the-art for representing the visual appearance in such 3D reconstruction systems is still volumetric averaging of per-vertex colors, such that the color resolution is limited to mesh resolution. Mostly, these vertex colors are computed as a weighted average of the observed colors for the respective vertices [16, 23, 21]. To improve the appearance, weights based on the normals computed from the depth image are employed; to remove further artifacts, pixels close to depth discontinuities are discarded.

However, to create photo-realistic 3D models of real-world objects, the challenging problem of generating and mapping high-quality textures from multiple input color images has been investigated intensively in the field of computer graphics for decades. Without increasing the geometric complexity, textures (usually of higher resolution than the mesh resolution) are mapped onto the mesh to enhance the visual quality, with camera poses assumed to be given. [15, 18] compute the texel colors using a weighted average of the observations in the input color images. Instead of using the weighted average, Coorg and Teller [2] use a color computation scheme based on weighted median to cope with color outliers in the observations. Eisemann et al. [4] correct for inaccuracies in camera poses and calibration using optical flow for mapping images to the texture map. Lempitsky and Ivanov [11] and Gal et al. [6] select a single input view per face and minimize seams, however the approaches suffer from high runtimes because of the computationally expensive combinatorial optimization. Variational super-resolution methods, e.g. by Goldlücke et al. [7], produce compelling results, paired with impractical computation times of several hours in a controlled setup with only a limited number of input views. Waechter et al. [22] texture large-scale scenes reconstructed with Structure-from-Motion, however they rely on high-quality input images and have long computation times with up to 80 minutes per dataset.

The scenario of improving the visual appearance in RGB-D based 3D reconstruction has not been tackled extensively yet. Meilland and Comport [14] fuse low-resolution images into a single high-resolution keyframe by applying a super-resolution technique. The fused keyframes exhibit an impressive level of detail, but the approach does not create a globally consistent 3D model. Recently, Zhou and Koltun [25] have shown that the colors of 3D models obtained from handheld RGB-D cameras can be improved substantially; within several minutes, they alternatingly optimize camera poses and non-rigid correction to correct for imprecise camera localization and for complex distortions resulting from inaccurate geometric models. However, they use vertex colors of an upsampled mesh, leading to an increasingly complex geometry with a still limited resolution compared to texture maps.

To the best of our knowledge, we present the first method for combining keyframe fusion with texture mapping in an RGB-D based 3D reconstruction scenario. Our practical approach is efficient, with runtimes within a few minutes, and suitable for generating high-quality texture maps from low-quality color images obtained from consumer RGB-D sensors.

1.2. Contributions

In summary, we propose a novel fast and robust 3D modeling approach that provides accurate geometry and high-quality appearance. Our method uses direct keyframe-based RGB-D SLAM to find a consistent global image alignment, and extracts a high-quality mesh from a fused TSDF representation of the images.

- The mesh is parametrized in a texture map, which we fill from fused super-resolution RGB-D keyframes.
- The super-resolution RGB-D keyframes are sharpened using image deconvolution.
- Fast texture mapping is performed using the super-resolution keyframes. High quality of the texture is obtained through weighted median filtering of the keyframe projections.

2. 3D Reconstruction System

In this section, we first describe the RGB-D sensor, the acquired data and the used camera model. We then introduce our 3D reconstruction system based on DVO-SLAM by Kerl et al. [9] and the data fusion into a TSDF volume as used by Newcombe et al. [16].

RGB-D Data Acquisition A calibrated Asus Xtion Pro Live RGB-D sensor provides us with RGB color and depth images at 30 fps at a resolution of $w \times h$ (in this case, 640×480 pixels). To limit automatic color correction during data acquisition, we fix exposure and white balance. We assume that depth and color images are registered. Since both color and depth images are utilized for real-time camera tracking, we cannot use the SXGA (1280×1024) color images provided at only 10 fps. We denote RGB images with $\mathcal{C} : \Omega_{\mathcal{C}} \to \mathbb{R}^3$ and depth images with $\mathcal{Z} : \Omega_{\mathcal{Z}} \to \mathbb{R}$.

Camera Model For the RGB-D sensor, we assume the pinhole camera model with focal length f_x, f_y and optical center c_x, c_y. The projection function π maps 3D points $\boldsymbol{p} = (X, Y, Z)^\top$ to 2D pixels $\boldsymbol{x} = (x, y)^\top$:

$$\boldsymbol{x} = \pi(\boldsymbol{p}) = \left(\frac{X}{Z} f_x + c_x, \frac{Y}{Z} f_y + c_y \right), \tag{1}$$

while 2D pixel locations $\boldsymbol{x}$ are mapped back to 3D points using their depth values $\mathcal{Z}(\boldsymbol{x})$ by the inverse projection π^{-1}:

$$\boldsymbol{p} = \pi^{-1}(\boldsymbol{x}, \mathcal{Z}(\boldsymbol{x})) = \left(\frac{x - c_x}{f_x}, \frac{y - c_y}{f_y}, 1 \right)^\top \mathcal{Z}(\boldsymbol{x}). \tag{2}$$

3D Reconstruction Framework DVO-SLAM performs dense camera tracking in real-time on the CPU and minimizes the photometric and geometric error between two RGB-D input frames to compute the relative pose. The use of color images significantly improves camera tracking and limits the drift of the SLAM system. Similarly, we perform an entropy-based loop closure detection and continuously optimize the pose graph in order to obtain a globally consistent camera trajectory.

To reconstruct a dense 3D model in a post-processing step, we fuse the N acquired RGB-D frames into a TSDF volume using their estimated absolute camera poses $\boldsymbol{T}_i = (\boldsymbol{R}, \boldsymbol{t}) \in \mathrm{SE}(3)$ (with $i \in 1 \dots N$, $\boldsymbol{t} \in \mathbb{R}^3$ and $\boldsymbol{R} \in \mathrm{SO}(3)$). We extract a 3D mesh $\mathcal{M} = (\mathcal{V}, \mathcal{F})$ with vertices $\mathcal{V}$ and faces $\mathcal{F}$ using the Marching Cubes algorithm. The camera poses exhibit only very limited drift due to the global pose graph optimization and hence the resulting 3D model is geometrically accurate.

3. Keyframe Fusion

Given an accurate geometric 3D model, reconstructed as described above, and the absolute camera poses for the input frames, we first fuse N_w neighboring frames into a common keyframe representation of higher resolution. We denote the color image of such a SR keyframe as $\mathcal{C}^* : \Omega_{\mathcal{C}} \to \mathbb{R}^3$ and the corresponding depth image as $\mathcal{Z}^* : \Omega_{\mathcal{Z}} \to \mathbb{R}$. To store the depth fusion weights, we introduce a depth weight image $\mathcal{W}^* : \Omega_{\mathcal{W}} \to \mathbb{R}$. These SR keyframes have the dimensions $sw \times sh$, where s is a scale factor that determines the amount of upsampling. We set the pose $\boldsymbol{T}^*$ of the SR keyframe to the first pose of the N_w LR frames to be fused. To integrate the LR images into the SR images, we additionally need to define the scale-dependent projection π_s and inverse projection π_s^{-1}, which use the upscaled intrinsic parameters sf_x, sf_y, sc_x, sc_y.

Depth Fusion We first fuse all N_w LR depth images into the corresponding SR depth image. Therefore, we compute the weights for the measured depth values, which is based on a theoretical random error model [10], as follows:

$$w_z(d) = \frac{fb}{\sigma_d} d^{-2}, \tag{3}$$

with the depth camera's focal length f, baseline b and disparity error standard deviation σ_d. Next, we transform the current depth image i into the keyframe's camera coordinate system using the relative transformation $\boldsymbol{T}^{*-1}\boldsymbol{T}_i$ between them:

$$\boldsymbol{p}^* = (X^*, Y^*, Z^*)^\top = \boldsymbol{T}^{*-1}\boldsymbol{T}_i \pi^{-1}(\boldsymbol{x}, \mathcal{Z}_i(\boldsymbol{x})). \tag{4}$$

We then use the image point $\boldsymbol{x}^* = \pi_s(\boldsymbol{p}^*)$ of the projection into the keyframe depth image to update the fused depth values and depth weights by weighted averaging:

$$\mathcal{Z}^*(\boldsymbol{x}^*) = \frac{\mathcal{W}^*(\boldsymbol{x}^*)\mathcal{Z}^*(\boldsymbol{x}^*) + w_z(\mathcal{Z}_i(\boldsymbol{x}))Z^*}{\mathcal{W}^*(\boldsymbol{x}^*) + w_z(\mathcal{Z}_i(\boldsymbol{x}))} \tag{5}$$

$$\mathcal{W}^*(\boldsymbol{x}^*) = \mathcal{W}^*(\boldsymbol{x}^*) + w_z(\mathcal{Z}_i(\boldsymbol{x})) \tag{6}$$

We achieve sub-pixel precision by updating all four neighboring depth values when transforming and projecting a depth value into the SR depth map. Occlusions are considered by fusing only the closest depth values within a given distance. After integrating all N_w LR depth images, we obtain the fused depth image $\mathcal{Z}^*$ for the SR keyframe.

Color Fusion We use the fused depth image $\mathcal{Z}^*$ to project the SR color image pixels into the LR color images. This allows us to directly look up the observed color values c_i using bilinear interpolation:

$$c_i = \mathcal{C}_i(\pi(\boldsymbol{T}_i^{-1}\boldsymbol{T}^*\pi_s^{-1}(\boldsymbol{x}, \mathcal{Z}_i(\boldsymbol{x})))). \tag{7}$$

For every observation c_i, we also compute its weight

$$w_i^c = B_i w_z(\mathcal{Z}_i(\boldsymbol{x})), \tag{8}$$

where B_i is a measure of blurriness of the color image $\mathcal{C}_i$ according to Crete et al. [3], which downweighs views with strong motion blur. Integrating the depth into the color weights enforces that objects closer to the camera obtain higher weights. We store the observed colors and weights for pixel $\boldsymbol{x}$ in its set of color observations $\mathcal{O}_{\boldsymbol{x}} = \{(c_i, w_i^c)\}$. In order to increase color fidelity, we prune observations from $\mathcal{O}_{\boldsymbol{x}}$ with missing depth values or that are within a window of 7×7 pixels around depth discontinuities. Instead of calculating the weighted mean for averaging the color, we calculate the weighted median, for each color channel separately:

$$\mathcal{C}^*(\boldsymbol{x}) = \arg\min_c \sum_{(c_i, w_i^c) \in \mathcal{O}_{\boldsymbol{x}}} w_i^c \|c - c_i\|. \tag{9}$$

Since we usually have many observations per pixel, the use of weighted median is valid, which results in an overall sharper texture. The median selects the center probable value in the distribution of colors, while the mean would be heavily affected by outliers. Integrating weights into the median allows for incorporating a confidence or a prioritization of the individual color samples.

Before fusing the LR color images into the SR keyframe, we apply a Wiener filter on these LR color images as a preprocessing step. This removes motion blur and notably improves the sharpness of the visual appearance.

Note that we perform the keyframe fusion as a postprocessing step; however, it is reasonable to perform this step online whenever a new keyframe is detected.

4. High-Quality Texture Mapping

In this section, we introduce our method for texture mapping from fused SR keyframes. First, we explain the computation of per-vertex colors based on a weighted median filtering scheme, applicable also for recomputing the vertex colors. We afterwards present our texture mapping approach, in which we compute the texel colors using the weighted median from SR keyframe color images.

4.1. Vertex Color Computation

In order to improve the colors of 3D meshes, a very common approach is to recompute the per-vertex colors of the 3D mesh vertices $v \in \mathcal{V}$. We therefore need to determine the views, in which a vertex is visible. To check if vertex $v \in \mathbb{R}^3$ is visible in view i, we render the mesh $\mathcal{M}$ into a virtual image using its pose $\boldsymbol{T}_i$ and the depth camera intrinsics. v is visible in the image, if its depth value is compatible with the depth in the depth buffer used for rendering. We then get the observed color c_i^v using bilinear interpolation:

$$c_i^v = \mathcal{C}_i(\pi(\boldsymbol{T}_i^{-1} v)). \tag{10}$$

The observation weights w_i^v of vertex v in its input views are computed as follows:

$$w_i^v = \frac{\cos(\theta) B_i}{d^2}, \tag{11}$$

where B_i is again the blurriness measure of color image $\mathcal{C}_i$ and d is the distance from v to the camera corresponding to $\mathcal{C}_i$; θ represents the angle between the vertex normal and the view vector at v for the camera. We store all color observations for vertex v and their respective weights in $\mathcal{O}_v = \{(c_i^v, w_i^v)\}$, observations close to depth discontinuities are discarded.

We can now compute the final vertex color c_v^* as the weighted mean of the observations:

$$c_v^* = \arg\min_{c_v} \sum_{(c_i^v, w_i^v) \in \mathcal{O}_v} w_i^v \|c_v - c_i^v\|^2. \tag{12}$$

Since we assume that each vertex has many observations, we can also compute the final color c_v^* (separately for each color channel) using a weighted median filtering scheme:

$$c_v^* = \arg\min_{c_v} \sum_{(c_i^v, w_i^v) \in \mathcal{O}_v} w_i^v \|c_v - c_i^v\|. \tag{13}$$

Given enough views that observe a vertex, this simple method already improves the mesh colors and results in a more detailed appearance, as demonstrated in Section 5.1.

4.2. Texture Mapping

Based on the introduced weighted median color computation scheme, we employ texture mapping to further improve the appearence of 3D models. In particular, we use the fused SR keyframes of Section 3 for texture mapping, leading to a significantly higher resolved visual appearance. We denote a texture as $\mathcal{T} : \Omega_{\mathcal{T}} \to \mathbb{R}^3$, which stores a color value at every texel $t \in \Omega_{\mathcal{T}}$.

Texture Parametrization For working with texture maps, a three-dimensional mesh needs to be projected onto a planar two-dimensional texture $\mathcal{T}$ first. We beforehand simplify the mesh geometry by decimating the number of mesh triangles. This usually results in larger triangles that can be textured more efficiently with larger patches, while

the geometry is still preserved well. While different planar parametrization methods exist, our approach is in general independent of the chosen parametrization, as long as the mesh faces contain texture coordinates. In practice, we mostly use Least Squares Conformal Maps by Levy et al. [12], or a simple arrangement of the mesh triangles on the texture within a rectangular grid.

Since there is a unique mapping from a texel to its containing face, we can determine the respective surrounding vertices for each texel. The barycentric mapping $\psi : \Omega_{\mathcal{T}} \rightarrow \mathbb{R}^3$ performs a one-to-one mapping from 2D texel coordinates to 3D world coordinates. Using barycentric interpolation, we can compute interpolated 3D vertices v_t corresponding to 2D texels t and vice versa:

$$v_t = \psi(t). \tag{14}$$

Texel Color Computation To compute the texel color for every texel t in the texture map, we employ only the N^* SR keyframes $(\mathcal{C}_l^*, \mathcal{Z}_l^*)$ with camera poses $\boldsymbol{T}_l^*$ (with $l \in 1 \dots N^*$), generated as described in Section 3.

We collect the observations of the texel by first computing its 3D vertex position v_t according to Equation (14). We then determine the set of color observations $\mathcal{O}_t = \{(c_l^t, w_l^t)\}$ for v_t analogous to Equations (10) and (11). From these observations, we compute the final texel colors by again applying a weighted median color computation scheme:

$$\mathcal{T}(t) = \arg\min_{c_t} \sum_{(c_l^t, w_l^t) \in \mathcal{O}_v} w_l^t \| c_t - c_l^t \|. \tag{15}$$

5. Experimental Results

In this section, we evaluated our approach on real-world datasets. Three evaluation sequences *face*, *phone* and *keyboard* were acquired using a handheld Asus Xtion Pro Live, details are given in Table 1. We captured RGB-D data at a low resolution of 640×480 pixels at 30 fps, with fixed exposure and white-balance.

The following experimental results demonstrate that (1) vertex recoloring using weighted median filtering improves the colors of 3D models compared to weighted mean, (2) fusing LR input frames into SR keyframes and using them for texture mapping improves the visual quality substantially, and (3) the proposed method is efficient and practical for real-world 3D scanning applications. All experiments were performed on a standard desktop PC with Intel Core i7-2600 CPU with 3.40GHz and 8GB RAM.

5.1. Vertex Recoloring using Weighted Median

First, we demonstrate that the visual appearance of 3D models can already be improved by using a weighted median color integration scheme. Figure 2 shows that the weighted mean in combination with discontinuity checks already improves the vertex colors significantly compared to unweighted mean. The weighted median increases the sharpness and level of detail even further and leads to a more realistic model. Still, the texture resolution is limited by the number of vertices so far. Mesh subdivision increases the number of vertices, but the increasing geometric mesh complexity makes processing the mesh intractable.

	face	*phone*	*keyboard*
# RGB-D frames	512	1359	642
# vertices (original)	159583	82942	155842
# triangles (original)	319176	165888	311686
# triangles (decimated)	40000	40000	40000

Table 1: Details of the acquired real-world datasets and the corresponding reconstructed 3D meshes.

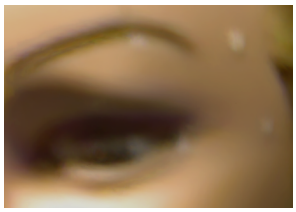
(a) Unweighted mean (b) Weighted mean (c) Weighted median

Figure 2: Improving the vertex colors of 3D models: (a) colors computed using the unweighted mean of the vertex can be improved by (b) using the weighted mean. (c) Applying the weighted median further improves the visual quality and preserves a higher level of detail.

5.2. Keyframe Fusion and Texture Mapping

After showing that a weighted median color computation scheme has advantages compared to weighted mean, we investigate how texture mapping with weighted median filtering further improves the appearence of 3D models. In the following, we show qualitative results of texture mapping from fused SR keyframes in comparison with per-vertex colors, which serves as currently most popular state-of-the-art.

By fusing several LR color images into a SR keyframe, we obtain high-quality frames from low-quality input data.

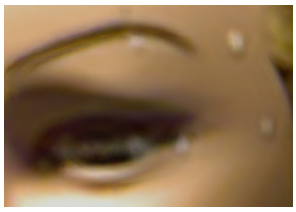


(a) Without deconvolution

(b) With deconvolution

Figure 4: (a) The textures computed from SR keyframes are substantially improved by (b) applying deconvolution (e.g. using a Wiener filter) to the input images before the keyframe fusion.

(a) Keyframes of dimensions 1280×960



(b) Keyframes of dimensions 2560×1920

Figure 5: (a) The textures generated from keyframes of dimensions 1280×960 show slightly fewer details than (b) the ones generated from keyframes of dimensions 2560×1920.

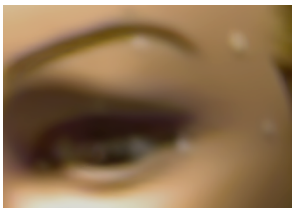
Depending on the scale factor s, the SR images have a resolution of 1280×960 ($s = 2$) or 2560×1920 ($s = 4$). Figure 3 illustrates that both color and depth of the resulting fused SR keyframes exhibit more details compared to the LR input color images and depth maps.

An important aspect of the keyframe fusion is the deconvolution of the input images with a Wiener filter for deblurring. Figure 4 shows the results of generating a texture map for a 3D model from SR keyframes with and without deconvolution. The texture computed from the deblurred SR keyframes (Figure 4b) exhibits a sharper texture with substantially more details compared to Figure 4a. For deblurring, a Wiener filter is applied on the LR input images as a pre-processing step before fusing them into the keyframes.

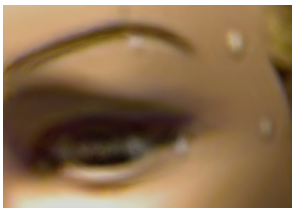
Next, we compare the reconstructed surface colors depending on the scale factor s for the SR keyframe dimensions. The textures shown in Figure 5 show that the level of detail can be slightly improved by using a higher keyframe resolution of 2560×1920 ($s = 4$) compared to a resolution of 1280×960 ($s = 2$).

For comparison, Figure 6 finally shows the improvements of texture mapping with SR keyframes compared to texture mapping with the LR input images only.

To demonstrate the practicability of our approach, we



(a) With LR input frames



(b) With SR keyframes

Figure 6: (a) Texture mapping with LR input frames only yields inferior results compared to (b) texture mapping with SR keyframe fusion.

have reconstructed 3D models of the *face*, *phone* and *keyboard* datasets. All textures have been computed by fusion into SR keyframes of dimensions 2560×1920 and using weighted median filtering for computing the texel colors. As a pre-processing step, a Wiener filter has been applied to the LR input RGB images. Figures 1, 7 and 8 show the results. The texture mapped 3D models provide a photo-realistic appearance and exhibit fine surface details that are not visible in the models with per-vertex colors only.

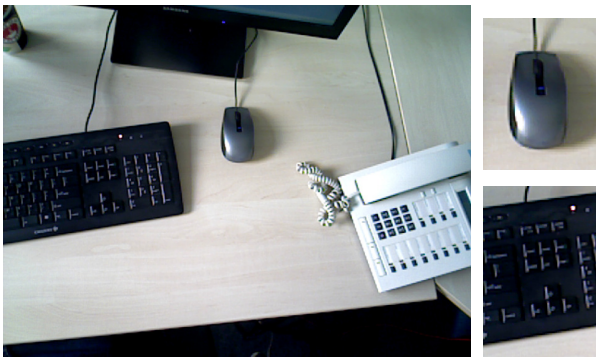
In Figure 8c, the cable at the top of the keyboard is however not represented correctly in the texture. This may either be due to inaccuracies in the estimated camera trajectory or due to an inaccurate geometric model. To compensate for this, an approach similar to Zhou and Koltun [25] must be developed, which optimizes the camera poses as well as non-rigid image corrections.

5.3. Runtime Evaluation

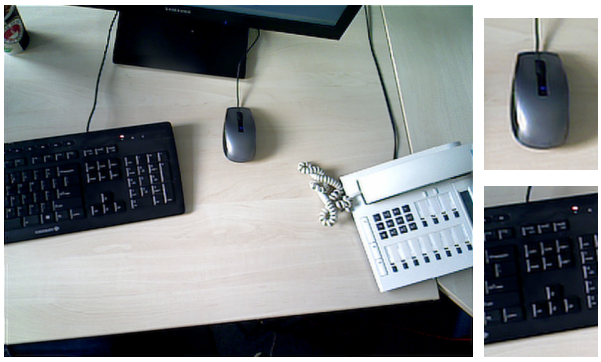
We finally evaluate the runtime and efficiency of the proposed texture mapping method, in particular the runtimes for keyframe fusion and texture mapping. Table 2 gives the results. With runtimes of between one and a few minutes, our approach is a very efficient method for generating high-quality texture maps. Since our implementation is based only on the CPU, a major speed-up can be achieved by porting the algorithm to the GPU. This holds in particular for the keyframe fusion, which has already been shown to work in real-time on a GPU [14].

6. Conclusion

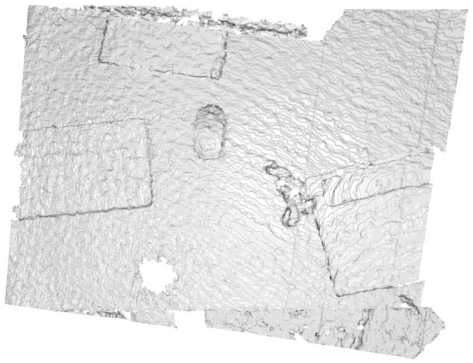
We presented a novel efficient method for high-quality texture mapping in RGB-D-based 3D reconstruction approaches. Our method fuses low-quality color images from commodity depth sensors into super-resolution keyframes. These high-quality keyframes in turn are then mapped into a global texture for the 3D model, resulting in a significantly improved texture quality compared to simple volumetric blending. We deblur input images and use the weighted median for computing the texel colors from observations,



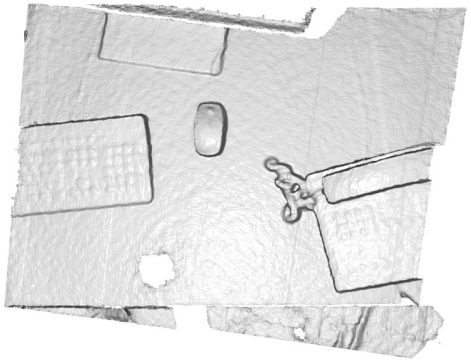
(a) LR input color image



(b) Fused SR color image



(c) LR input depth map (Phong shading)



(d) Fused SR depth map (Phong shading)

Figure 3: Fusing several LR input color images into a single SR keyframe allows to directly obtain high-quality color images. Compared to the LR input color image (a), the fused SR color image (b) with a resolution of 2560×1920 (scale $s = 4$) exhibits more details. Similarly, the LR input depth map (c) shows significantly more noise than the fused SR depth map (d).

		face		*phone*		*keyboard*	
	s	t [s]	fps	t [s]	fps	t [s]	fps
Texture Mapping		91.5	5.6	330.8	4.1	128.8	5.0
Keyframe Fusion	2	57.5	8.9	222.0	6.1	72.1	8.9
SR Texture Mapping	2	18.7	2.8	50.7	2.7	18.8	3.5
Keyframe Fusion	4	100.9	5.1	362.8	2.2	214.9	3.0
SR Texture Mapping	4	26.4	2.0	58.2	1.4	42.6	1.5

Table 2: Runtimes (in seconds) for texture mapping without SR keyframe fusion and texture mapping with SR keyframe fusion.

which preserves a high level of detail. Using the weighted median already provides better results for vertex coloring compared to the weighted mean. The weights in our method consider criteria such as view-angle, motion blur, and distance to the surface.

We have shown in experimental results that our method produces high-quality textures that substantially increase the photo-realism of the reconstructed 3D models. At the same time, our method is a very efficient and practical post-processing step with runtimes within a few minutes, making it useful for real-world 3D scanning application scenarios.

Acknowledgements

This work has been partially funded by the ERC Proof of Concept grant CopyMe3D (GA 632200) and the BMWi ZIM project 2Dzu3D (KF2080213CR4).

References

[1] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-time camera tracking and 3D reconstruction using signed distance functions. In *RSS*, 2013.

[2] S. Coorg and S. Teller. Automatic extraction of textured vertical facades from pose imagery. 1998.

[3] F. Crete, T. Dolmiere, P. Ladret, and M. Nicolas. The blur effect: perception and estimation with a new no-reference perceptual blur metric. In *SPIE*, pages 64920I–64920I, 2007.

(a) RGB input images (b) Vertex colors (c) Texture mapping using SR keyframes

Figure 7: 3D model of the *phone* dataset reconstructed and textured using our approach: We show (a) some input images and (b) the 3D model with vertex colors only. (c) The texture mapped reconstruction provides a significantly more detailed visual appearance.

(a) RGB input images (b) Vertex colors (c) Texture mapping using SR keyframes

Figure 8: 3D model of the *keyboard* dataset reconstructed and textured using our approach: We show (a) some input images and (b) the 3D model with vertex colors only. (c) The texture mapped reconstruction provides a significantly more detailed visual appearance.

[4] M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. de Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating textures. In *Computer Graphics Forum*, volume 27, pages 409–418, 2008.

[5] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *ICRA*, 2012.

[6] R. Gal, Y. Wexler, E. Ofek, H. Hoppe, and D. Cohen-Or. Seamless montage for texturing models. In *Computer Graphics Forum*, volume 29, pages 479–486, 2010.

[7] B. Goldlücke, M. Aubry, K. Kolev, and D. Cremers. A super-resolution framework for high-accuracy multiview reconstruction. *IJCV*, 106(2):172–191, Jan. 2014.

[8] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *ISER*, volume 20, pages 22–25, 2010.

[9] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for RGB-D cameras. In *IROS*, 2013.

[10] K. Khoshelham and S. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[11] V. Lempitsky and D. Ivanov. Seamless mosaicing of image-based texture maps. In *CVPR*. IEEE, 2007.

[12] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (TOG)*, 21(3):362–371, 2002.

[13] R. Maier, J. Sturm, and D. Cremers. Submap-based bundle adjustment for 3D reconstruction from RGB-D data. In *GCPR*, 2014.

[14] M. Meilland and A. Comport. Super-resolution 3D tracking and mapping. In *ICRA*, 2013.

[15] P. Neugebauer and K. Klein. Texturing 3d models of real world objects from multiple unregistered photographic views. In *Computer Graphics Forum*, volume 18, pages 245–256, 1999.

[16] R. Newcombe, A. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[17] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, 32(6):169, 2013.

[18] I. Stamos and P. Allen. 3-d model construction using range and image data. In *CVPR*, 2000.

[19] F. Steinbrücker, J. Sturm, and D. Cremers. Volumetric 3D mapping in real-time on a CPU. In *ICRA*, 2014.

[20] J. Stückler and S. Behnke. Multi-resolution surfel maps for efficient dense 3D modeling and tracking. *Journal of Visual Communication and Image Representation*, 25(1):137–147, 2014.

[21] J. Sturm, E. Bylow, F. Kahl, and D. Cremers. CopyMe3D: Scanning and printing persons in 3D. In *GCPR*, 2013.

[22] M. Waechter, N. Moehrle, and M. Goesele. Let there be color! Large-scale texturing of 3D reconstructions. In *ECCV*. 2014.

[23] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald. Robust real-time visual odometry for dense RGB-D mapping. In *ICRA*, 2013.

[24] C. Wu, M. Zollhöfer, M. Nießner, M. Stamminger, S. Izadi, and C. Theobalt. Real-time shading-based refinement for consumer depth cameras. *SIGGRAPH Asia*, 2014.

[25] Q.-Y. Zhou and V. Koltun. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Transactions on Graphics (TOG)*, 33(4):155, 2014.

C.2 Intrinsic3D: High-Quality 3D Reconstruction by Joint Appearance and Geometry Optimization with Spatially-Varying Lighting

Copyright

Robert Maier, Kihwan Kim, Daniel Cremers, Jan Kautz, and Matthias Niessner
Intrinsic3D: High-Quality 3D Reconstruction by Joint Appearance and Geometry Optimization with Spatially-Varying Lighting
IEEE International Conference on Computer Vision (ICCV) 2017
DOI: 10.1109/ICCV.2017.338

Summary

We introduce a novel method to obtain high-quality 3D reconstructions from consumer RGB-D sensors. Our core idea is to simultaneously optimize for geometry encoded in a signed distance field (SDF), textures from automatically-selected keyframes, and their camera poses along with material and scene lighting. To this end, we propose a joint surface reconstruction approach that is based on Shape-from-Shading (SfS) techniques and utilizes the estimation of spatially-varying spherical harmonics (SVSH) from subvolumes of the reconstructed scene. Through extensive examples and evaluations, we demonstrate that our method dramatically increases the level of detail in the reconstructed scene geometry and contributes highly to consistent surface texture recovery.

Individual contributions

Leading role in realizing the scientific project.

Problem definition	*significantly contributed*
Literature survey	*significantly contributed*
Implementation	*significantly contributed*
Experimental evaluation	*significantly contributed*
Preparation of the manuscript	*significantly contributed*

In accordance with the *IEEE Thesis / Dissertation Reuse Permissions*, we include the accepted version of the original publication [4] in the following.

Intrinsic3D: High-Quality 3D Reconstruction by Joint Appearance and Geometry Optimization with Spatially-Varying Lighting

Robert Maier[1,2] Kihwan Kim[1] Daniel Cremers[2] Jan Kautz[1] Matthias Nießner[2,3]
[1]NVIDIA [2]Technical University of Munich [3]Stanford University

Abstract

We introduce a novel method to obtain high-quality 3D reconstructions from consumer RGB-D sensors. Our core idea is to simultaneously optimize for geometry encoded in a signed distance field (SDF), textures from automatically-selected keyframes, and their camera poses along with material and scene lighting. To this end, we propose a joint surface reconstruction approach that is based on Shape-from-Shading (SfS) techniques and utilizes the estimation of spatially-varying spherical harmonics (SVSH) from subvolumes of the reconstructed scene. Through extensive examples and evaluations, we demonstrate that our method dramatically increases the level of detail in the reconstructed scene geometry and contributes highly to consistent surface texture recovery.

1. Introduction

With the wide availability of commodity RGB-D sensors such as the Microsoft Kinect, Intel RealSense, or Google Tango, reconstruction of 3D scenes has gained significant attention. Along with new hardware, researchers have developed impressive approaches that are able to reconstruct 3D surfaces from the noisy depth measurements of these low-cost devices. A very popular strategy to handle strong noise characteristics is volumetric fusion of independent depth frames [7], which has become the core of many state-of-the-art RGB-D reconstruction frameworks [17, 18, 21, 5, 8].

Volumetric fusion is a fast and efficient solution for regularizing out sensor noise; however, due to its ℓ_2-regularization property, it tends to oversmooth the reconstruction, leaving little fine-scale surface detail in the result. The same problem also translates to reconstruction of surface textures. Most RGB-D reconstruction frameworks simply map RGB values of associated depth pixels onto the geometry by averaging all colors that have been observed for a given voxel. This typically leads to blurry textures, as wrong surface geometry and misaligned poses introduce re-projection errors where one voxel is associated with different color values that are then incorrectly averaged.

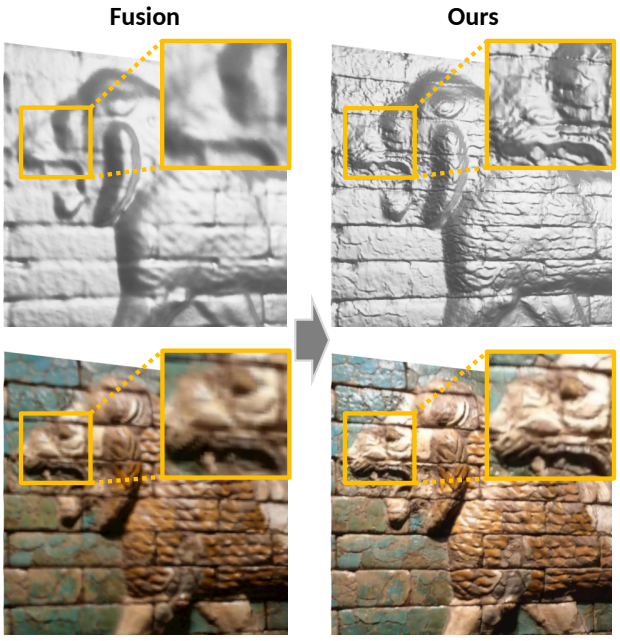


Figure 1. Our 3D reconstruction method jointly optimizes geometry and intrinsic material properties encoded in a Signed Distance Field (SDF), as well as the image formation model to produce high-quality models of fine-detail geometry (top) and compelling visual appearance (bottom).

Very recent approaches address these two problems independently. For instance, Zhou and Koltun [29] optimize for consistent surface textures by iteratively solving for rigid pose alignment and color averages. To compensate for wrong surface geometry where re-projection consistency is infeasible, they non-rigidly warp RGB frames on top of the reconstructed mesh, thus obtaining a high-quality surface texture. On the other end of the spectrum, shading-based refinement techniques enhance depth frames [24] or surface geometry [30] by adding shading constraints from higher resolution color frames; i.e., they leverage RGB signal to refine the geometry. These reconstruction pipelines are sequential; for instance, Zollhöfer et al. [30] first compute the alignment between RGB-D frames, then fuse both RGB

and depth data into a volumetric grid, and finally refine the 3D reconstruction. This results in visually promising reconstructions; however, the pipeline fundamentally cannot recover errors in its early stages; e.g., if pose alignment is off due to wrong depth measures, fused colors will be blurry, causing the following geometry refinement to fail.

In our work, we bring these two directions together by addressing these core problems simultaneously rather than separately. Our main idea is to compute accurate surface geometry such that color re-projections of the reconstructed texture are globally consistent. This leads to sharp surface colors, which can again provide constraints for correct 3D geometry. To achieve this goal, we introduce a novel joint optimization formulation that solves for all parameters of a global scene formation model: (1) surface geometry, represented by an implicit signed distance function, is constrained by input depth measures as well as a shading term from the RGB frames; (2) correct poses and intrinsic camera parameters are enforced by global photometric and geometric consistency; (3) surface texture inconsistency is minimized considering all inputs along with the 3D model; and (4) spatially-varying lighting as well as surface albedo values are constrained by RGB measures and surface geometry. The core contribution of our work is to provide a parametric model for all of these intrinsic 3D scene parameters and optimize them in a joint, continuous energy minimization for a given RGB-D sequence. As a result, we achieve both sharp color reconstruction, highly-detailed and physically-correct surface geometry (Figure 1), and an accurate representation of the scene lighting along with the surface albedo. In a series of thorough evaluations, we demonstrate that our method outperforms state-of-the-art approaches by a significant margin, both qualitatively and quantitatively.

To sum up, our technical contributions are as follows:

- We reconstruct a volumetric signed distance function by jointly optimizing for 3D geometry, surface material (albedo), camera poses, camera intrinsics (including lens distortion), as well as accurate scene lighting using spherical harmonics basis functions.
- Instead of estimating only a single, global scene illumination, we estimate spatially-varying spherical harmonics to retrieve accurate scene lighting.
- We utilize temporal view sampling and filtering techniques to mitigate the influence of motion blur, thus efficiently handling data from low-cost consumer-grade RGB-D sensor devices.

2. Related Work

3D Reconstruction using Signed Distance Functions Implicit surface representations have been widely used in 3D modeling and reconstruction algorithms. In particular, signed distance fields (SDF) [7] are often used to encode 3D surfaces in a voxel grid, and have become the basis of many successful RGB-D surface reconstruction algorithms [17, 18]. More recently, Choi et al. [5] propose a robust optimization for high-quality pose alignment using only geometry, and Dai et al. [8] present a global optimization for large-scale scenes in real time. While most SDF-based fusion methods efficiently regularize noisy depth input, they spend little focus on reconstructing consistent and sharp surface textures. In particular, in the context of wide baseline views and small surface misalignments, this leads to blurry voxel colors that are obtained by averaging the input RGB values of associated color images.

High-quality texture recovery In order to compute consistent colors on the reconstructed surface, Zhou and Koltun [29] introduce a method to optimize the mapping of colors onto the geometry (camera poses and 2D deformation grid), Klose et al. [13] propose to filter colors in scene space, and Jeon et al. [12] suggest a more efficient way of color optimization through texture coordinates. In addition to directly optimizing for consistent surface textures, refining texture quality also helps to improve the quality of reconstructed surface colors [16, 9]. While these methods achieve visually impressive RGB reconstructions (e.g., by warping RGB input), they do not address the core problem of color inconsistency, which is caused by wrong surface geometry that leads to inconsistent RGB-to-RGB and RGB-to-geometry re-projections.

Shading- and reflectance-based geometry refinement Shape-from-Shading [11, 28] aims to extract 3D geometry from a single RGB image, and forms the mathematical basis of shading-based refinement, targeted by our work. The theory behind Shape-from-Shading is well-studied, in particular when the surface reflectance, light source and camera locations are known. Unfortunately, the underlying optimizations are highly under-constrained, particularly in uncontrolled environments. Thus, one direction is to refine coarse image-based shape models based on incorporation of shading cues [4]. For instance, this can be achieved with images captured by multiple cameras [23, 22] or with RGB-D cameras that provide an initial depth estimate for every pixel [10, 26, 2].

Hence, shading and reflectance estimation has become an important contextual cue for refining geometry. Many methods leverage these cues to develop high-quality surface refinement approaches [24, 19, 3]. In particular, Zollhöfer et al. [30] motivates our direction of using volumetric signed distance fields to represent the 3D model. Unfortunately, the method has significant drawbacks; first, it only assumes a single global lighting setting based on spherical harmonics [20] that is constant over the entire scene; second, its pipeline is sequential, meaning that poses and surface colors are optimized only once in a pre-process,

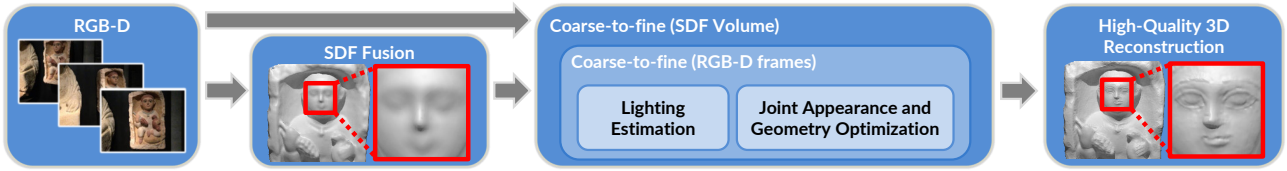


Figure 2. Overview of our method for joint appearance and geometry optimization. Our pipeline takes RGB-D data of a scene as input and fuses it into a Signed Distance Field (SDF). In a nested coarse-to-fine approach, spatially-varying lighting is estimated and used to jointly optimize for appearance and geometry of the scene, producing a high-quality 3D model.

suffering from erroneous depth measures and small pose misalignments. In our approach, we systematically address these shortcomings with a joint optimization strategy, as well as a much more flexible spatially-varying lighting parametrization. Other related methods focus on specular surfaces with an alternating optimization strategy [25], represent lighting with illumination maps [14], or retrieve a box-like 3D representation with material parameters [27].

3. Overview

Our method first estimates a coarse sparse Signed Distance Field (SDF) similar to Nießner et al. [18] from an input RGB-D sequence with initial camera poses. To mitigate the influence of views with motion blur, we automatically select views based on a blurriness measure and constrain the optimization only based on color values from these keyframes.

Our joint optimization employs a nested hierarchical approach (see Figure 2): in an outer loop, we refine the SDF in a coarse-to-fine manner on multiple SDF grid pyramid levels in order to reconstruct fine detail. At the coarsest grid pyramid level, we use multiple RGB-D frame pyramid levels of all keyframes obtained through downsampling in order to improve the convergence and robustness of the joint camera pose estimation.

Within each inner iteration, we approximate complex scene lighting by partitioning the SDF volume into subvolumes of fixed size with separate spherical harmonics parameters. During estimation, we jointly solve for all SH parameters on a global scale with a Laplacian regularizer. The lighting at a given point is defined as the trilinear interpolation of the associated subvolumes.

In the main stage of our framework, we employ the estimated illumination to jointly refine surface and albedo of the SDF as well as the image formation model (camera poses of the input frames, camera intrinsics and lens distortion). As a consequence of this extensive set of optimized parameters, we implicitly obtain optimal colors. We re-compute the voxel colors from the keyframes using the refined parameters after each optimization. Finally, a 3D mesh is extracted from the refined SDF using Marching Cubes [15].

3.1. Signed Distance Field

At the core of our framework lies the reconstructed surface, which we implicitly store as a sparse Truncated Signed Distance Function (TSDF) [7], denoted by $\mathbf{D}$. Hereby, each voxel stores the raw (truncated) signed distance to the closest surface $\mathbf{D}(\boldsymbol{v})$, its color $\mathbf{C}(\boldsymbol{v})$, an integration weight $\mathbf{W}(\boldsymbol{v})$, an illumination albedo $\mathbf{a}(\boldsymbol{v})$, and an optimized signed distance $\tilde{\mathbf{D}}(\boldsymbol{v})$. We denote the current estimate of the iso-surface by $\mathbf{D}_0$ and the number of voxels in the SDF volume by N.

Following state-of-the-art reconstruction methods, we integrate depth maps into the SDF using a weighted running average scheme:

$$\mathbf{D}(\boldsymbol{v}) = \frac{\sum_{i=1}^{M} w_i(\boldsymbol{v}) d_i(\boldsymbol{v})}{\mathbf{W}(\boldsymbol{v})}, \quad \mathbf{W}(\boldsymbol{v}) = \sum_{i=1}^{M} w_i(\boldsymbol{v}), \tag{1}$$

with sample integration weight $w_i(\boldsymbol{v}) = \cos(\theta)$, based on the angle θ between the viewing direction and the normal computed from the input depth map. The truncated signed distance $d_i(\boldsymbol{v})$ between a voxel and a depth frame $\mathcal{Z}_i$ with pose $\mathcal{T}_i$ is computed as follows:

$$d_i(\boldsymbol{v}) = \Psi((\mathcal{T}_i^{-1}\boldsymbol{v})_z - \mathcal{Z}_i(\pi(\mathcal{T}_i^{-1}\boldsymbol{v}))), \tag{2}$$

with truncation $\Psi(d) = \min(|d|, t_{\text{trunc}}) \cdot \text{sgn}(d)$. After integrating all frames of the RGB-D sequence in the implicit 3D model representation, we initialize the optimized SDF $\tilde{\mathbf{D}}$ with the integrated SDF $\mathbf{D}$. We directly compute the surface normal for each voxel from the gradient of the refined signed distance field using forward differences:

$$\mathbf{n}(\boldsymbol{v}) = (n_x, n_y, n_z)^\top = \frac{\nabla\tilde{\mathbf{D}}(\boldsymbol{v})}{||\nabla\tilde{\mathbf{D}}(\boldsymbol{v})||_2}, \tag{3}$$

with the gradient

$$\nabla\tilde{\mathbf{D}}(\boldsymbol{v}) = \nabla\tilde{\mathbf{D}}(i,j,k) = \begin{pmatrix} \tilde{\mathbf{D}}(i+1,j,k) - \tilde{\mathbf{D}}(i,j,k) \\ \tilde{\mathbf{D}}(i,j+1,k) - \tilde{\mathbf{D}}(i,j,k) \\ \tilde{\mathbf{D}}(i,j,k+1) - \tilde{\mathbf{D}}(i,j,k) \end{pmatrix} \tag{4}$$

where $\tilde{\mathbf{D}}(i,j,k)$ is the optimized distance value at the (discrete) voxel location (i,j,k). Since each voxel encodes the distance to its closest surface, it is possible to derive a corresponding 3D point on the iso-surface $\boldsymbol{v}_0$. Thus, the voxel center point $\boldsymbol{v}_c \in \mathbb{R}^3$ in world coordinates is projected onto the (nearest) iso-surface using the transformation ψ:

$$\boldsymbol{v}_0 = \psi(\boldsymbol{v}) = \boldsymbol{v}_c - \mathbf{n}(\boldsymbol{v})\tilde{\mathbf{D}}(\boldsymbol{v}). \tag{5}$$

3.2. Image Formation Model and Sampling

RGB-D Data As input, our framework takes M RGB-D frames with registered color images $\mathcal{C}_i$, derived intensity images $\mathcal{I}_i$, and depth maps $\mathcal{Z}_i$ (with $i \in 1 \dots M$). We assume exposure and white balance of the sensor to be fixed, which is a common setting in RGB-D sensors. Moreover, we are given an initial estimate of the absolute camera poses $\mathcal{T} = \{\mathcal{T}_i\}$ of the respective frames, with $\mathcal{T}_i = (\boldsymbol{R}_i, \boldsymbol{t}_i) \in \mathrm{SE}(3)$, $\boldsymbol{R}_i \in \mathrm{SO}(3)$ and $\boldsymbol{t}_i \in \mathbb{R}^3$. We denote the transformation of a point $\boldsymbol{p}$ using a pose $\mathcal{T}_i$ by $g(\mathcal{T}_i, \boldsymbol{p}) = \boldsymbol{R}_i\boldsymbol{p} + \boldsymbol{t}_i$. While our approach is based on the VoxelHashing framework [18], the initial camera poses can in principle be computed using any state-of-the-art RGB-D based 3D reconstruction system; e.g., [5, 8].

Camera Model Our camera model is defined by the focal length f_x, f_y, the optical center c_x, c_y and three coefficients $\kappa_1, \kappa_2, \rho_1$ describing radial and tangential lens distortion respectively. 3D points $\boldsymbol{p} = (X, Y, Z)^\top$ are mapped to 2D image pixels $\boldsymbol{x} = (x, y)^\top$ with the projection function $\pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$.

Keyframe Selection In hand-held RGB-D scanning, input images often exhibit severe motion blur due to fast camera motion. To mitigate the effect of motion blur, we discard bad views by selecting views using the blurriness measure by Crete et al. [6]. More specifically, we choose the least blurred frame within a fixed size window of t_{KF} neighboring frames. We set $t_{\text{KF}} = 20$ for regular datasets that are captured with commodity RGB-D sensors, and $t_{\text{KF}} = 5$ for short sequences with less than 100 frames. Our method can also be applied to multi-view stereo datasets consisting of only few images; here, we use all frames (i.e., $t_{\text{KF}} = 1$).

Observations Sampling and Colorization After generating the SDF volume, we initially compute the voxel colors by sampling the selected keyframes. Given a frame $(\mathcal{C}_i, \mathcal{Z}_i)$ and its pose $\mathcal{T}_i$, we re-compute the color of a voxel $\boldsymbol{v}$ by sampling its 3D iso-surface point $\boldsymbol{v}_0$ in the input views. To check whether voxel $\boldsymbol{v}$ is visible in view i, we transform $\boldsymbol{v}_0$ back into the input view's coordinate system using the (refined) pose $\mathcal{T}_i$, project it into its depth map $\mathcal{Z}_i$ and look up the respective depth value. $\boldsymbol{v}$ is considered visible in the image if the voxel's z-coordinate in the camera coordinate system is compatible with the sampled depth value.

We collect all color observations of a voxel in its views and their respective weights in $\mathcal{O}_{\boldsymbol{v}} = \{(c_i^{\boldsymbol{v}}, w_i^{\boldsymbol{v}})\}$. The observed colors $c_i^{\boldsymbol{v}}$ are obtained by sampling from the input color image $\mathcal{C}_i$ using bilinear interpolation:

$$c_i^v = \mathcal{C}_i(\pi(\mathcal{T}_i^{-1}\boldsymbol{v}_0)). \tag{6}$$

The observation weight $w_i^{\boldsymbol{v}}$ is view-dependent on both normal and depth in the view:

$$w_i^{\boldsymbol{v}} = \frac{\cos(\theta)}{d^2}, \tag{7}$$

where d is the distance from $\boldsymbol{v}$ to the camera corresponding to $\mathcal{C}_i$. θ represents the angle between the voxel normal $\mathbf{n}(\boldsymbol{v})$ rotated into the camera coordinate system, and the view direction of the camera.

Colorization We sort the observations in $\mathcal{O}_{\boldsymbol{v}}$ by their weight and keep only the best t_{best} observations. The voxel color $c_{\boldsymbol{v}}^*$ is computed as the weighted mean of its observations $\mathcal{O}_{\boldsymbol{v}}$ (for each color channel independently):

$$c_{\boldsymbol{v}}^* = \arg\min_{c_{\boldsymbol{v}}} \sum_{(c_i^{\boldsymbol{v}}, w_i^{\boldsymbol{v}}) \in \mathcal{O}_{\boldsymbol{v}}} w_i^{\boldsymbol{v}} (c_{\boldsymbol{v}} - c_i^{\boldsymbol{v}})^2. \tag{8}$$

Note that the per-voxel colors are only used before each optimization step (for up-to-date chromaticity weights) and as a final postprocess during mesh extraction. The optimization itself directly constrains the input RGB images of the selected views and does not use the per-voxel color values.

4. Lighting Estimation using Spatially-varying Spherical Harmonics

Lighting Model In order to represent the lighting of the scene, we use a fully-parametric model that defines the shading at every surface point w.r.t. global scene lighting. To make the problem tractable, we follow previous methods and assume that the scene environment is Lambertian.

The shading $\mathbf{B}$ at a voxel $\boldsymbol{v}$ is then computed from the voxel surface normal $\mathbf{n}(\boldsymbol{v})$, the voxel albedo $\mathbf{a}(\boldsymbol{v})$ and scene lighting parameters l_m:

$$\mathbf{B}(\boldsymbol{v}) = \mathbf{a}(\boldsymbol{v}) \sum_{m=1}^{b^2} l_m H_m(\mathbf{n}(\boldsymbol{v})), \tag{9}$$

with shading basis H_m. As Equation 9 defines the forward shading computation, our aim is to tackle the inverse rendering problem by estimating the parameters of $\mathbf{B}$.

Spherical Harmonics In order to estimate the reflected irradiance $\mathbf{B}$ (cf. Equation 9) at a voxel $\boldsymbol{v}$, we parametrize the lighting with spherical harmonics (SH) basis functions [20], which is known to be a good approximation and smooth for Lambertian surface reflectance. The SH basis functions H_m are parametrized by a unit normal $\mathbf{n}$. In our implementation, we use SH coefficients up to the second order, which includes $b = 3$ SH bands and leaves us with nine unknown lighting coefficients $\ell = (l_1, \dots, l_{b^2})$. For a given surface point, the SH basis encodes the incident lighting, parameterized as a spherical distribution. However, a single SH basis cannot faithfully represent scene lighting for all surface points simultaneously, as lights are assumed to be infinitesimally far away (i.e., purely directional), and neither visibility nor occlusion is taken into account.

Subvolume Partitioning To address the shortcoming of a single, global spherical harmonics basis that globally defines the scene lighting, we extend the traditional formulation. To this end, we partition the reconstruction volume

into subvolumes $\mathcal{S} = \{s_1 \dots, s_K\}$ of fixed size t_{sv}; the number of subvolumes is denoted as K. We now assign an SH basis – each with its own SH coefficients – to every subvolume. Thus, we substantially increase the number of lighting parameters per scene and allow for spatially-adaptive lighting changes. In order to avoid aliasing artifacts at subvolume boundaries, we define the global lighting function as a trilinear interpolation of local SH coefficients; i.e., for a voxel, we obtain a smooth function defining the actual SH coefficients as an interpolation of the lighting parameters of its eights adjacent subvolumes.

Spatially-varying Spherical Harmonics The ability of subvolumes to define local spherical harmonics coefficients along with a global interpolant introduces the concept of spatially-varying spherical harmonics (SVSH). Instead of only representing lighting with a single set of SH coefficients, we have now $K \times b^2$ unknown parameters, that provide for significantly more expressibility in the scene lighting model. The lighting for subvolumes is estimated by minimizing the following objective:

$$E_{\text{lighting}}(\ell_1, \dots, \ell_K) = E_{\text{appearance}} + \lambda_{\text{diffuse}} E_{\text{diffuse}}. \quad (10)$$

The intuition is that we try to approximate complex global illumination with varying local illumination models for smaller subvolumes. We estimate the spherical harmonics in a subvolume by minimizing the differences between the measured averaged voxel intensity and the estimated appearance:

$$E_{\text{appearance}} = \sum_{\boldsymbol{v} \in \tilde{\mathbf{D}}_0} (\mathbf{B}(\boldsymbol{v}) - \mathbf{I}(\boldsymbol{v}))^2, \quad (11)$$

where only voxels close to the current estimate of the iso-surface $\tilde{\mathbf{D}}_0$ are considered. Initially, we assume the albedo to be constant. However, the albedo is refined as the optimization commences. After the surface refinement on each level, we recompute the voxel colors (and hence voxel intensity). We further regularize the distribution of lighting coefficients with a Laplacian regularizer that considers the 1-ring neighborhood $\mathcal{N}_s$ of a subvolume s, thus effectively constraining global smoothness of the spherical harmonics:

$$E_{\text{diffuse}} = \sum_{s \in \mathcal{S}} \sum_{r \in \mathcal{N}_s} (\ell_s - \ell_r)^2. \quad (12)$$

5. Joint Optimization of Geometry, Albedo, and Image Formation Model

One of the core ideas of our method is the joint optimization of the volumetric 3D reconstruction as well as the image formation model. In particular, we simultaneously optimize for the signed distance and albedo values of each voxel of the volumetric grid, as well as the camera poses and camera intrinsics such as focal length, center pixel, and (radial and tangential) lens distortion coefficients. We stack all parameters in the unknown vector

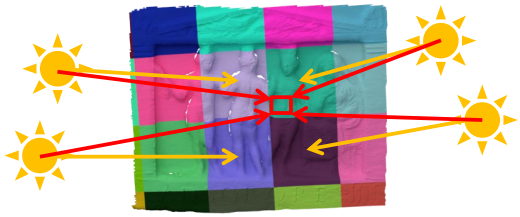


Figure 3. We partition the SDF volume into subvolumes of fixed size and estimate independent spherical harmonics (SH) coefficients for each subvolume (yellow). Per-voxel SH coefficients are obtained through tri-linear interpolation of the lighting of neighboring subvolumes (red).

$\mathcal{X} = (\mathcal{T}, \tilde{\mathbf{D}}, \mathbf{a}, f_x, f_y, c_x, c_y, \kappa_1, \kappa_2, \rho_1)$ and formulate our minimization objective as follows:

$$E_{\text{scene}}(\mathcal{X}) = \sum_{\boldsymbol{v} \in \tilde{\mathbf{D}}_0} \lambda_g E_g + \lambda_v E_v + \lambda_s E_s + \lambda_a E_a, \quad (13)$$

with λ_g, λ_v, λ_s, λ_a the weighting parameters that define the influence of each cost term. For efficiency, we only optimize voxels within a thin shell close to the current estimate of the iso-surface $\tilde{\mathbf{D}}_0$, i.e., $|\tilde{\mathbf{D}}| < t_{\text{shell}}$.

5.1. Camera Poses and Camera Intrinsics

For initial pose estimates, we use poses obtained by the frame-to-model tracking of VoxelHashing [18]. However, this merely serves as an initialization of the non-convex energy landscape for our global pose optimization, which is performed jointly along with the scene reconstruction (see below). In order to define the underlying residuals of the energy term, we project each voxel into its associated input views by using the current state of the estimated camera parameters. These parameters involve not only the extrinsic poses, but also the pinhole camera settings defined by focal length, pixel center, and lens distortion parameters. During the coarse-to-fine pyramid optimization, we derive the camera intrinsics according to the resolution of the corresponding pyramid levels.

5.2. Shading-based SDF Optimization

In order to optimize for the 3D surface that best explains the re-projection and follows the RGB shading cues, we directly solve for the parameters of the refined signed distance field $\tilde{\mathbf{D}}$, which is directly coupled to the shading through its surface normals $\mathbf{n}(\boldsymbol{v})$. In addition to the distance values, the volumetric grid also contains per-voxel albedo parameters, which again is coupled with the lighting computation (cf. Equation 9); the surface albedo is initialized with a uniform constant value. Although this definition of solving for a distance field follows the direction of Zollhöfer et al. [30], it is different at its core: here, we dynamically constrain the reconstruction with the RGB input images, which contrasts Zollhöfer et al. who simply rely on the initially precomputed per-voxel colors. In the following, we introduce all terms of the shading-based SDF objective.

Gradient-based Shading Constraint In our data term, we want to maximize the consistency between the estimated shading of a voxel and its sampled observations in the corresponding intensity images. Our objective follows the intuition that high-frequency changes in the surface geometry result in shading cues in the input RGB images, while more accurate geometry and a more accurate scene formation model result in better sampling of input images.

We first collect all observations in which the iso-surface point $\psi(\boldsymbol{v})$ of a voxel $\boldsymbol{v}$ is visible; we therefore transform the voxel into each frame using the pose $\mathcal{T}_i$ and check whether the sampled depth value in the respective depth map $\mathcal{Z}_i$ is compatible. We collect all valid observations $\mathcal{O}_{\boldsymbol{v}}$, sort them according to their weights $w_i^{\boldsymbol{v}}$ (cf. Equation 7), and keep only the best t_{best} views $\mathcal{V}_{\text{best}} = \{\mathcal{I}_i\}$. Our objective function is defined as follows:

$$E_g(\boldsymbol{v}) = \sum_{\mathcal{I}_i \in \mathcal{V}_{\text{best}}} w_i^{\boldsymbol{v}} \|\nabla \mathbf{B}(\boldsymbol{v}) - \nabla \mathcal{I}_i(\pi(v_i))\|_2^2, \quad (14)$$

where $v_i = g(\mathcal{T}_i, \psi(\boldsymbol{v}))$ is the 3D position of the voxel center transformed into the view's coordinate system. Observations are weighted with their view-dependent observation weights $w_i^{\boldsymbol{v}}$. By transforming and projecting a voxel $\boldsymbol{v}$ into its associated input intensity images $\mathcal{I}_i$, our joint optimization framework optimizes for all parameters of the scene formation model, including camera poses, camera intrinsics, and lens distortion parameters. The shading $\mathbf{B}(\boldsymbol{v})$ depends on both surface and material parameters and allows to optimize for signed distances, implicitly using the surface normals, and voxel albedo on-the-fly. Instead of comparing shading and intensities directly, we achieve improved robustness by comparing their gradients, which we obtain by discrete forward differences from its neighboring voxels.

To improve convergence, we compute an image pyramid of the input intensity images and run the optimization in a coarse-to-fine manner for all levels. This inner loop is embedded into a coarse-to-fine grid optimization strategy, that increases the resolution of the SDF with each level.

Regularization We add multiple cost terms to regularize our energy formulation required for the ill-posed problem of Shape-from-Shading and to mitigate the effect of noise.

First, we use a Laplacian smoothness term to regularize our signed distance field. This volumetric regularizer enforces smoothness in the distance values between neighboring voxels:

$$E_v(\boldsymbol{v}) = (\Delta \tilde{\mathbf{D}}(\boldsymbol{v}))^2. \quad (15)$$

To constrain the surface and keep the refined reconstruction close to the regularized original signed distances, we specify a surface stabilization constraint:

$$E_s(\boldsymbol{v}) = (\tilde{\mathbf{D}}(\boldsymbol{v}) - \mathbf{D}(\boldsymbol{v}))^2. \quad (16)$$

Given spherical harmonics coefficients, the shading computed at a voxel depends on both its albedo as well as its surface normal. We constrain to which degree the albedo or normal should be refined by introducing an additional term that regularizes the albedo. In particular, the 1-ring neighborhood $\mathcal{N}_{\boldsymbol{v}}$ of a voxel is used to constrain albedo changes based on the chromaticity differences of two neighboring voxels. This follows the idea that chromaticity changes often go along with changes of intrinsic material:

$$E_a(\boldsymbol{v}) = \sum_{\boldsymbol{u} \in \mathcal{N}_{\boldsymbol{v}}} \phi(\mathbf{\Gamma}(\boldsymbol{v}) - \mathbf{\Gamma}(\boldsymbol{u})) \cdot (\mathbf{a}(\boldsymbol{v}) - \mathbf{a}(\boldsymbol{u}))^2, \quad (17)$$

where the voxel chromaticity $\mathbf{\Gamma} = \mathbf{C}(\boldsymbol{v})/\mathbf{I}(\boldsymbol{v})$ is directly computed from the voxel colors and $\phi(x)$ is a robust kernel with $\phi(x) = 1/(1 + t_{\text{rob}} \cdot x)^3$.

5.3. Joint Optimization Problem

We jointly solve for all unknown scene parameters stacked in the unknown vector $\mathcal{X}$ by minimizing the proposed highly non-linear least squares objective:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} E_{\text{scene}}(\mathcal{X}) \quad (18)$$

We solve the optimization using the well-known *Ceres Solver* [1], which provides automatic differentiation and an efficient Levenberg-Marquardt implementation.

By jointly refining the SDF and image formation model, we implicitly obtain optimal colors for the reconstruction at minimal re-projection error. In the optimization, the color and shading constraints are directly expressed with respect to associated input images; however, for the final mesh generation, we recompute voxel colors in a postprocess after the optimization. Finally, we extract a mesh from the refined signed distance field using Marching Cubes [15].

6. Results

We evaluated our approach on publicly available RGB-D datasets as well as on own datasets acquired using a Structure Sensor; Table 1 gives an overview. For *Lucy* and *Relief* we used the camera poses provided with the datasets as initializations, while we estimated the poses using Voxel Hashing [18] for all other datasets. Our evaluations were performed on a workstation with Intel Core i7-5930 CPU with 3.50GHz and 32GB RAM.

We used $\lambda_{\text{diffuse}} = 0.01, \lambda_g = 0.2, \lambda_v = 160 \rightarrow 20, \lambda_s = 120 \rightarrow 10, \lambda_a = 0.1$ for our evaluations, with $a \rightarrow b$ indicating changing weights with every iteration. For objects with constant albedo, we fixed the albedo; i.e., we set $\lambda_a = \infty$. We used three RGB-D frame pyramid levels and three grid levels, such that the finest grid level has a resolution of 0.5mm (or 1.0mm, depending on object size). We set $t_{\text{best}} = 5$ to limit the number of data term residuals per voxel. To reduce the increase of the number of voxels close to the surface considered for optimization, we used an adaptive thin shell size t_{shell}, linearly decreasing

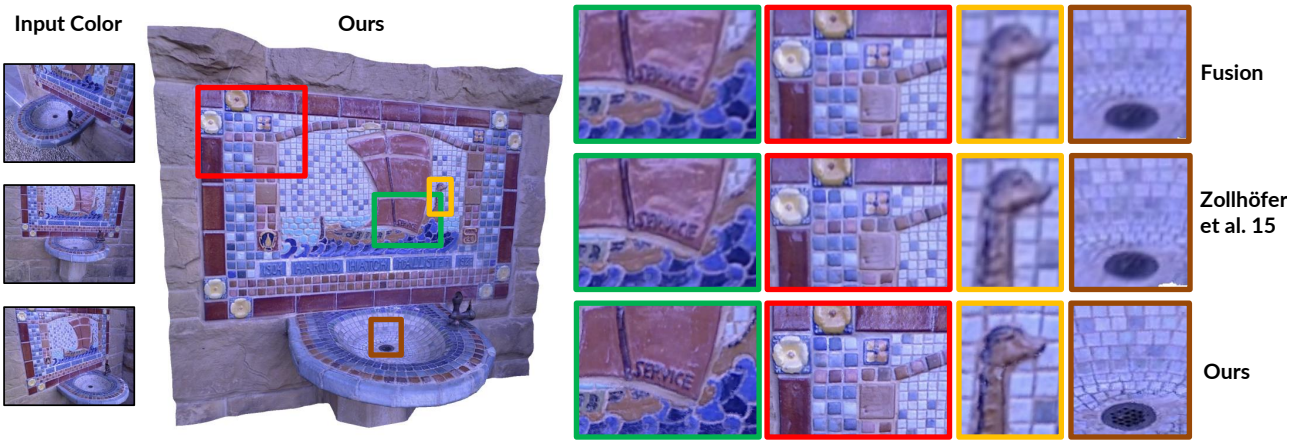


Figure 4. Appearance of the *Fountain* reconstruction. Our method shows a visually more appealing result compared to volumetric fusion and Zollhöfer et al. [30].

Dataset	# frames	# keyframes	Resolution	
			color	depth
Fountain [29]	1086	55	1280x1024	640x480
Lucy [30]	100	20	640x480	640x480
Relief [30]	40	8	1280x1024	640x480
Lion	515	26	1296x968	640x480
Tomb Statuary	523	27	1296x968	640x480
Bricks	773	39	1296x968	640x480
Hieroglyphics	919	46	1296x968	640x480
Gate	1213	61	1296x968	640x480

Table 1. Test RGB-D datasets used for the evaluation.

from 2.0 $\rightarrow$ 1.0 times the voxel size with each grid pyramid level.

Appearance Using our method, we implicitly obtain optimal voxel colors as a consequence of the joint optimization of intrinsic material properties, surface geometry and image formation model. Figure 4 shows qualitative results from the *Fountain* dataset. While volumetric blending [17, 18] produces blurry colors, camera poses are corrected in advance by Zollhöfer et al. [30] using dense bundle adjustment to yield significantly better color and geometry. However, their static color integration cannot correct for small inaccuracies, resulting in slightly blurry colors. In contrast, our method adjusts the surface and image formation model jointly to produce highly detailed texture at the same voxel grid resolution of 1mm. Within our joint optimization, we also estimate varying albedo. Figure 7 shows the estimated albedo for the *Fountain* dataset.

Surface Geometry We qualitatively compare the quality of refined surfaces using our method with the approach of Zollhöfer et al. [30] in Figure 5. The results of the *Relief* dataset visualize that our method reveals finer geometric details by directly sampling from high-resolution input color images instead of using averaged voxel colors. Moreover, we benefit from simultaneously optimizing for camera poses and camera intrinsics.

Additionally, we provide a quantitative ground truth evaluation of the geometry refinement on the synthetic *Frog*

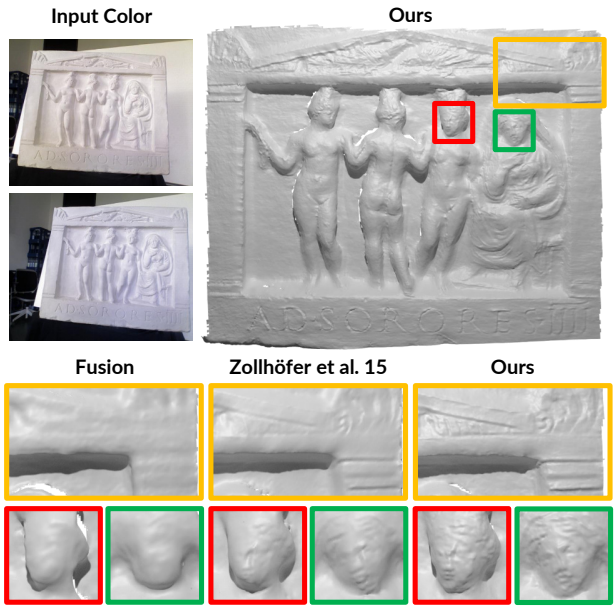


Figure 5. Comparison of the reconstructed geometry of the *Relief* dataset. Our method (right) reveals finer geometric details compared to volumetric fusion (left) and Zollhöfer et al. [30] (middle).

RGB-D dataset, which was generated by rendering a ground truth mesh with a high level of detail into synthetic color and depth images. Both depth and camera poses were perturbed with realistic noise. Figure 6 shows that, in contrast to fusion and [30], our method is able to reveal even smaller details. Quantitatively, the mean absolute deviation (MAD) between our reconstruction and the ground truth mesh is 0.222mm (with a standard deviation of 0.269mm), while the reconstruction generated using our implementation of [30] results in a higher error of 0.278mm (with a standard deviation of 0.299mm). This corresponds to an overall accuracy improvement of 20.14% of our method compared to [30]. We refer the reader to the supplementary material for a quantitative evaluation on real data and further results.

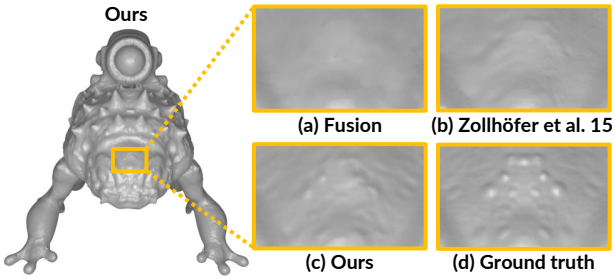


Figure 6. Refined geometry of the *Frog* dataset: while fusion (a) smooths out high-frequency details, Zollhöfer et al. [30] (b) can reconstruct some geometric details. Our method (c) recovers even smaller surface details present in the ground truth mesh (d).

Dataset	Global SH	SVSH (subvolume size)			
		0.5	0.2	0.1	0.05
Fountain	22.973	18.831	15.891	13.193	**10.263**
Lucy	22.190	19.408	16.564	14.141	**11.863**
Relief	13.818	12.432	11.121	9.454	**8.339**
Lion	30.895	25.775	20.811	16.243	**13.468**
Tomb Statuary	33.716	30.873	30.639	29.675	**26.433**
Bricks	29.327	27.110	25.318	22.850	**19.476**
Hieroglyphics	15.710	15.206	11.140	12.448	**9.998**
Gate	46.463	40.104	33.045	20.176	**12.947**

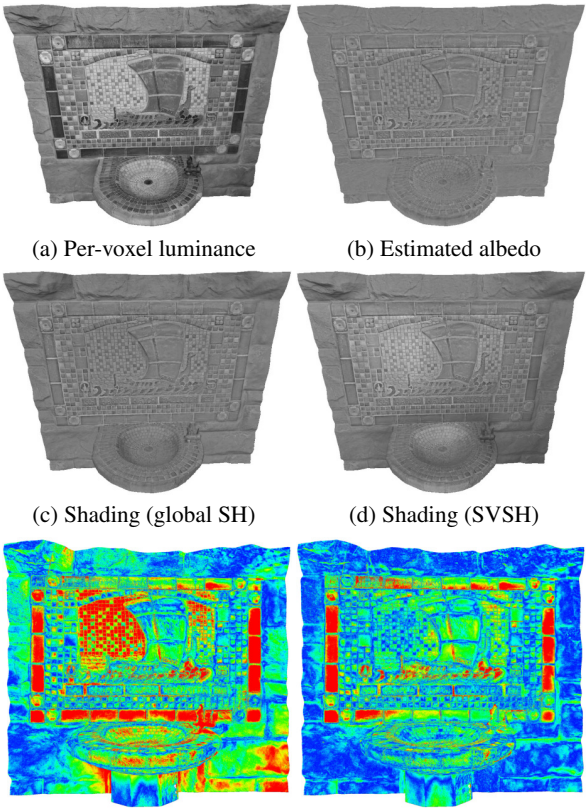
Table 2. Quantitative evaluation of spatially-varying spherical harmonics. The Mean Absolute Deviation (MAD) between averaged per-voxel intensity and estimated shading decreases with decreasing subvolume sizes.

Lighting In the following, we evaluate lighting estimation via spatially-varying spherical harmonics, both qualitatively and quantitatively. In particular, we demonstrate that a single global set of SH coefficients cannot accurately reflect real-world environments with complex lighting. To analyze the effects of the illumination, we re-light the reconstruction using the surface normals and estimated voxel albedo according to Equation 9. The computed shading $\mathbf{B}(\boldsymbol{v})$ of a voxel is in the ideal case identical to the measured voxel intensity $\mathbf{I}(\boldsymbol{v})$ computed from the voxel color.

We exploit the absolute difference $|\mathbf{B}(\boldsymbol{v}) - \mathbf{I}(\boldsymbol{v})|$ as an error metric in order to quantitatively evaluate the quality of the illumination for given geometry and albedo. In particular, we measure the mean absolute deviation (MAD) for all N voxels of the SDF volume:

$$\epsilon_{\text{shading}} = \frac{1}{N} \sum_{\boldsymbol{v} \in \mathbf{D}} |\mathbf{B}(\boldsymbol{v}) - \mathbf{I}(\boldsymbol{v})| \tag{19}$$

Table 2 gives the results of global SH coefficents and SVSH with varying subvolume sizes for multiple datasets. In summary, the more the SDF volume is partitioned into subvolumes, the better the approximation to complex lighting scenarios. The illumination in the *Fountain* dataset is clearly spatially varying, violating the assumptions of distant and spatially invariant illumination for SH lighting coefficients. Figure 7 shows that the estimated shading is better approximated with SVSH coefficients compared to only with global SH coefficients, while the underlying surface and albedo are exactly the same for both shadings.



(a) Per-voxel luminance (b) Estimated albedo

(c) Shading (global SH) (d) Shading (SVSH)

(e) Differences (global SH) (f) Differences (SVSH)

Figure 7. Quantitative evaluation of global SH vs. SVSH: the heatmaps in (e) and (f) represent the differences between the per-voxel input luminance (a) and the shadings with global SH (c) and with SVSH (d), both with underlying albedo (b).

7. Conclusion

We have presented a novel method for simultaneous optimization of scene reconstruction along with the image formation model. This way, we obtain high-quality reconstructions along with well-aligned sharp surface textures using commodity RGB-D sensors by efficiently combining information from (potentially noisy) depth and (possibly) higher resolution RGB data. In comparison to existing Shape-from-Shading techniques (e.g., [24, 30]), we tackle the core problem of fixing wrong depth measurements jointly with pose alignment and intrinsic scene parameters. Hence, we minimize re-projection errors, thus avoiding oversmoothed geometry and blurry surface textures. In addition, we introduce a significantly more flexible lighting model that is spatially-adaptive, thus allowing for a more precise estimation of the scene lighting.

Acknowledgment We would like to thank Qian-Yi Zhou and Vladlen Koltun for the *Fountain* data and Michael Zollhöfer for the *Socrates* laser scan. This work was partially funded by the ERC Consolidator grant *3D Reloaded*.

References

[1] S. Agarwal and K. Mierle. *Ceres Solver: Tutorial & Reference*. Google Inc. 6

[2] J. Barron and J. Malik. Intrinsic scene properties from a single rgb-d image. In *CVPR*, pages 17–24, 2013. 2

[3] J. Barron and J. Malik. Shape, illumination, and reflectance from shading. *PAMI*, 37(8):1670–1687, 2015. 2

[4] T. Beeler, D. Bradley, H. Zimmer, and M. Gross. Improved reconstruction of deforming surfaces by cancelling ambient occlusion. In *ECCV*, pages 30–43. Springer, 2012. 2

[5] S. Choi, Q.-Y. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. In *CVPR*, 2015. 1, 2, 4

[6] F. Crete, T. Dolmiere, P. Ladret, and M. Nicolas. The blur effect: perception and estimation with a new no-reference perceptual blur metric. In *SPIE*, pages 64920I–64920I, 2007. 4

[7] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, 1996. 1, 2, 3

[8] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics (TOG)*, 2017. 1, 2, 4

[9] B. Goldlücke, M. Aubry, K. Kolev, and D. Cremers. A super-resolution framework for high-accuracy multiview reconstruction. *IJCV*, 106(2):172–191, Jan. 2014. 2

[10] Y. Han, J. Lee, and I. So Kweon. High quality shape from a single rgb-d image under uncalibrated natural illumination. In *ICCV*, pages 1617–1624, 2013. 2

[11] B. Horn. Obtaining shape from shading information. *The Psychology of Computer Vision*, pages 115–155, 1975. 2

[12] J. Jeon, Y. Jung, H. Kim, and S. Lee. Texture map generation for 3d reconstructed scenes. *The Visual Computer*, 32(6):955–965, 2016. 2

[13] F. Klose, O. Wang, J.-C. Bazin, M. Magnor, and A. Sorkine-Hornung. Sampling based scene-space video processing. *ACM Transactions on Graphics (TOG)*, 34(4):67:1–67:11, July 2015. 2

[14] S. Lombardi and K. Nishino. Radiometric scene decomposition: Scene reflectance, illumination, and geometry from rgb-d images. In *3DV*, 2016. 3

[15] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM Transactions on Graphics (TOG)*, 21(4):163–169, 1987. 3, 6

[16] R. Maier, J. Stückler, and D. Cremers. Super-resolution keyframe fusion for 3D modeling with high-quality textures. In *3DV*, 2015. 2

[17] R. Newcombe, A. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011. 1, 2, 7

[18] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, 32(6):169, 2013. 1, 2, 3, 4, 5, 6, 7

[19] R. Or-El, G. Rosman, A. Wetzler, R. Kimmel, and A. M. Bruckstein. RGBD-Fusion: Real-time high precision depth recovery. In *CVPR*, pages 5407–5416, 2015. 2

[20] R. Ramamoorthi and P. Hanrahan. A signal-processing framework for inverse rendering. In *SIGGRAPH*, pages 117–128. ACM, 2001. 2, 4

[21] F. Steinbruecker, C. Kerl, J. Sturm, and D. Cremers. Large-scale multi-resolution surface reconstruction from rgb-d sequences. In *ICCV*, 2013. 1

[22] C. Wu, C. Stoll, L. Valgaerts, and C. Theobalt. On-set performance capture of multiple actors with a stereo camera. *ACM Transactions on Graphics (TOG)*, 32(6):161, 2013. 2

[23] C. Wu, K. Varanasi, Y. Liu, H.-P. Seidel, and C. Theobalt. Shading-based dynamic shape refinement from multi-view video under general illumination. In *ICCV*, pages 1108–1115. IEEE, 2011. 2

[24] C. Wu, M. Zollhöfer, M. Nießner, M. Stamminger, S. Izadi, and C. Theobalt. Real-time shading-based refinement for consumer depth cameras. *ACM Transactions on Graphics (TOG)*, 2014. 1, 2, 8

[25] H. Wu, Z. Wang, and K. Zhou. Simultaneous localization and appearance estimation with a consumer rgb-d camera. *IEEE Trans. Visualization and Computer Graphics*, 2016. 3

[26] L.-F. Yu, S.-K. Yeung, Y.-W. Tai, and S. Lin. Shading-based shape refinement of rgb-d images. In *CVPR*, pages 1415–1422, 2013. 2

[27] E. Zhang, M. Cohen, and B. Curless. Emptying, refurnishing, and relighting indoor spaces. *ACM Transactions on Graphics (TOG)*, 35(6):174, 2016. 3

[28] R. Zhang, P.-S. Tsai, J. Cryer, and M. Shah. Shape-from-shading: a survey. *PAMI*, 21(8):690–706, 1999. 2

[29] Q.-Y. Zhou and V. Koltun. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Transactions on Graphics (TOG)*, 33(4):155, 2014. 1, 2, 7

[30] M. Zollhöfer, A. Dai, M. Innmann, C. Wu, M. Stamminger, C. Theobalt, and M. Nießner. Shading-based refinement on volumetric signed distance functions. *ACM Transactions on Graphics (TOG)*, 34(4):96, 2015. 1, 2, 5, 7, 8

C.3 Efficient Online Surface Correction for Real-time Large-Scale 3D Reconstruction

Copyright

Robert Maier, Raphael Schaller, and Daniel Cremers
Efficient Online Surface Correction for Real-time Large-Scale 3D Reconstruction
British Machine Vision Conference (BMVC) 2017
DOI: 10.5244/C.31.158

Summary

State-of-the-art methods for large-scale 3D reconstruction from RGB-D sensors usually reduce drift in camera tracking by globally optimizing the estimated camera poses in real-time without simultaneously updating the reconstructed surface on pose changes. We propose an efficient on-the-fly surface correction method for globally consistent dense 3D reconstruction of large-scale scenes. Our approach uses a dense Visual RGB-D SLAM system that estimates the camera motion in real-time on a CPU and refines it in a global pose graph optimization. Consecutive RGB-D frames are locally fused into keyframes, which are incorporated into a sparse voxel hashed Signed Distance Field (SDF) on the GPU. On pose graph updates, the SDF volume is corrected on-the-fly using a novel keyframe re-integration strategy with reduced GPU-host streaming. We demonstrate in an extensive quantitative evaluation that our method is up to 93% more runtime efficient compared to the state-of-the-art and requires significantly less memory, with only negligible loss of surface quality. Overall, our system requires only a single GPU and allows for real-time surface correction of large environments.

Individual contributions

Significant contribution in realizing the scientific project.

Problem definition	*significantly contributed*
Literature survey	*significantly contributed*
Implementation	*helped*
Experimental evaluation	*contributed*
Preparation of the manuscript	*significantly contributed*

Efficient Online Surface Correction for Real-time Large-Scale 3D Reconstruction

Robert Maier*
maierr@in.tum.de
Raphael Schaller*
schaller@in.tum.de
Daniel Cremers
cremers@in.tum.de

Computer Vision Group
Technical University of Munich
Munich, Germany

* equal contribution

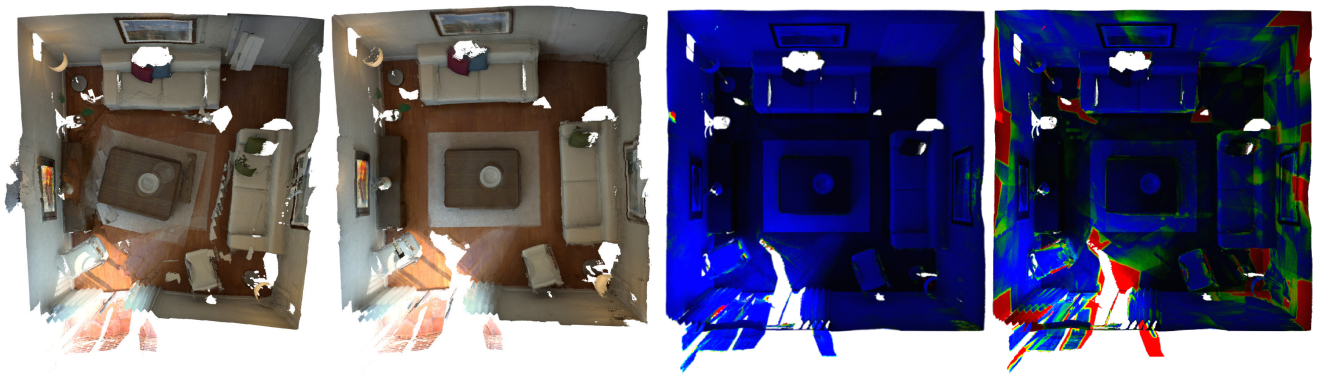
Abstract

State-of-the-art methods for large-scale 3D reconstruction from RGB-D sensors usually reduce drift in camera tracking by globally optimizing the estimated camera poses in real-time without simultaneously updating the reconstructed surface on pose changes. We propose an efficient on-the-fly surface correction method for globally consistent dense 3D reconstruction of large-scale scenes. Our approach uses a dense Visual RGB-D SLAM system that estimates the camera motion in real-time on a CPU and refines it in a global pose graph optimization. Consecutive RGB-D frames are locally fused into keyframes, which are incorporated into a sparse voxel hashed Signed Distance Field (SDF) on the GPU. On pose graph updates, the SDF volume is corrected on-the-fly using a novel keyframe re-integration strategy with reduced GPU-host streaming. We demonstrate in an extensive quantitative evaluation that our method is up to 93% more runtime efficient compared to the state-of-the-art and requires significantly less memory, with only negligible loss of surface quality. Overall, our system requires only a single GPU and allows for real-time surface correction of large environments.

1 Introduction

In recent years, there has been a boost of research in the field of dense 3D reconstruction due to the wide availability of low-cost depth sensors such as the Microsoft Kinect. Most of the approaches fuse depth maps obtained from such sensors in real-time into a volumetric surface representation [3] to compensate for sensor noise and perform frame-to-model camera tracking against the fused volume. While researchers have shown the suitability of these methods for accurate geometric reconstruction of objects or scenes of limited size [16], global drift in camera tracking is not compensated, limiting the reconstruction of large-scale environments [8, 17, 19].

However, only few methods tackle the problem of globally optimizing the camera poses in real-time and simultaneously correcting the reconstructed surface on-the-fly. BundleFusion by Dai et al.[4] represents the state-of-the-art and estimates highly accurate camera poses on a high-end GPU. They require a second graphics card for integrating input RGB-D frames into a sparse Signed Distance Field (SDF) volume, making the entire framework computationally demanding. On pose graph updates, BundleFusion corrects the reconstructed



a) Without and with on-the-fly surface correction b) Surface completeness (5 vs. 60 frames per keyframe)

Figure 1: Our method efficiently corrects the surface during the 3D scanning process on-the-fly (a) using an efficient keyframe re-integration strategy. Fusing fewer frames into each keyframe allows to maintain the completeness of the reconstructed 3D model (b).

surface on-the-fly by frame re-integration. However, all previous frames need to be held in memory to allow for a fast re-integration on pose updates; this limits its suitability for scanning large-scale environments with long sequences.

To enable state-of-the-art large-scale 3D reconstruction from RGB-D sensors, our SLAM framework is based on DVO-SLAM by Kerl et al. [10] for estimating a globally consistent camera motion. The system is computationally significantly less expensive than BundleFusion and works in real-time on a single CPU, with only slightly less accurate estimated camera poses. To obtain globally consistent and up-to-date reconstructions of large environments, we couple it with our novel 3D surface correction method. Figure 1 shows the result of our online surface re-integration method at the end of a 3D scanning session and indicates the effect of keyframe fusion on the completeness of the reconstruction.

In summary, the main contributions of our work are:

- We integrate our 3D surface correction framework with a dense Visual SLAM system, such that our entire 3D reconstruction system runs in real-time with a single GPU only.
- We fuse consecutive RGB-D input frames in keyframes of high depth and color quality using different keyframe strategies.
- Our surface correction framework is highly efficient by only re-integrating fused keyframes into a sparse SDF volume on pose graph updates.
- Our strategy for selecting keyframes to be updated substantially reduces streaming between host and GPU.
- An extensive quantitative evaluation shows that our method is overall 93% more efficient compared to the state-of-the-art while maintaining surface quality.

2 Related Work

The field of dense 3D reconstruction from RGB-D data has been investigated extensively in recent years. KinectFusion by Newcombe et al. [16] enabled dense 3D reconstruction in real-time through extensive use of GPU programming. Like most of the following approaches, it stores the 3D model in an SDF volume [3], which regularizes the noisy depth maps from RGB-D sensors, and performs ICP-based camera tracking against the raycasted 3D model. Voxel Hashing [17] better exploits scarce GPU memory and allocates only occupied voxel blocks of the SDF. A hash map flexibly maps 3D voxel block coordinates onto memory locations. Kähler et al. [8] designed an optimized version of Voxel Hashing for mobile devices. However, the frame-to-model camera tracking of the frameworks above is only of

limited use for reconstructing larger scenes. To reduce drift explicitly, recent approaches [1, 13, 18, 22] rely on loop closure detection in combination with global pose optimization.

In order to efficiently estimate camera poses in real-time, DVO-SLAM by Kerl et al. [10] minimizes a photometric and geometric error to accurately align RGB-D frames. For global consistency, it continuously performs a pose graph optimization to reduce global drift. While there is no dense volumetric model representation, they exploit keyframes to reduce the influence of noise. The system provides an excellent trade-off between runtime and accuracy, making it highly suitable for our 3D reconstruction framework. Utilizing keyframes as intermediate representation for reducing noise has also been exploited for improving camera tracking [15] and reconstruction appearance [14]. Following this idea, we also employ keyframes in our work as memory efficient intermediate 2.5D representations of 3D surfaces.

There are only few works on real-time large-scale RGB-D based 3D reconstruction that incorporate online surface correction. Fioraio et al. [5] reconstruct overlapping subvolumes, register their poses globally and update the subvolumes using volume blending. However, the absence of loop closure detection avoids to cope with larger drift. Kähler et al. [9] perform real-time tracking against multiple submaps independently and globally optimize the estimated trajectories. Submaps are fused on-the-fly during raycasting.

Whelan et al. use a deformation graph for online update of a surfel-based model [21] and of an SDF model [20] with an as-rigid-as-possible surface deformation. In ElasticFusion [21] input frames are fused into surfels and then discarded. However, wrong camera poses (e.g. due to drift) result in inconsistent surfel observations and hence increase their uncertainties; surfels with high uncertainty are ultimately filtered out. When a loop closure is detected, only the existing surface can be corrected along with the deformation graph, while surface information lost through inconsistent fusion cannot be recovered. In contrast, our method keeps all keyframes fused from input data and allows to re-integrate them at any pose graph update without a loss of surface information. Additionally, despite correcting the model online, the frame-to-model camera tracking may fail to compensate for drift due to delayed surface updates and undetected (or too late detected) loop closures.

BundleFusion et al. [4] represents the state-of-the-art both w.r.t. SLAM system accuracy as well as on-the-fly surface re-integration. The system first optimizes consecutive frame poses locally within chunks, which are then aligned globally in a hierarchical global optimization. New RGB-D input frames are matched brute-force against all previous chunk keyframes and subsequently aligned using a sparse-then-dense alignment. The global alignment regularly changes camera poses; to correct the reconstructed sparse SDF volume on-the-fly, the system first de-integrates frames with their former poses and then integrates them with their updated poses using a simple re-integration strategy. The 3D model is gradually adapted to the updated poses while still enabling real-time reconstruction. In contrast to BundleFusion, our method needs only a single GPU for surface modeling instead of two high-end graphics cards. Our online surface re-integration combines keyframe fusion with a more intelligent keyframe selection strategy, resulting in a significantly more efficient re-integration. Moreover, the use of keyframes requires substantially less memory and enables on-the-fly surface correction for large environments.

3 3D Reconstruction System

Our framework consists of a real-time RGB-D SLAM framework for globally consistent camera pose estimation and a sparse SDF volume for storing the reconstructed 3D model.

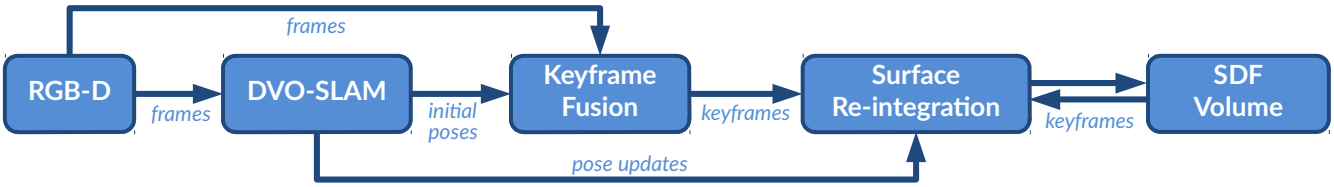


Figure 2: Overview of our online surface correction method. RGB-D frames are fused into keyframes, which are (re-)integrated into the SDF on-the-fly on DVO-SLAM pose updates.

While dense SDF-based 3D reconstruction methods usually integrate new RGB-D input frames directly into the volume, we first fuse them into keyframes as intermediate data representation. We integrate and re-integrate them online into the SDF on pose updates to efficiently correct the surface. This way, we can reduce the number of individual frames that we integrate into the SDF volume, which helps especially when we need to correct the 3D model due to a pose update. Figure 2 shows an overview of our approach.

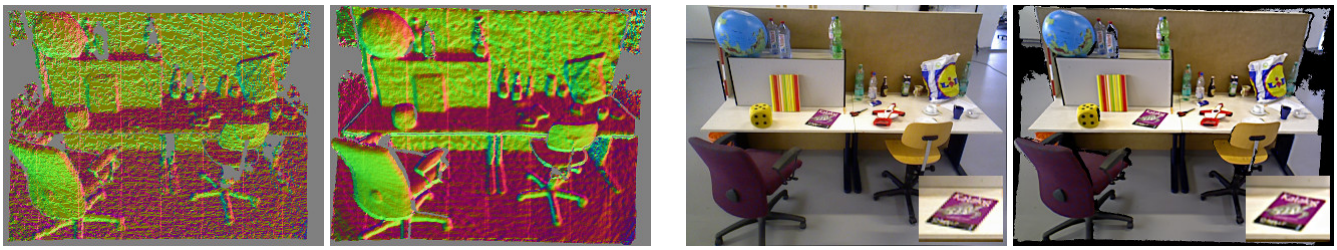
Preliminaries We acquire RGB-D data from commodity depth sensors with 30 fps at a resolution of 640×480 pixels. The N captured RGB-D frames consist of registered color images $\mathcal{C}_i$, depth maps $\mathcal{Z}_i$ and camera poses $\mathcal{T}_i = (R_i, t_i) \in \mathrm{SE}(3)$ (with $R_i \in \mathrm{SO}(3)$, $t_i \in \mathbb{R}^3$ and $i \in 1 \ldots N$). A 3D point $p = (X, Y, Z)^\top$ is transformed using a pose $\mathcal{T}_i$ through $g(\mathcal{T}_i, p) = R_i p + t_i$. We use the pinhole camera model, which projects 3D points p to 2D pixels $x = (u, v)^\top = \pi(p)$ using the projection $\pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$. The inverse projection $\pi^{-1} : \mathbb{R}^2 \times \mathbb{R} \mapsto \mathbb{R}^3$ maps a 2D pixel location x back to the respective 3D point $p = \pi^{-1}(x, \mathcal{Z}(x))$ using its depth.

Dense Visual RGB-D SLAM To estimate globally consistent camera poses $\mathcal{T}_i$, we utilize the DVO-SLAM system by Kerl et al. [10]. It runs in real-time on a CPU and employs a robust dense visual odometry approach that minimizes the photometric and geometric error of all pixels to estimate the rigid body motion between two RGB-D frames. To reduce drift in camera pose estimation, input frames are aligned against the preceding keyframe. Keyframes are selected using the differential entropy of the motion estimate and a pose distance threshold. In the following, we refer to the keyframes selected by DVO-SLAM as *DVO keyframes*. DVO-SLAM detects loop closures by aligning keyframes against candidates of previous keyframes within a sphere of predefined radius and validates them using their entropy ratio. Estimated frame-to-(key)frame camera motions and successful loop closures are integrated as constraints into a graph based map representation. This keyframe pose graph is steadily optimized in the background during runtime, yielding a globally consistent camera trajectory with continuously updated keyframe poses $\mathcal{T}_i$. Please note that our surface correction method works in principle with any SLAM system that incorporates loop closures.

Keyframe Fusion Our keyframe fusion builds up on [14] and consists of separate steps for depth and color fusion (cf. Figure 3). While new depth maps are immediately fused into the keyframe depth, color fusion relies on the more complete fused keyframe depth.

For depth fusion, we first compute for each pixel x of $\mathcal{Z}_i$ its respective view- and distance-dependent weight $w_z(x) = \cos(\theta_i(x)) \cdot \mathcal{Z}_i(x)^{-2}$, where $\theta_i(x)$ is the angle between the depth normal at x and the camera axis. Furthermore, we discard error-prone depth values close to depth discontinuities. We then warp each pixel with the frame pose $\mathcal{T}_i$ into the keyframe with pose $\mathcal{T}^*$ and obtain $p^* = (X^*, Y^*, Z^*)^\top = g(\mathcal{T}^{*-1}, g(\mathcal{T}_i, \pi^{-1}(x, \mathcal{Z}_i(x))))$. The keyframe depth $\mathcal{Z}^*$ and the depth fusion weights $\mathcal{W}^*$ at the projected 2D image point $x^* = \pi(p^*)$ are then updated as follows:

$$\mathcal{Z}^*(x^*) = \frac{\mathcal{W}^*(x^*)\mathcal{Z}^*(x^*) + w_z(x)Z^*}{\mathcal{W}^*(x^*) + w_z(x)}, \qquad \mathcal{W}^*(x^*) = \mathcal{W}^*(x^*) + w_z(x). \tag{1}$$



a) Input depth b) Fused depth c) Input color d) Fused color

Figure 3: Keyframe fusion: several consecutive input depth maps (a) are fused into the keyframe depth (b). Our color fusion creates sharp color keyframes (d) from input color (c).

For color fusion, we first deblur the input color images using *Unsharp Masking* and compute a per-frame blurriness measure w_b [2] from $\mathcal{C}_i$ to alleviate frames with strong motion blur. In contrast to depth fusion, the fused keyframes are warped back into each input frame i and the observed color values $c_i(x^*) = \mathcal{C}_i(\pi(g(\mathcal{T}_i^{-1}, g(\mathcal{T}^*, \pi^{-1}(x^*, \mathcal{Z}^*(x^*))))))$ are sampled using bilinear interpolation. For a pixel x^* in the keyframe, we collect all valid color observations in the input views and their color weights $w_{c_i}(x^*) = w_b \cdot w_z$. We compute the final keyframe color $\mathcal{C}^*(x^*)$ as the weighted median of the collected observations.

SDF volume At the core of our method, we store a memory efficient sparse SDF volume based on Voxel Hashing [17] as volumetric 3D model representation for large-scale 3D reconstructions. The implemented data structure is tailored to GPUs and only occupied space is allocated in voxel blocks, which are efficiently addressed using spatial hashing. For each voxel v, we store its signed distance $\mathbf{D}(v)$, its color $\mathbf{C}(v)$ and its integration weight $\mathbf{W}(v)$. We extract the iso-surface from the SDF using Marching Cubes [12]. To overcome the limitations of scarce GPU memory for large-scale environments, voxel blocks are streamed from GPU to host (and vice versa) before integration of a new frame. In particular, only voxel blocks within a sphere of constant radius around the current camera position are kept in GPU memory, while all other voxel blocks are streamed to the host. When an RGB-D frame is integrated into the SDF volume, voxel blocks are first allocated and the voxels are then updated using a running weighted average.

4 Efficient Online Surface Re-Integration

In the following, we introduce our online surface correction method that combines keyframe fusion with our sparse SDF volume implementation. Firstly, we incorporate on-the-fly keyframe re-integration into the 3D reconstruction pipeline; secondly, we show different strategies for starting new keyframes; thirdly, we propose an efficient surface correction procedure that is based on a re-integration strategy that reduces GPU-host transfer.

4.1 System Pipeline

While DVO-SLAM selects *DVO keyframes* for camera tracking based on an entropy criteria, we introduce *KF keyframes* (Keyframe Fusion keyframes) as intermediate representation for surface (re-)integration. When a new frame arrives, DVO-SLAM provides an initial pose estimate which is used to fuse the input frame into the current KF keyframe. Depending on the chosen keyframe selection strategy, a new keyframe will be started if some criteria are met and the previous KF keyframe is integrated into the SDF volume. The KF keyframe is also stored in memory for later re-integration on pose updates. Since DVO-SLAM issues only pose updates for DVO keyframes, we convert by expressing KF keyframe poses relative

Frame	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Distance	1	3	4	3	5	4	1	7	2	1	1	8	6	2	0
Distance (sum)			16	19	17	**20**	19	15	12	19	18	18	17		

Figure 4: Selection of frames for re-integration: BundleFusion [4] chooses the frames with highest distances between integrated pose and new pose. However, selecting frames $12,8,13,5,3$ results here in disadvantageous shifts of the streaming sphere. Our method selects the group of most-moved m consecutive frames, which results in frames 4 to 8 ($j^* = 4$).

to DVO keyframe poses. The KF keyframes are then de-integrated from the SDF volume with their former camera poses and re-integrated on-the-fly with their updated poses.

4.2 Keyframe Strategies

In the following, we investigate keyframe selection strategies w.r.t. obtaining optimal surface quality. With only few input frames fused into KF keyframes, many KF keyframes need to be (re-)integrated into the SDF volume on pose updates. On the other hand, fusing many input frames into KF keyframes leads to a degradation in 3D reconstruction quality, since the 2.5D keyframes cannot fully represent the incorporated 3D information. We present keyframe strategies to find the optimal trade-off between re-integration performance and reconstruction quality.

The KF_CONST strategy is a simple but effective strategy and fuses a constant number κ of frames into each KF keyframe. KF_DVO uses the frames selected as DVO keyframes also as KF keyframes. The distance based strategy KF_DIST issues a new KF keyframe whenever the rotational distance Δ_r or translational distance Δ_t of the relative pose $\mathcal{T}_{ij}$ between the current frame and the current KF keyframe exceeds a certain threshold, similar to [11]. The overlap strategy KF_OVRLP is derived from [7] and generates a new KF keyframe when the ratio of the pixels visible in both current frame and keyframe drops below a threshold.

4.3 On-the-fly Surface Correction

Our surface correction method follows the frame re-integration approach of [4]. However, we substantially improve it at critical points w.r.t. runtime efficiency by implementing a more intelligent strategy for selecting the KF keyframes to be re-integrated. Since we only need to correct KF keyframes instead of all frames, our surface correction is highly efficient w.r.t. runtime and memory consumption.

Frame de-integration For de-integrating an RGB-D frame i from the SDF volume, we simply reverse the integration procedure. We therefore retrieve the KF keyframe from the memory and compute the projective distance d_i (along the z axis) of v in depth map $\mathcal{Z}_i$ (with sampling weight w_i) and its sampled color c_i in the input color image $\mathcal{C}_i$. The de-integration steps for updating signed distance, color and weight of a voxel are denoted as follows:

$$\mathbf{D}'(v) = \frac{\mathbf{D}(v)\mathbf{W}(v) - d_i w_i}{\mathbf{W}(v) - w_i} \qquad \mathbf{C}'(v) = \frac{\mathbf{C}(v)\mathbf{W}(v) - c_i w_i}{\mathbf{W}(v) - w_i} \qquad \mathbf{W}'(v) = \mathbf{W}(v) - w_i \tag{2}$$

Re-integration strategy While the poses of all keyframes are updated when DVO-SLAM issues a pose update, it is computationally too expensive to correct them all immediately. Instead, we re-integrate only m changed frames whenever we receive a pose update. Poses that were not corrected on-the-fly are re-integrated in a final pass after the reconstruction. We denote the SDF integration pose of a frame by $\mathcal{T}_i$ and the updated pose by $\mathcal{T}_i'$.

To select the m frames for re-integration, BundleFusion orders all frames by descending distance between $\mathcal{T}_i$ and $\mathcal{T}_i'$ $\|st_i - st_i'\|$ and selects the m most-moved frames. The vectors t_i and t_i' contain the Euler rotation angles and the translation of the poses $\mathcal{T}_i$ and $\mathcal{T}_i'$, with a constant scale vector $s = (2,2,2,1,1,1)^\top$. However, since the corrected frames (and the respective SDF voxel blocks) may be spatially distant, suboptimal expensive GPU-host-streaming of voxel blocks may be required. To limit the streaming overhead, it is beneficial to correct close frames within the same re-integration procedure. We therefore keep the original temporal ordering of frames and select the group of most-moved *m consecutive* frames:

$$j^* = \underset{j \in [1, K-m+1]}{\operatorname{argmax}} \sum_{i=j}^{j+m-1} \left\| st_i - st_i' \right\|, \tag{3}$$

where K is the total number of frames integrated so far. The resulting j^* represents the first frame of our m consecutive frames thats need to be re-integrated. Figure 4 exemplifies the advantages of this procedure. Additionally, we adjust the streaming procedure of [17] to the re-integration process: We first stream in all voxel blocks inside the sphere around pose $\mathcal{T}_{j^*}$ to safely access them. Then, we successively de-integrate frames $[j^*, j^*+m-1]$ with regular streaming. After de-integration, we stream in the sphere around the updated pose $\mathcal{T}_{j^*}'$ und successively re-integrate frames $[j^*, j^*+m-1]$ using their updated poses with regular streaming. We finally stream the sphere back to the next integration pose.

5 Evaluation and Experimental Results

To demonstrate the effectiveness of our surface reconstruction algorithm, we provide a thorough quantitative evaluation w.r.t. runtime efficiency and surface accuracy. In particular, we analyze the effects of combining keyframe fusion with our surface re-integration method.

Datasets We use publicly available RGB-D datasets of large-scale scenes with loop closures that provide registered depth and color images as well as the respective camera poses. *AUG_ICL/Liv1* (noisy) [1] is a synthetic RGB-D sequence that is rendered from a modeled scene of a living room with realistic sensor noise. In addition to ground truth poses it also provides the ground truth 3D scene model that allows for a quantitative comparison of surface quality of reconstructed 3D models. *BundleFusion/apt0* [4] features a long camera trajectory of 8560 frames with poses estimated from BundleFusion.

Surface evaluation methods and metrics The evaluation procedure for comparing our reconstructed 3D models with synthetic ground truth is adapted from [6] and first extracts a 3D mesh $\mathcal{M}$ from the reconstructed SDF volume. We use CLOUDCOMPARE [1] to uniformly sample a reference point cloud $\mathcal{R}$ with 50 million points from the ground truth mesh of *AUG_ICL/Liv1*. We measure the distance of each vertex of $\mathcal{M}$ to its closest vertex in $\mathcal{R}$ with SURFREG [2] and compute the mean absolute deviation MAD. This technique assesses the *correctness* CORR of the model, i.e. the accuracy of the successfully reconstructed surfaces. However, we also want to measure the *completeness* COMPL of reconstructions to determine the information loss from keyframe fusion. For measuring COMPL, we inversely compare every vertex of $\mathcal{R}$ to the nearest neighbor in $\mathcal{M}$. For a fair comparison and to only compare surfaces visible in the synthetic frames, we re-generate the reference $\mathcal{R}$ by fusing all input frames into the SDF with ground truth poses. We rely on the poses from the datasets for

[1] http://www.danielgm.net/cc/
[2] https://github.com/mp3guy/SurfReg

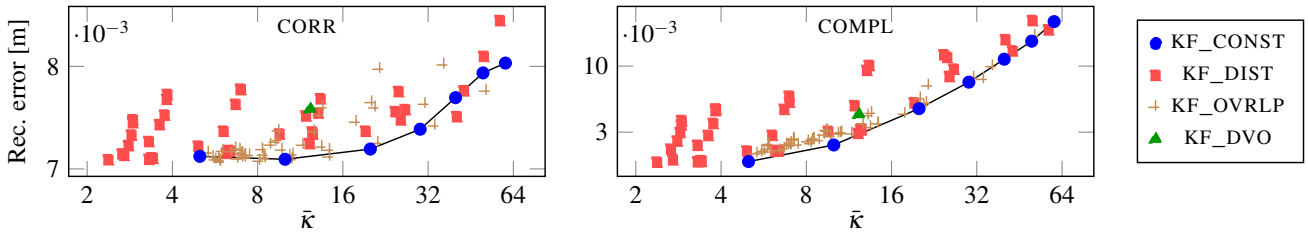


Figure 5: Quantitative evaluation of reconstruction correctness (left) and completeness (right) w.r.t. different keyframe strategies on *AUG_ICL/Liv1*. The x-axis shows the average keyframe size $\bar{\kappa}$ produced in each run, the y-axis shows the MAD error (axes are logarithmic). The KF_CONST strategy achieves the best reconstruction results for both CORR and COMPL.

assessing the surface quality to eliminate a substantial source of error. We used a workstation with Intel Core i7-3770 CPU, 32GB RAM and an NVIDIA GeForce GTX 1070 GPU.

5.1 Keyframe Fusion

We quantitatively investigate the effect of keyframe fusion on the reconstruction quality, i.e. surface completeness and correctness, of the noisy *AUG_ICL/Liv1* dataset.

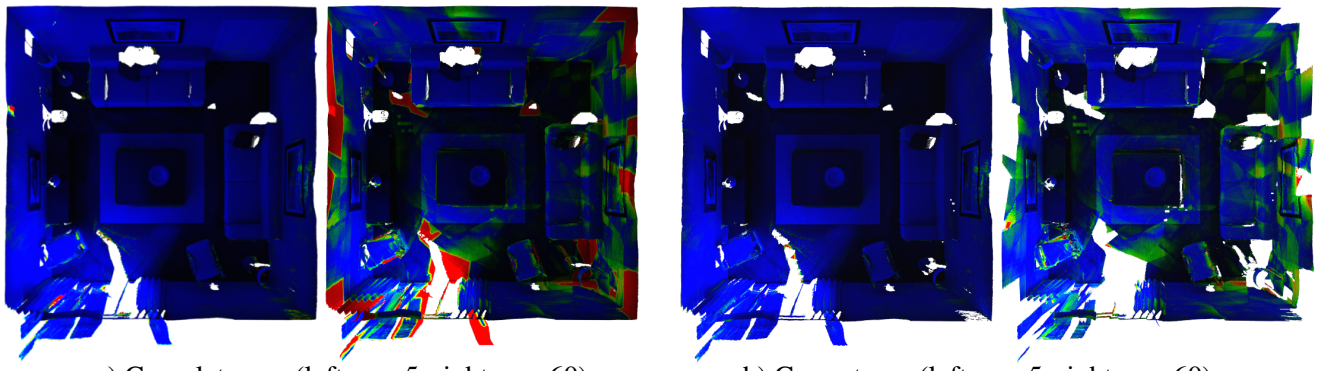
Keyframe strategies Figure 5 shows the results of different keyframe selection strategies and their average keyframe sizes on the reconstructed surface quality. Each mark represents a separate evaluation run of a given strategy with a different set of specified parameters. For KF_CONST, we vary the number of consecutive frames κ that are fused into each keyframe. In KF_DIST we adjusted the pose distance threshold Δ_t and Δ_r, while we varied the overlap ratio parameters in KF_OVRLP. In KF_DVO we use the same keyframes as DVO-SLAM. In summary, more relaxed parameters result in a higher average number of fused frames per keyframe $\bar{\kappa}$ for all strategies; different parameter combinations for the same strategy may result in a similar $\bar{\kappa}$. With an increasing $\bar{\kappa}$, the completeness of reconstructions decreases rapidly, since 3D surface information gets lost in 2.5D keyframe fusion. The effect on surface correctness is less significant, since the deviation for the remaining surfaces is still reasonably close to the ground truth 3D model. Compared to KF_CONST, the strategies KF_DIST, KF_OVRLP and KF_DVO result mostly in worse quantitative results, a hardly predictable number of keyframes and barely tunable interdependent parameters. We found KF_CONST to give good quantitative results, while it is also highly predictable w.r.t. fusion and re-integration runtime as well as memory consumption ($\sim 1/\bar{\kappa}$) due to its priorly known number of frames per keyframe. We refer the reader to the supplementary material for more details.

Completeness Figure 6 shows color coded distance renderings for KF_CONST keyframe fusion with $\kappa = 5$ and $\kappa = 60$ on *AUG_ICL/Liv1*. The colors represent errors from 0mm (blue) to 50mm (red). Again, the completeness COMPL of reconstructions decreases with more fused frames per keyframe because of the loss of surface information with 2.5D keyframes. The reconstructed surfaces are still accurate w.r.t. ground truth (CORR).

5.2 Surface Re-integration

We finally assess our surface correction w.r.t. real-time performance and show results of on-the-fly surface re-integration on real-world data.

Runtime While BundleFusion requires two high-end GPUs to operate in real-time, our system requires only a single GPU. Figure 7 gives the average amortized runtimes of our system, specifically for (re-)integration of frames w.r.t. κ (red), for DVO-SLAM (green) and



a) Completeness (left $\kappa = 5$, right $\kappa = 60$) b) Correctness (left $\kappa = 5$, right $\kappa = 60$)

Figure 6: Completeness and correctness after integration of *AUG_ICL/Liv1* with KF_CONST keyframe strategy (with keyframe sizes 5 and 60).

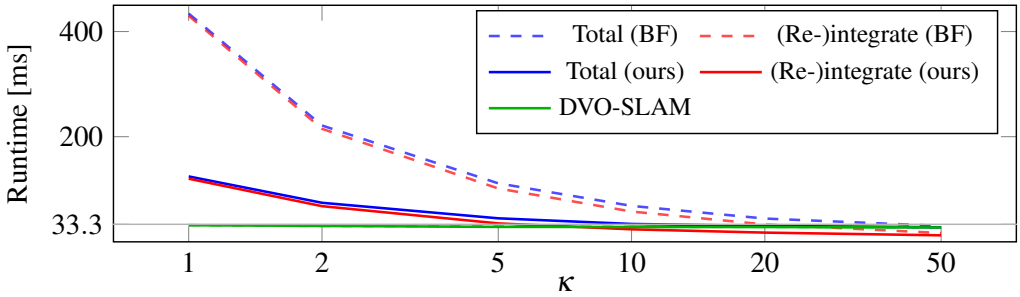


Figure 7: Average runtime per frame for reconstruction of *AUG_ICL/Liv1* with KF_CONST w.r.t. κ. Our re-integration strategy (solid) is substantially faster than BundleFusion's (dashed). m was set to $100/\kappa$, yielding a constant effective re-integration rate.

the total runtime of both combined (blue). Generally, the higher κ is, the fewer keyframes are generated and thus need to be updated. We accomplish real-time performance with $\kappa = 20, m = 5$. Here, the re-integration strategy of updating the m most-moved *consecutive* keyframes (solid lines) saves already 47% of (re-)integration runtime compared to BundleFusion's simple strategy (dashed). This is further accelerated through the use of keyframes only: Overall, the reconstruction with $\kappa = 20, m = 5$ and our re-integration strategy takes 93% less time than BundleFusion's re-integration strategy without keyframe fusion ($\kappa = 1, m = 100$). We found $m \in [10, 20]$ to be a good trade-off between reconstruction quality and model correction speed for most data sets.

On-the-fly surface re-integration As DVO-SLAM steadily optimizes a pose graph and issues pose updates, our surface correction method gradually improves the reconstructed 3D model on-the-fly by re-integrating the most-moved consecutive m keyframes into the SDF. While updating all changed keyframes at once is too expensive, we can control the speed of incorporating pose updates into the 3D model by adjusting m. Also, with decreasing $\bar{\kappa}$ more keyframes are generated and need to be updated. Figure 8 shows an example of how a 3D reconstruction is corrected on-the-fly during the reconstruction to be as globally consistent as possible (with $m = 5$, $\kappa = 20$ and KF_CONST strategy).

An isolated comparison of the surface correction of ElasticFusion [21] with our method is not applicable since the respective SLAM systems may result in different camera trajectories. Nevertheless, Figure 9 shows a qualitative comparison of the generated models, which is in accordance with the findings in [4]. While ElasticFusion might benefit from camera tracking against the corrected model, the point cloud reconstructed with ElasticFusion with default parameters exhibits double walls and artifacts due to potentially undetected loop closures and surface warping artifacts. These effects are mitigated in the continuous surface mesh reconstructed from our method, which successfully corrects the model on-the-fly.

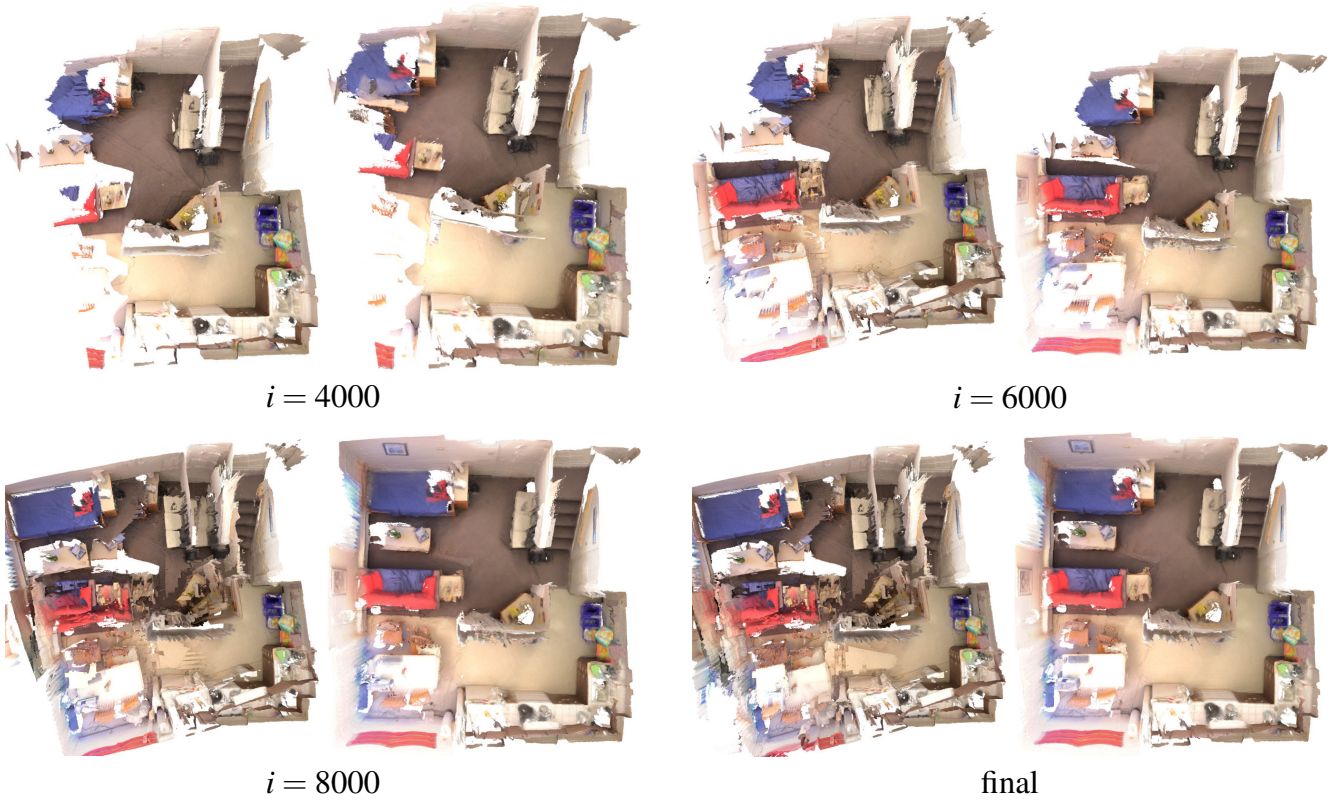


Figure 8: Reconstruction of *BundleFusion/apt0*. Every 2000 frames, a model was generated without (left) and with (right) on-the-fly surface correction (KF_CONST keyframe strategy with $m = 5$, $\kappa = 20$).

Figure 9: Qualitative comparison of ElasticFusion [21] (left) with our method (right) on *BundleFusion/apt0*. The point cloud reconstructed with ElasticFusion exhibits artifacts due to potentially undetected loop closures and surface warping artifacts, whereas our method successfully corrects the model.

6 Conclusion

We presented an efficient online surface re-integration method for globally consistent 3D reconstruction of large-scale scenes from RGB-D sensors in real-time on a single GPU only. Our SLAM system based on DVO-SLAM estimates the camera motion in real-time on a CPU and employs pose graph optimization for obtaining globally optimized camera poses. Multiple RGB-D frames are first fused into keyframes, which are then integrated into a sparse voxel hashed SDF model representation. Continuous keyframe pose updates are gradually incorporated into the SDF volume by on-the-fly re-integration of changed keyframes. Our improved re-integration strategy with correction of keyframes and significantly reduced host-GPU-streaming saves about 93% of runtime compared to the state-of-the-art. By re-integrating keyframes (instead of all frames), we substantially reduce the number of frames to be re-integrated with only a slight degradatation of reconstruction quality.

Acknowledgement This work was funded by the ERC Consolidator grant *3D Reloaded*.

References

[1] S. Choi, Q.Y. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. In *CVPR*, 2015.

[2] F. Crete, T. Dolmiere, P. Ladret, and M. Nicolas. The blur effect: perception and estimation with a new no-reference perceptual blur metric. In *SPIE*, pages 64920I–64920I, 2007.

[3] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *SIGGRAPH*, 1996.

[4] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface re-integration. *ACM Transactions on Graphics (TOG)*, 2017.

[5] N. Fioraio, J. Taylor, A. Fitzgibbon, L. Di Stefano, and S. Izadi. Large-scale and drift-free surface reconstruction using online subvolume registration. In *CVPR*, 2015.

[6] A. Handa, T. Whelan, J.B. McDonald, and A.J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *ICRA*, 2014.

[7] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *IJRR*, 31(5):647–663, 2012.

[8] O. Kähler, V.A. Prisacariu, C.Y. Ren, X. Sun, P. Torr, and D. Murray. Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Transactions on Visualization and Computer Graphics*, 21(11):1241–1250, 2015.

[9] O. Kähler, V.A. Prisacariu, and D. Murray. Real-time large-scale dense 3D reconstruction with loop closure. In *ECCV*, 2016.

[10] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for RGB-D cameras. In *IROS*, 2013.

[11] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *ISMAR*, 2007.

[12] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*, 1987.

[13] R. Maier, J. Sturm, and D. Cremers. Submap-based bundle adjustment for 3D reconstruction from RGB-D data. In *GCPR*, 2014.

[14] R. Maier, J. Stückler, and D. Cremers. Super-resolution keyframe fusion for 3D modeling with high-quality textures. In *3DV*, 2015.

[15] M. Meilland and A.I. Comport. Super-resolution 3D tracking and mapping. In *ICRA*, 2013.

[16] R.A. Newcombe, A.J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[17] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, 32(6):169, 2013.

[18] F. Steinbrücker, J. Sturm, and D. Cremers. Volumetric 3D mapping in real-time on a CPU. In *ICRA*, 2014.

[19] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially extended kinectfusion. 2012.

[20] T. Whelan, M. Kaess, R. Finman, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Real-time large scale dense rgb-d slam with volumetric fusion. *IJRR*, 2014.

[21] T. Whelan, S. Leutenegger, R.F. Salas-Moreno, B. Glocker, and A.J. Davison. Elasticfusion: Dense slam without a pose graph. In *RSS*, 2015.

[22] Q.Y. Zhou, S. Miller, and V. Koltun. Elastic fragments for dense scene reconstruction. In *ICCV*, 2013.

List of Figures

List of Tables

List of Abbreviations

AR	Augmented Reality
BA	Bundle Adjustment
BRDF	Bidirectional Reflectance Distribution Function
CAD	Computer Aided Design
CPU	Central Processing Unit
DDA	Digital Differential Analyzer
DoF	Degrees-of-Freedom
fps	frames per second
GD	Gradient Descent
GN	Gauss-Newton
GPU	Graphics Processing Unit
GPGPU	General Purpose GPU
HMD	Head-mounted Display
ICP	Iterative Closest Point
IR	Infrared
IRLS	Iteratively Reweighted Least-Squares
KF	Keyframe Fusion
LM	Levenberg-Marquardt
LR	Low-Resolution
MAD	Mean Absolute Deviation

MVS Multi-View Stereo

NLS Non-Linear Least-Squares

ORB Oriented FAST and rotated BRIEF

PGO Pose Graph Optimization

RANSAC RANdom SAmple Consensus

RGB Red-Green-Blue

RGB-D Red-Green-Blue + Depth

SBR Shading-based Refinement

SDF Signed Distance Function

SfM Structure-from-Motion

SfS Shape-from-Shading

SH Spherical Harmonics

SIFT Scale-Invariant Feature Transform

SLAM Simultaneous Localization and Mapping

SR Super-Resolution

SVSH Spatially-Varying Spherical Harmonics

ToF Time-of-Flight

TSDF Truncated Signed Distance Function

VR Virtual Reality

Own Publications

[1] E. Bylow, R. Maier, F. Kahl, and C. Olsson. Combining depth fusion and photometric stereo for fine-detailed 3D models. In *Scandinavian Conference on Image Analysis (SCIA)*, Norrköping, Sweden, June 2019. DOI: 10.1007/978-3-030-20205-722 (cited on pp. 13, 14).

[2] M. Dzitsiuk, J. Sturm, R. Maier, L. Ma, and D. Cremers. De-noising, stabilizing and completing 3D reconstructions on-the-go using plane priors. In *International Conference on Robotics and Automation (ICRA)*, Singapore, May 2017. DOI: 10.1109/ICRA.2017.7989457 (cited on pp. 9, 14, 100).

[3] V. Golyanik, K. Kim, R. Maier, M. Nießner, D. Stricker, and J. Kautz. Multiframe scene flow with piecewise rigid motion. In *International Conference on 3D Vision (3DV)*, Qingdao, China, Oct. 2017. DOI: 10.1109/3DV.2017.00039 (cited on p. 14).

[4] R. Maier, K. Kim, D. Cremers, J. Kautz, and M. Nießner. Intrinsic3D: high-quality 3D reconstruction by joint appearance and geometry optimization with spatially-varying lighting. In *International Conference on Computer Vision (ICCV)*, Venice, Italy, Oct. 2017. DOI: 10.1109/ICCV.2017.338 (cited on pp. 12–15, 61, 125, 137).

[5] R. Maier, R. Schaller, and D. Cremers. Efficient online surface correction for real-time large-scale 3D reconstruction. In *British Machine Vision Conference (BMVC)*, London, United Kingdom, Sept. 2017. DOI: 10.5244/C.31.158 (cited on pp. 12–14, 16, 79, 125).

[6] R. Maier, J. Stückler, and D. Cremers. Super-resolution keyframe fusion for 3D modeling with high-quality textures. In *International Conference on 3D Vision (3DV)*, Lyon, France, Oct. 2015. DOI: 10.1109/3DV.2015.66 (cited on pp. 12–15, 45, 64, 82, 84, 117, 125, 127).

[7] R. Maier, J. Sturm, and D. Cremers. Submap-based bundle adjustment for 3D reconstruction from RGB-D data. In *German Conference on Pattern Recognition (GCPR)*, Münster, Germany, Sept. 2014. DOI: 10.1007/978-3-319-11752-25 (cited on pp. 7, 13, 14, 47, 82).

[8] S. Roy, A. T. D. Grünwald, A. Alves-Pinto, R. Maier, D. Cremers, D. Pfeiffer, and R. Lampe. A non-invasive 3D body scanner and software tool towards analysis of scoliosis. *BioMed Research International (BMRI)*, Volume 2019, Article ID 4715720, May 2019. DOI: 10.1155/2019/4715720 (cited on p. 14).

Bibliography

[9] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011 (cited on p. 3).

[10] S. Agarwal and K. Mierle. *Ceres Solver: Tutorial & Reference*. Google Inc. (cited on pp. 24, 73).

[11] J. Amanatides and A. Woo. A fast voxel traversal algorithm for ray tracing. In *Eurographics*, volume 87 of number 3, pages 3–10, 1987 (cited on p. 33).

[12] A. Avetisyan, M. Dahnert, A. Dai, M. Savva, A. Chang, and M. Nießner. Scan2CAD: learning CAD model alignment in RGB-D scans. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019 (cited on p. 100).

[13] D. Azinović, T.-M. Li, A. Kaplanyan, and M. Nießner. Inverse path tracing for joint material and lighting estimation. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019 (cited on p. 100).

[14] J. Barron and J. Malik. Intrinsic scene properties from a single RGB-D image. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17–24, 2013 (cited on pp. 11, 65).

[15] J. Barron and J. Malik. Shape, illumination, and reflectance from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 37(8):1670–1687, 2015 (cited on pp. 11, 65).

[16] T. Beeler, D. Bradley, H. Zimmer, and M. Gross. Improved reconstruction of deforming surfaces by cancelling ambient occlusion. In *European Conference on Computer Vision (ECCV)*, pages 30–43. Springer, 2012 (cited on pp. 10, 64).

[17] P. Belhumeur, D. Kriegman, and A. Yuille. The bas-relief ambiguity. *International journal of computer vision*, 35(1):33–44, 1999 (cited on pp. 10, 39).

[18] P. Besl and N. McKay. Method for registration of 3-D shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992 (cited on p. 8).

[19] J.-L. Blanco. A tutorial on se(3) transformation parameterizations and on-manifold optimization. *University of Malaga, Technical Report*, 2010 (cited on p. 19).

[20] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017 (cited on p. 100).

[21] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-time camera tracking and 3D reconstruction using signed distance functions. In *Robotics: Science and Systems (RSS)*, 2013 (cited on pp. 8, 47).

[22] Y.-P. Cao, L. Kobbelt, and S.-M. Hu. Real-time high-accuracy three-dimensional reconstruction with consumer RGB-D cameras. *ACM Transactions on Graphics (TOG)*, 37(5):171, 2018 (cited on pp. 7, 8).

[23] S. Choi, Q.-Y. Zhou, and V. Koltun. Robust reconstruction of indoor scenes. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015 (cited on pp. 7, 9, 62, 64, 67, 82, 89, 115, 116).

[24] S. Coorg and S. Teller. Automatic extraction of textured vertical facades from pose imagery, 1998 (cited on pp. 9, 47).

[25] F. Crete, T. Dolmiere, P. Ladret, and M. Nicolas. The blur effect: perception and estimation with a new no-reference perceptual blur metric. In *SPIE*, pages 64920I–64920I, 2007 (cited on pp. 51, 67, 85).

[26] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *ACM Transactions on Graphics (TOG)*, 1996 (cited on pp. 8, 15, 30, 62, 64, 66, 80, 82).

[27] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner. ScanComplete: large-scale scene completion and semantic segmentation for 3D scans. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4578–4587, 2018 (cited on p. 100).

[28] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. Bundlefusion: real-time globally consistent 3D reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (TOG)*, 2017 (cited on pp. 4, 7, 9, 12, 28, 62, 64, 67, 80, 83, 87–89, 92, 101, 115, 116, 119).

[29] Z. DeVito, M. Mara, M. Zollöfer, G. Bernstein, C. Theobalt, P. Hanrahan, M. Fisher, and M. Nießner. Opt: a domain specific language for non-linear least squares optimization in graphics and imaging. *ACM Transactions on Graphics (TOG)*, 2017 (cited on p. 100).

[30] M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. de Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating textures. In *Computer Graphics Forum*, volume 27 of number 2, pages 409–418, 2008 (cited on pp. 9, 47).

[31] R. Or-El, G. Rosman, A. Wetzler, R. Kimmel, and A. M. Bruckstein. RGBD-Fusion: real-time high precision depth recovery. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5407–5416, 2015 (cited on pp. 11, 65).

[32] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *International Conference on Robotics and Automation (ICRA)*, 2012 (cited on pp. 7, 47).

[33] N. Fioraio, J. Taylor, A. Fitzgibbon, L. Di Stefano, and S. Izadi. Large-scale and drift-free surface reconstruction using online subvolume registration. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015 (cited on pp. 11, 82).

[34] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, et al. Building rome on a cloudless day. In *European Conference on Computer Vision (ECCV)*, pages 368–381. Springer, 2010 (cited on p. 3).

[35] Y. Fu, Q. Yan, L. Yang, J. Liao, and C. Xiao. Texture mapping for 3D reconstruction with RGB-D sensor. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4645–4653, 2018 (cited on p. 10).

[36] R. Gal, Y. Wexler, E. Ofek, H. Hoppe, and D. Cohen-Or. Seamless montage for texturing models. In *Computer Graphics Forum*, volume 29 of number 2, pages 479–486, 2010 (cited on pp. 9, 48).

[37] M. Garon, K. Sunkavalli, S. Hadap, N. Carr, and J.-F. Lalonde. Fast spatially-varying indoor lighting estimation. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6908–6917, 2019 (cited on p. 100).

[38] G. Gkioxari, J. Malik, and J. Johnson. Mesh R-CNN. *arXiv preprint arXiv:1906.02739*, 2019 (cited on p. 100).

[39] B. Goldlücke, M. Aubry, K. Kolev, and D. Cremers. A super-resolution framework for high-accuracy multiview reconstruction. *International Journal of Computer Vision*, 106(2):172–191, Jan. 2014 (cited on pp. 9, 48, 64).

[40] B. Haefner, Y. Quéau, T. Möllenhoff, and D. Cremers. Fight ill-posedness with ill-posedness: single-shot variational depth super-resolution from shading. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 164–174, 2018 (cited on p. 11).

[41] Y. Han, J. Lee, and I. So Kweon. High quality shape from a single RGB-D image under uncalibrated natural illumination. In *ICCV*, pages 1617–1624, 2013 (cited on pp. 11, 65).

[42] A. Handa, T. Whelan, J. McDonald, and A. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *International Conference on Robotics and Automation (ICRA)*, 2014 (cited on pp. 9, 89, 116).

[43] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003 (cited on p. 17).

[44] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: using depth cameras for dense 3D modeling of indoor environments. In *International Symposium on Experimental Robotics (ISER)*, volume 20, pages 22–25, 2010 (cited on p. 47).

[45] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research (IJRR)*, 31(5):647–663, 2012 (cited on pp. 7, 87).

[46] B. Horn. Obtaining shape from shading information. *The Psychology of Computer Vision*:115–155, 1975 (cited on pp. 10, 64).

[47] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013 (cited on p. 8).

[48] J. Hou, A. Dai, and M. Nießner. 3D-SIC: 3D semantic instance completion for RGB-D scans. In *arXiv preprint arXiv:1904.12012*, 2019 (cited on p. 100).

[49] J. Huang, A. Dai, L. Guibas, and M. Nießner. 3DLite: towards commodity 3D scanning for content creation. *ACM Transactions on Graphics (TOG)*, 2017 (cited on p. 100).

[50] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger. VolumeDeform: real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision (ECCV)*, pages 362–379. Springer, 2016 (cited on p. 9).

[51] J. Jeon, Y. Jung, H. Kim, and S. Lee. Texture map generation for 3D reconstructed scenes. *The Visual Computer*, 32(6):955–965, 2016 (cited on pp. 10, 64).

[52] O. Kähler, V. Prisacariu, and D. Murray. Real-time large-scale dense 3D reconstruction with loop closure. In *European Conference on Computer Vision (ECCV)*, 2016 (cited on pp. 11, 82, 101).

[53] O. Kähler, V. Prisacariu, C. Ren, X. Sun, P. Torr, and D. Murray. Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 21(11):1241–1250, 2015 (cited on pp. 4, 9, 80, 82, 101).

[54] H. Kato, Y. Ushiku, and T. Harada. Neural 3D mesh renderer. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3907–3916, 2018 (cited on p. 100).

[55] W. Kehl, N. Navab, and S. Ilic. Coloured signed distance fields for full 3D object reconstruction. In *British Machine Vision Conference (BMVC)*, 2014 (cited on p. 10).

[56] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. Real-time 3D reconstruction in dynamic scenes using point-based fusion. In *International Conference on 3D Vision (3DV)*, pages 1–8, 2013 (cited on p. 8).

[57] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for RGB-D cameras. In *International Conference on Intelligent Robots and Systems (IROS)*, 2013 (cited on pp. 7–9, 15, 16, 27, 47, 49, 80, 82, 84, 98).

[58] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for RGB-D cameras. In *International Conference on Robotics and Automation (ICRA)*, 2013 (cited on p. 7).

[59] K. Khoshelham and S. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012 (cited on pp. 26, 50).

[60] K. Kim, J. Gu, S. Tyree, P. Molchanov, M. Nießner, and J. Kautz. A lightweight approach for on-the-fly reflectance estimation. In *International Conference on Computer Vision (ICCV)*, 2017 (cited on p. 100).

[61] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007 (cited on p. 87).

[62] F. Klose, O. Wang, J.-C. Bazin, M. Magnor, and A. Sorkine-Hornung. Sampling based scene-space video processing. *ACM Transactions on Graphics (TOG)*, 34(4):67:1–67:11, July 2015. ISSN: 0730-0301. DOI: 10.1145/2766920 (cited on pp. 10, 64).

[63] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. G2o: a general framework for graph optimization. In *International Conference on Robotics and Automation (ICRA)*, pages 3607–3613. IEEE, 2011 (cited on p. 29).

[64] F. Langguth, K. Sunkavalli, S. Hadap, and M. Goesele. Shading-aware multi-view stereo. In *European Conference on Computer Vision (ECCV)*, pages 469–485. Springer, 2016 (cited on p. 11).

[65] C. LeGendre, W.-C. Ma, G. Fyffe, J. Flynn, L. Charbonnel, J. Busch, and P. Debevec. DeepLight: learning illumination for unconstrained mobile mixed reality. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5918–5928, 2019 (cited on p. 100).

[66] V. Lempitsky and D. Ivanov. Seamless mosaicing of image-based texture maps. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2007 (cited on pp. 9, 48).

[67] B. Lévy, S. Petitjean, N. Ray, and J. Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (TOG)*, 21(3):362–371, 2002 (cited on pp. 15, 34, 53).

[68] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen. Differentiable monte carlo ray tracing through edge sampling. In *ACM Transactions on Graphics (TOG)*, page 222. ACM, 2018 (cited on p. 100).

[69] C. Liu, J. Gu, K. Kim, S. Narasimhan, and J. Kautz. Neural RGB-D sensing: depth and uncertainty from a video camera. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10986–10995, 2019 (cited on p. 99).

[70] C. Liu, K. Kim, J. Gu, Y. Furukawa, and J. Kautz. PlaneRCNN: 3D plane detection and reconstruction from a single image. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4450–4459, 2019 (cited on p. 100).

[71] H. Liu, C. Li, G. Chen, G. Zhang, M. Kaess, and H. Bao. Robust keyframe-based dense SLAM with an RGB-D camera. *arXiv preprint arXiv:1711.05166*, 2017 (cited on p. 12).

[72] Q. Liu-Yin, R. Yu, L. Agapito, A. Fitzgibbon, and C. Russell. Better together: joint reasoning for non-rigid 3D reconstruction with specularities and shading. In *British Machine Vision Conference (BMVC)*, 2016 (cited on p. 11).

[73] S. Lombardi and K. Nishino. Radiometric scene decomposition: scene reflectance, illumination, and geometry from rgb-d images. In *International Conference on 3D Vision (3DV)*, 2016 (cited on p. 65).

[74] M. Loper and M. Black. OpenDR: an approximate differentiable renderer. In *European Conference on Computer Vision (ECCV)*, pages 154–169. Springer, 2014 (cited on p. 100).

[75] W. Lorensen and H. Cline. Marching cubes: a high resolution 3D surface construction algorithm. *ACM Transactions on Graphics (TOG)*, 21(4):163–169, 1987 (cited on pp. 8, 33, 66, 73, 86, 108, 116).

[76] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004 (cited on p. 7).

[77] R. Maier. *Out-of-Core Bundle Adjustment for 3D Workpiece Reconstruction*. Master's thesis, Technische Universität München, München, Germany, Sept. 2013 (cited on p. 13).

[78] M. Meilland and A. Comport. On unifying key-frame and voxel-based dense visual slam at large scales. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3677–3683, 2013 (cited on p. 8).

[79] M. Meilland and A. Comport. Super-resolution 3D tracking and mapping. In *International Conference on Robotics and Automation (ICRA)*, 2013 (cited on pp. 7, 8, 10, 48, 59, 82).

[80] A. Meka, M. Maximov, M. Zollhoefer, A. Chatterjee, H.-P. Seidel, C. Richardt, and C. Theobalt. LIME: Live Intrinsic Material Estimation. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018 (cited on p. 100).

[81] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5115–5124, 2017 (cited on p. 100).

[82] R. Mur-Artal and J. Tardós. ORB-SLAM2: an open-source slam system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017 (cited on pp. 7, 28).

[83] P. Neugebauer and K. Klein. Texturing 3d models of real world objects from multiple unregistered photographic views. In *Computer Graphics Forum*, volume 18 of number 3, pages 245–256, 1999 (cited on pp. 9, 47).

[84] R. Newcombe, A. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. KinectFusion: real-time dense surface mapping and tracking. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011 (cited on pp. 4, 8, 9, 30, 33, 46, 47, 49, 62, 64, 74, 80, 82).

[85] R. Newcombe, D. Fox, and S. Seitz. DynamicFusion: reconstruction and tracking of non-rigid scenes in real-time. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 343–352, 2015 (cited on p. 9).

[86] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)*, 32(6):169, 2013 (cited on pp. 4, 5, 8, 9, 16, 34, 35, 47, 62, 64, 65, 67, 71, 74, 80, 82, 86, 88, 98, 109).

[87] H. Pfister, M. Zwicker, J. Van Baar, and M. Gross. Surfels: surface elements as rendering primitives. In *Proceedings of the 27th annual conference on computer graphics and interactive techniques*, pages 335–342. ACM Press/Addison-Wesley Publishing Co., 2000 (cited on p. 8).

[88] R. Ramamoorthi and P. Hanrahan. A signal-processing framework for inverse rendering. In *ACM Transactions on Graphics (TOG)*, pages 117–128. ACM, 2001 (cited on pp. 10, 40, 65, 69).

[89] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an efficient alternative to SIFT or SURF. In *International Conference on Computer Vision (ICCV)*, pages 2564–2571. IEEE, 2011 (cited on p. 7).

[90] M. Rünz and L. Agapito. Co-fusion: real-time segmentation, tracking and fusion of multiple objects. In *International Conference on Robotics and Automation (ICRA)*, pages 4471–4478. IEEE, 2017 (cited on p. 9).

[91] M. Rünz, M. Buffier, and L. Agapito. Maskfusion: real-time recognition, tracking and reconstruction of multiple moving objects. In *International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20. IEEE, 2018 (cited on p. 9).

[92] R. Schaller. *Combining Voxel Hashing and Keyframe Fusion for Efficient Online Surface Re-Integration*. Master's thesis, Technische Universität München, München, Germany, Nov. 2016 (cited on p. 13).

[93] J. Schonberger and J.-M. Frahm. Structure-from-motion revisited. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4104–4113, 2016 (cited on p. 3).

[94] T. Schops, T. Sattler, and M. Pollefeys. BAD SLAM: bundle adjusted direct rgb-d slam. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 134–144, 2019 (cited on pp. 7, 101).

[95] T. Schöps, T. Sattler, and M. Pollefeys. SurfelMeshing: online surfel-based mesh reconstruction. *arXiv preprint arXiv:1810.00729*, 2018 (cited on pp. 11, 101).

[96] R. Scona, M. Jaimez, Y. Petillot, M. Fallon, and D. Cremers. StaticFusion: background reconstruction for dense RGB-D SLAM in dynamic environments. In *International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018 (cited on p. 9).

[97] M. Slavcheva, M. Baust, D. Cremers, and S. Ilic. KillingFusion: non-rigid 3D reconstruction without correspondences. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3 of number 4, page 7, 2017 (cited on p. 9).

[98] M. Slavcheva, M. Baust, and S. Ilic. SobolevFusion: 3D reconstruction of scenes undergoing free non-rigid motion. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2646–2655, 2018 (cited on p. 9).

[99] N. Snavely, S. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008 (cited on p. 3).

[100] C. Sommer and D. Cremers. Joint representation of primitive and non-primitive objects for 3D vision. In *International Conference on 3D Vision (3DV)*, pages 160–169, 2018 (cited on p. 100).

[101] S. Song and T. Funkhouser. Neural illumination: lighting prediction for indoor environments. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6918–6926, 2019 (cited on p. 100).

[102] S. Song, F. Yu, A. Zeng, A. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1746–1754, 2017 (cited on p. 100).

[103] I. Stamos and P. Allen. 3-d model construction using range and image data. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000 (cited on pp. 9, 47).

[104] F. Steinbrücker, C. Kerl, J. Sturm, and D. Cremers. Large-scale multi-resolution surface reconstruction from RGB-D sequences. In *International Conference on Computer Vision (ICCV)*, 2013 (cited on pp. 7–9, 62).

[105] F. Steinbrücker, J. Sturm, and D. Cremers. Real-time visual odometry from dense RGB-D images. In *International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 719–722. IEEE, 2011 (cited on p. 7).

[106] F. Steinbrücker, J. Sturm, and D. Cremers. Volumetric 3D mapping in real-time on a CPU. In *International Conference on Robotics and Automation (ICRA)*, 2014 (cited on pp. 8, 47, 82).

[107] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. Engel, R. Mur-Artal, C. Ren, S. Verma, A. Clarkson, M. Yan, B. Budge, Y. Yan, X. Pan, J. Yon, Y. Zou, K. Leon, N. Carter, J. Briales, T. Gillingham, E. Mueggler, L. Pesqueira, M. Savva, D. Batra, H. M. Strasdat, R. de Nardi, M. Goesele, S. Lovegrove, and R. Newcombe. The replica dataset: a digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019 (cited on p. 10).

[108] M. Strecke and J. Stückler. EM-Fusion: dynamic object-level SLAM with probabilistic data association. In *International Conference on Computer Vision (ICCV)*, 2019 (cited on p. 9).

[109] J. Stückler and S. Behnke. Multi-resolution surfel maps for efficient dense 3D modeling and tracking. *Journal of Visual Communication and Image Representation*, 25(1):137–147, 2014 (cited on pp. 8, 47).

[110] J. Sturm, E. Bylow, F. Kahl, and D. Cremers. CopyMe3D: scanning and printing persons in 3D. In *German Conference on Pattern Recognition (GCPR)*, 2013 (cited on pp. 8, 9, 46, 47).

[111] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *International Conference on Intelligent Robots and Systems (IROS)*, 2012 (cited on pp. 9, 115, 116).

[112] R. Szeliski. *Computer vision: Algorithms and applications*. Springer, 2010 (cited on p. 17).

[113] M. Teschner, B. Heidelberger, M. Müller, D. Pomerantes, and M. Gross. Optimized spatial hashing for collision detection of deformable objects. In *Vision, Modeling, Visualization (VMV)*, volume 3, pages 47–54, 2003 (cited on p. 35).

[114] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *International Conference on Computer Vision (ICCV)*, volume 98 of number 1, page 2, 1998 (cited on p. 26).

[115] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. *Vision algorithms: theory and practice*:153–177, 2000 (cited on p. 28).

[116] M. Waechter, N. Moehrle, and M. Goesele. Let there be color! Large-scale texturing of 3D reconstructions. In *European Conference on Computer Vision (ECCV)*. 2014 (cited on pp. 9, 48).

[117] M. Weinmann and R. Klein. Advances in geometry and reflectance acquisition (course notes). In *SIGGRAPH Asia 2015 Courses*, page 1. ACM, 2015 (cited on p. 11).

[118] T. Whelan, M. Goesele, S. Lovegrove, J. Straub, S. Green, R. Szeliski, S. Butterfield, S. Verma, and R. Newcombe. Reconstructing scenes with mirror and glass surfaces. *ACM Transactions on Graphics (TOG)*, 37(4):102–1, 2018 (cited on pp. 10, 99).

[119] T. Whelan, H. Johannsson, M. Kaess, J. Leonard, and J. McDonald. Robust real-time visual odometry for dense RGB-D mapping. In *International Conference on Robotics and Automation (ICRA)*, 2013 (cited on pp. 8, 9, 47).

[120] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: spatially extended kinectfusion, 2012 (cited on pp. 4, 7–9, 80).

[121] T. Whelan, M. Kaess, R. Finman, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Real-time large scale dense rgb-d slam with volumetric fusion. *The International Journal of Robotics Research (IJRR)*, 2014 (cited on pp. 11, 82).

[122] T. Whelan, M. Kaess, J. Leonard, and J. McDonald. Deformation-based loop closure for large scale dense RGB-D SLAM. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 548–555. IEEE, 2013 (cited on p. 11).

[123] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison. Elasticfusion: dense slam without a pose graph. In *Robotics: Science and Systems (RSS)*, 2015 (cited on pp. 7, 8, 11, 82, 92, 93, 101).

[124] C. Wu, C. Stoll, L. Valgaerts, and C. Theobalt. On-set performance capture of multiple actors with a stereo camera. *ACM Transactions on Graphics (TOG)*, 32(6):161, 2013 (cited on pp. 10, 64).

[125] C. Wu, K. Varanasi, Y. Liu, H.-P. Seidel, and C. Theobalt. Shading-based dynamic shape refinement from multi-view video under general illumination. In *International Conference on Computer Vision (ICCV)*, pages 1108–1115. IEEE, 2011 (cited on pp. 10, 64).

[126] C. Wu, M. Zollhöfer, M. Nießner, M. Stamminger, S. Izadi, and C. Theobalt. Real-time shading-based refinement for consumer depth cameras. *ACM Transactions on Graphics (TOG)*, 2014 (cited on pp. 11, 46, 63, 65, 77).

[127] H. Wu, Z. Wang, and K. Zhou. Simultaneous localization and appearance estimation with a consumer rgb-d camera. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 22(8):2012–2023, 2016 (cited on pp. 11, 65).

[128] L.-F. Yu, S.-K. Yeung, Y.-W. Tai, and S. Lin. Shading-based shape refinement of rgb-d images. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1415–1422, 2013 (cited on pp. 11, 65).

[129] M. Zeng, F. Zhao, J. Zheng, and X. Liu. A memory-efficient KinectFusion using octree. In *International Conference on Computational Visual Media*, pages 234–241. Springer, 2012 (cited on p. 8).

[130] E. Zhang, M. Cohen, and B. Curless. Emptying, refurnishing, and relighting indoor spaces. *ACM Transactions on Graphics (TOG)*, 35(6):174, 2016 (cited on pp. 9, 65).

[131] R. Zhang, P.-S. Tsai, J. Cryer, and M. Shah. Shape-from-shading: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 21(8):690–706, 1999 (cited on pp. 10, 38, 64).

[132] H. Zhou, B. Ummenhofer, and T. Brox. DeepTAM: deep tracking and mapping. In *European Conference on Computer Vision (ECCV)*, pages 822–838, 2018 (cited on p. 99).

[133] Q.-Y. Zhou and V. Koltun. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Transactions on Graphics (TOG)*, 33(4):155, 2014 (cited on pp. 10, 48, 56, 63, 64, 74).

[134] Q.-Y. Zhou, S. Miller, and V. Koltun. Elastic fragments for dense scene reconstruction. In *International Conference on Computer Vision (ICCV)*, 2013 (cited on p. 82).

[135] M. Zollhöfer, A. Dai, M. Innmann, C. Wu, M. Stamminger, C. Theobalt, and M. Nießner. Shading-based refinement on volumetric signed distance functions. *ACM Transactions on Graphics (TOG)*, 34(4):96, 2015 (cited on pp. 11, 63, 65, 71, 74–77, 107–110).

[136] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb. State of the art on 3D reconstruction with RGB-D cameras. In *Computer Graphics Forum*, volume 37 of number 2, pages 625–652. Wiley Online Library, 2018 (cited on p. 7).

About the Author

Robert Maier received his Bachelor's degree (2010) and his Master's degree (2013, with distinction) in Computer Science from the Technical University of Munich (Germany). He spent an exchange semester at UPC Barcelona (Spain) in 2011/2012 and worked as a Computer Vision Research Intern at Siemens Corporate Research in Princeton, NJ (USA) in 2012/2013. After graduation, he worked as a Computer Vision Engineer at ImFusion (Munich) from 2013 to 2014. In May 2014, Robert became a Research Assistant and PhD Candidate at the Chair for Computer Vision and Artificial Intelligence at TU Munich, headed by Professor Daniel Cremers. Within the scope of his PhD activities, he spent some time as a Computer Vision Research Intern at NVIDIA Research in Santa Clara, CA (USA) and at Facebook Reality Labs in Redmond, WA (USA). With a completed Computer Science Expert degree (2007), he has many years working experience as a software engineer.

His research interests include (real-time) RGB-D based 3D Reconstruction, Visual SLAM, Visual Odometry, Shape-from-Shading and Bundle Adjustment techniques. While pursuing his PhD degree, he has worked on various topics evolving around 3D reconstruction from low-cost RGB-D sensors, resulting in altogether 8 peer-reviewed conference and journal publications. As a teaching assistant at TUM, he supervised various lab courses and lecture exercises, along with altogether 10 student projects (Master theses, etc.). He has served as a reviewer for several Computer Vision and Robotics conferences (ICCV, 3DV, RSS, ICRA, IROS, ACCV) and journals (IEEE Transactions on Robotics, IEEE Transactions on Image Processing, Elsevier Computer Vision and Image Understanding), with an Outstanding Reviewer Award at ACCV 2016.

Robert has a remarkably low *Erdös-Bacon number* of 7, with an Erdös number of 5 and a Bacon number of 2 (due to appearing in the 2016 Hollywood film *Snowden*).