Shading-based Refinement on Volumetric Signed Distance Functions

Michael Zollhöfer[1,4] Angela Dai[2] Matthias Innmann[1] Chenglei Wu[3]
Marc Stamminger[1] Christian Theobalt[4] Matthias Nießner[2]
[1]University of Erlangen-Nuremberg [2]Stanford University [3]ETH Zurich [4]MPI Informatics

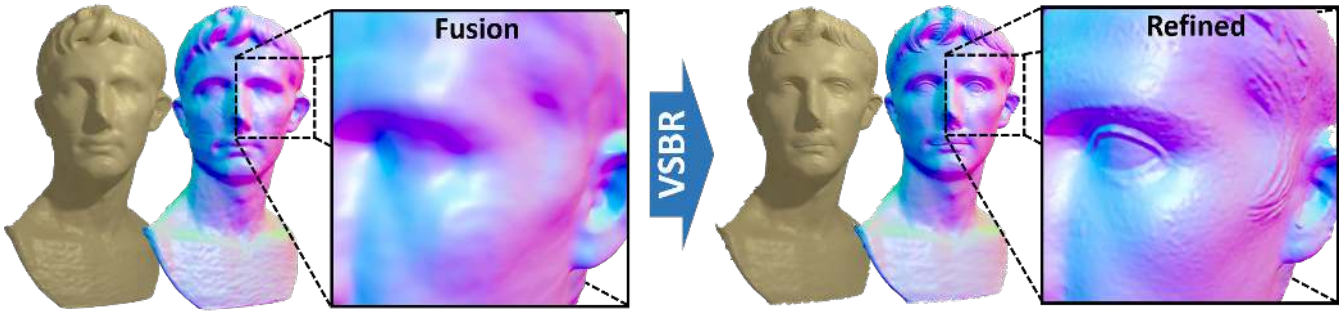

Figure 1: *Our method obtains fine-scale detail through volumetric shading-based refinement (VSBR) of a distance field. We scan an object using a commodity sensor – here, a PrimeSense – to generate an implicit representation. Unfortunately, this leads to over-smoothing. Exploiting the shading cues from the RGB data allows us to obtain reconstructions at previously unseen resolutions within only a few seconds.*

Abstract

We present a novel method to obtain fine-scale detail in 3D reconstructions generated with low-budget RGB-D cameras or other commodity scanning devices. As the depth data of these sensors is noisy, truncated signed distance fields are typically used to regularize out the noise, which unfortunately leads to over-smoothed results. In our approach, we leverage RGB data to refine these reconstructions through shading cues, as color input is typically of much higher resolution than the depth data. As a result, we obtain reconstructions with high geometric detail, far beyond the depth resolution of the camera itself. Our core contribution is shading-based refinement directly on the implicit surface representation, which is generated from globally-aligned RGB-D images. We formulate the inverse shading problem on the volumetric distance field, and present a novel objective function which jointly optimizes for fine-scale surface geometry and spatially-varying surface reflectance. In order to enable the efficient reconstruction of sub-millimeter detail, we store and process our surface using a sparse voxel hashing scheme which we augment by introducing a grid hierarchy. A tailored GPU-based Gauss-Newton solver enables us to refine large shape models to previously unseen resolution within only a few seconds.

CR Categories: I.4.1 [Image Processing and Computer Vision]: Digitization and Image Capture—Scanning;

Keywords: shading-based refinement, 3D reconstruction

1 Introduction

The advent of low-cost RGB-D cameras, such as the Microsoft Kinect, triggered the development of new algorithms that allow consumers to densely scan objects at real-time frame rates. A prominent example of this line of research is Kinect Fusion [Newcombe et al. 2011; Izadi et al. 2011] and its extensions [Roth and Vona 2012; Whelan et al. 2012; Chen et al. 2013; Nießner et al. 2013], which led to significant impact in the computer graphics, vision, and HCI communities. Their efficient reconstruction of 3D environments is beneficial to a large variety of applications, ranging from content creation to augmented reality scenarios.

Core to these approaches is the underlying surface representation of a truncated signed distance field (TSDF) [Curless and Levoy 1996]. This representation stores the distance values to the closest surface point in 3D in a voxel grid, and has several advantages compared to alternative models, such as point-based or mesh-based representations, in particular if efficiency is the goal. It enables efficient alignment and fusion of scans, while systematically considering the drastic noise and distortions in depth data of many RGB-D cameras. Further, it allows scan integration without complex connectivity and topology handling, and is the basis of many surface extraction algorithms. Unfortunately, despite these benefits, aligning and integrating depth scans on a TSDF leads to strong over-smoothing of the surfaces, as the depth data is fused projectively using a weighted average from different viewing directions. This is further compounded by drift due to alignment errors of input depth frames. Thus, while TSDFs efficiently regularize noise, resulting reconstructions lack fine-scale geometric detail. Additionally, scanning quality is limited by the limits of the depth cameras themselves. Most of them deliver images of much lower depth resolution than RGB resolution; depth is also very noisy, and may contain systematic distortions. Overall, the benefits of implicit surface representations have brought them to be prevalent in online and offline 3D reconstruction approaches, but they fail to capture fine-scale detail for noisy depth input. Note that this extends beyond TSDFs, as other implicit functions have been used as representations for 3D reconstruction; e.g., [Carr et al. 2001; Kazhdan et al. 2006; Fuhrmann and Goesele 2014].

In this paper, we address the problem of over-smoothed TSDFs, and propose a new approach to efficiently reconstruct fine-scale detail on them. To this end, we leverage RGB data to refine the implicit

surface representation using shading cues. Since color input is typically of much higher resolution than the depth data, we can obtain reconstructions with geometric detail far beyond the depth resolution of the camera itself. Operating directly *on the implicit surface* representation, as opposed to previous approaches on meshes, has several distinct advantages. The regular structure of the TSDF naturally yields a spatially uniform sampling, in contrast to the overhead required to regularly sample a mesh. Further, we employ a sparse hashing-based scheme to store the TSDF, providing efficient storage and fast refinement. Additionally, operating on an implicit surface representation gives our approach versatility, as implicit surface representations are used in many other reconstruction approaches; e.g., using other active triangulation sensors, laser scanning, or passive image-based reconstruction. In addition to geometric refinement of consumer RGB-D sensor input, we also demonstrate clear improvements in other settings in Section 7.

From globally-aligned color and depth input, we estimate the incident lighting distribution, and present a novel objective function to jointly optimize for geometric refinement and dense albedo in general and unconstrained environments. To solve this non-linear optimization problem, we present a tailored, parallel Gauss-Newton solver. Our TSDF structure also uses a hierarchy of resolutions, which is required for efficient convergence. All our data structures and algorithms are designed for efficient execution on the GPU. Our new refinement method is geared towards RGB-D scanning, improves over the state-of-the-art in many ways, and is based on the following contributions:

- a formulation of the inverse shading problem on a TSDF, allowing for joint optimization of fine-scale geometric detail and dense, spatially-varying albedo (Section 5)
- an extension of the sparse voxel hashing scheme to support a hierarchy of resolutions, which provides efficient storage and fast refinement of TSDFs (Section 4)
- a GPU-based non-linear Gauss-Newton solver crafted to solve for shading-based refinement on tens of millions of variables in a few seconds (Section 6)

2 Related Work

Implicit functions are popular scene representations used in many 3D reconstruction algorithms [Hoppe et al. 1992; Carr et al. 2001; Kazhdan et al. 2006; Fuhrmann and Goesele 2014]. Implicit models facilitate partial scan alignment and integration without complex topology handling (as needed for meshes). A popular variant is the signed distance field (SDF) model which stores distances to the surface on a voxel grid [Curless and Levoy 1996]. It was used for offline reconstruction of large models from partial range scans [Levoy et al. 2000], but has also been used in real-time structured light scanning [Rusinkiewicz et al. 2002] where its efficient storage and simple update is beneficial.

Recently, new cheap consumer-grade RGB-D cameras, such as the triangulation-based Kinect or time-of-flight (TOF) cameras, have become increasingly popular for 3D scanning [Henry et al. 2012]. Some hand-held RGB-D scanning approaches resort to point-based scene models [Keller et al. 2013; Weise et al. 2009]. However, SDFs are the more widely used representation in recent real-time methods. The Kinect Fusion algorithm [Newcombe et al. 2011; Izadi et al. 2011] was one of the first to do online alignment and integration of RGB-D depth data using weighted averaging of partial scans in a truncated SDF. Several extensions of the approach were proposed; e.g., a direct variational depth-to-SDF alignment instead of ICP [Bylow et al. 2013], or to use a combination of such alignment with color optimization of aligned partial SDFs [Kehl et al. 2014]. The SDF model also simplifies consideration of the often drastic noise in the depth data of consumer depth cameras when integrating the scene model. Kinect Fusion stores the implicit model on a regular grid, which limits scalability to larger scenes. To scan larger scenes, the use of shifting volumes [Whelan et al. 2012], hierarchical grids [Chen et al. 2013] or sparse voxel hashing data structures [Nießner et al. 2013] was proposed. Unfortunately, most RGB-D camera scanning approaches suffer from over-smoothed reconstructions due to the averaging during scan integration; the reconstructed geometric detail is thus not sufficient for many professional applications. This problem is amplified by the fact that most RGB-D cameras have a very low depth resolution, inhibiting capture of high geometric detail. Some methods attempt to overcome the depth camera resolution limit by time-consuming multi-depth-frame super-resolution from nearby RGB-D images [Schuon et al. 2009; Cui et al. 2010], but reconstructions are still of limited detail. Other techniques improve depth resolution by leveraging the fact that most RGB-D cameras have drastically higher RGB resolution than depth resolution. Another approach is to assume the alignment of color and depth edges, which can be exploited in a joint edge-preserving upsampling filter, such as a bilateral or multi-lateral filter [Lindner et al. 2007; Kopf et al. 2007; Chan et al. 2008; Dolson et al. 2010; Richardt et al. 2012], or explicitly phrased in an optimization problem [Park et al. 2011; Diebel and Thrun 2006]. Despite more detailed and less noisy results, many of the approaches suffer from texture-copy artifacts since assumptions about lighting are often wrong and shading effects are mistaken for geometry detail. Our approach also exploits the higher RGB resolution of consumer depth cameras, but takes inspiration from recent progresses in scene reconstruction from single or multi-view RGB images only. Recent methods for 3D scene reconstruction from a hand-held RGB camera use a combination of sparse feature tracking, structure-from-motion, and stereo to reconstruct the scene geometry by integration in an SDF [Newcombe and Davison 2010; Pradeep et al. 2013]. However, the reconstructed models show similar over-smoothing as the aforementioned RGB-D results. This is not only due to averaging in the SDF, but also due to the stereo reconstruction itself, which often requires strong regularization to find image correspondences from scene texture [Seitz et al. 2006; Scharstein et al. 2014].

Shape-from-shading (SfS) is able to overcome some of these resolution limits, and also succeeds on texture-less objects [Horn 1975; Zhang et al. 1999]. SfS is well-understood, particularly when surface reflectance and light source positions are known [Prados and Faugeras 2005]. It can also refine coarse image-based shape models, for instance from multi-view stereo [Beeler et al. 2012], even if they were captured under general uncontrolled lighting with several cameras [Wu et al. 2011; Wu et al. 2013]. To this end, illumination and albedo distributions, as well as refined geometry, are found via inverse rendering optimizations. SfS is inherently ill-posed in uncontrolled scenes, and achieving compelling results requires strong scene and lighting assumptions, as well as computationally complex algorithms, particularly to solve hard non-linear inverse rendering optimizations.

Several reconstruction approaches use prior models on reflectance to alleviate some of these problems; e.g., [Haber et al. 2009]. An alternative strategy is photometric stereo, which uses images of a scene captured under different controlled illumination [Mulligan and Brolly 2004; Hernández et al. 2008; Ghosh et al. 2011; Debevec 2012; Nehab et al. 2005]. However, these approaches depend on complex controlled lighting setups, which are not available for scanning in general environments. Other methods create super-resolution texture maps from multi-view RGB images that were aligned on coarse image-based 3D models [Goldluecke et al. 2014]; however, they do not refine reconstructed geometry. Since they jointly consider cues from multiple aligned RGB images, many of

these image-based approaches produce results of higher detail than current RGB-D camera methods, but computation times are very long even on moderate shape resolutions.

Some methods thus combine image-based stereo and RGB-D depth for reconstruction [Nair et al. 2013], but comparably high geometric detail to that obtained by shading-based refinement is usually not attained. Zhou et al. [2014] combine Kinect Fusion with a texture warping and frame bundling approach. This yields ghost-free textures warped to a coarse SDF model, which means that the textured model is not geometrically accurate. Recently, shape-from-shading under general illumination was used to up-sample and refine a single RGB-D depth image [Han et al. 2013; Yu et al. 2013] at offline rates, and Wu et al. [2014] refine a single RGB-D camera depth frame at video rate. However, single frame RGB-D methods are specialized to the image domain and cannot process a 3D reconstruction. Unfortunately, simply integrating their results in a Kinect Fusion style, causes a notable loss of the refined detail.

Current work on intrinsic image and video decomposition [Chen and Koltun 2013; Lee et al. 2012] deals with the problem of separating albedo and shading. These methods employ sophisticated albedo regularization strategies, but are computationally more expensive and do not easily extend to the volume.

3 Overview

In this section, we provide a brief overview of our method. We first capture input color and depth using commodity sensors (e.g., Microsoft Kinect). This yields a sequence of depth images $\mathcal{D}_i$ and color images $\mathcal{C}_i$, which we use to generate an implicit surface representing the scanned scene (Section 4.3). We follow Curless and Levoy [1996], and use a truncated signed distance function to represent the implicit surface. From $\mathcal{D}_i$ and $\mathcal{C}_i$, we obtain an initial truncated signed distance field $\mathbf{D}$, from which we then solve for the refined signed distance field $\tilde{\mathbf{D}}$. To this end, we augment each voxel $\mathbf{v}$ with the shading attributes – i.e., albedo $\mathbf{a}(\mathbf{v})$ and refined distance $\tilde{\mathbf{D}}(\mathbf{v})$ – needed for our geometric refinement. In addition, we build a hierarchy of sparse voxel grids with varying voxel sizes to account for the differing depth and color input resolutions. This enables our shading-based refinement to run efficiently in a coarse-to-fine fashion. Note that to obtain sharp color data in the fused model, we optimize for the rigid camera poses $\mathcal{T}_i$ bringing each frame i back to the space of the first frame (Section 4.4).

We then run our shading-based refinement on the sparse voxel hierarchy (Section 5). To accommodate general and uncontrolled lighting environments, we continually estimate incident irradiance. We rephrase the inverse shading problem for an implicit TSDF surface, simultaneously optimizing for refined surface geometry and dense, spatially-varying albedo. These dense albedos, as opposed to coarsely aligned clusters of albedos as in many previous shading-based refinement methods (e.g., [Wu et al. 2011; Wu et al. 2013]), further inform the lighting estimation and enable more accurate shape refinement results. We optimize our new objective function with a custom GPU-based parallel Gauss-Newton optimizer which allows to solve for tens of millions of variables within a few seconds. This yields the refined scene model $\tilde{\mathbf{D}}$ with fine-scale detail from the RGB data mapped to the shape model. Fig. 2 shows an overview of our reconstruction pipeline.

4 Implicit Surface Generation

We follow Curless and Levoy [1996], and represent surface geometry of a scanned scene using a truncated signed distance function (TSDF), denoted as $\mathbf{D}$. The TSDF is defined as a piecewise linear function, where supporting data is stored in a regular grid, composed of a set of voxels $\{\mathbf{v}_{i,j,k}\}$. Every voxel is a sampled point of $\mathbf{D}$, containing a (signed) distance data point $\mathbf{D}(\mathbf{v})$. Accessing the volumetric distance field $\mathbf{D}$ for non-discrete points $\mathbf{p} \in \mathbb{R}^3$ is done through tri-linear interpolation. Typically, we are interested in points $\mathbf{p}$ on the iso-surface of $\mathbf{D}$, where $\mathbf{D}(\mathbf{p}) = 0$, which we will denote as $\mathbf{D}_0$. Defining geometry as an implicit signed distance function has many advantages. A TSDF enables the regularization and noise-aware integration of noisy input depth data (e.g., obtained from commodity RGB-D sensors), and is a versatile representation that has been used in many scanning pipelines and surface reconstruction approaches. Additionally, it is easy to convert to other shape representations. Similar to Kinect Fusion [Newcombe et al. 2011; Izadi et al. 2011], we incrementally align and integrate RGB-D images to get an initial coarse signed distance field $\mathbf{D}$ (Section 4.3) which is subsequently geometrically refined (Section 5).

4.1 Extended Sparse Voxel Data Structure

Our ultimate goal is to obtain a refined distance field $\tilde{\mathbf{D}}$ by utilizing color data and shape-from-shading under general lighting conditions. We thus require a sufficiently high voxel resolution to capture all geometric detail from RGB images. To this end, we aim to set the voxel resolution such that the 2D projections of voxels are smaller than the pixel size of input color images; as shown by our results, this is typically well below 1 mm^3 (cf. Section 7). For efficient surface reconstruction and refinement at this extremely high spatial resolution, we store and process all data structures on the GPU.

In addition to the signed distance value, each voxel stores values of additional attributes for color $\mathbf{C}(\mathbf{v})$ and integration weight $\mathbf{W}(\mathbf{v})$, both of which are explained in Section 4.3. In contrast to previous work, our voxels also need to store additional attributes for shape refinement (see Section 5); i.e., a luminance albedo $\mathbf{a}(\mathbf{v})$ and a refined signed distance value $\tilde{\mathbf{D}}(\mathbf{v})$. Our per-voxel data structure is depicted in Listing 1. Note that the requirement to store both $\mathbf{D}$ and $\tilde{\mathbf{D}}$ is due to the stabilization term in our objective function (cf. Section 5.2).

The above considerations show that we reconstruct at a very high spatial resolution, matching high-end offline scanning systems [Levoy et al. 2000], even while book-keeping more data per voxel. Since we do not want to sacrifice computational efficiency, we exploit the capabilities of modern GPUs. To store and process this high-resolution data on the GPU, we store voxels sparsely using an extension of the voxel hashing scheme of Nießner et al. [2013]. The core idea is to use a spatial hashing function to reference a relatively small number of voxel blocks close to 3D locations where the actual iso-surface resides; in our implementation, we use 4^3 voxel blocks. Note that a sparse representation is key to efficient storage and fast shading-based refinement.

Instead of a single, high-resolution voxel grid, as in [Nießner et al. 2013], we maintain a hierarchy of sparse grids with varying voxel sizes to facilitate the later shading-based refinement. Since color resolution is typically much higher than the corresponding input depth, we set the voxel resolution on the coarsest level to correspond to the input depth resolution, and the voxel size at the finest level to correspond to the input RGB data. This enables an efficient coarse-to-fine optimization, ultimately allowing us to obtain the refined signed distance function $\tilde{\mathbf{D}}$ at the highest resolution (see Section 5.2). Note that our hierarchical TSDF model stores both the initial integration result $\mathbf{D}$ and the final refined result $\tilde{\mathbf{D}}$. Before minimizing the shading-based refinement objective (see Section 5.2), we initialize $\tilde{\mathbf{D}}$ with $\mathbf{D}$.

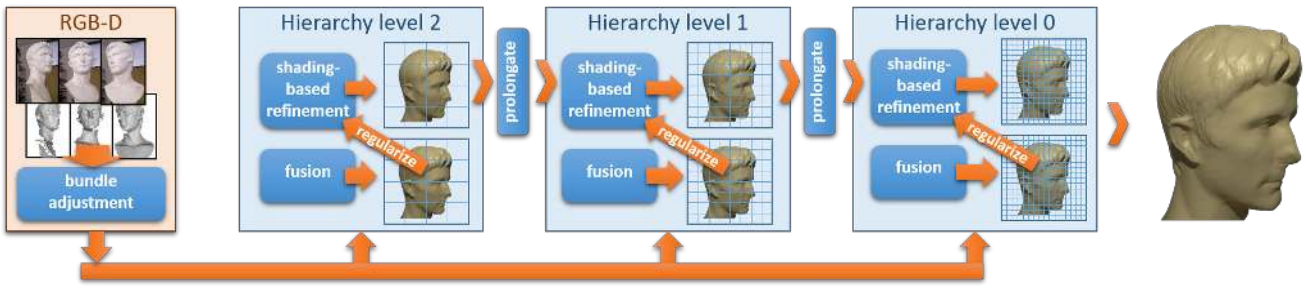

Figure 2: *Our reconstruction pipeline. With a set of RGB-D images as input, we first run bundle adjustment to align the RGB-D frames (if necessary). Based on this, an initial volumetric signed distance field of the surface is computed by depth fusion. Afterwards, hierarchical shading-based refinement based on RGB input is used to add fine scale detail to the reconstruction.*

4.2 Allocation

Allocation of the sparse voxel hierarchy begins on the coarsest level. New voxel blocks are allocated based upon input depth, and then depth and color data are fused into the coarsest signed distance field as described in Section 4.3; i.e., $\mathbf{D}$, $\mathbf{C}$, and $\mathbf{W}$ are computed. We then optimize for refined distances $\tilde{\mathbf{D}}$ and reflectance values $\mathbf{a}$ for the coarsest level (Section 5).

```
struct Voxel {
    //! integrated values
    float  signedDistance;  // D(v)
    uchar3 colorRGB;        // C(v)
    uchar  weight;          // W(v)

    //! unknowns of our objective
    float luminanceAlbedo   // a(v)
    float refinedDistance   // D'(v)

    // chromaticity and intensity are
    // computed from C(v) on-the-fly
};
```

Listing 1: *Voxel data structure: the set of voxels defines* $\mathbf{D}$, $\mathbf{C}$, *and* $\mathbf{W}$. *We solve for albedos* $\mathbf{a}$ *and the refined distance field* $\tilde{\mathbf{D}}$ *in the refinement step (see Section 5.2).*

For each level q of the hierarchy beyond the coarsest level, we then repeat the following process in a coarse-to-fine manner. Voxel blocks are allocated on level q based upon the refined depth of the previous level $q+1$; that is, for each voxel block of level $q+1$, if the minimum $|\tilde{\mathbf{D}}|$ in the sub-block corresponding to a voxel block on level q is less than t_{trunc}, then the block is allocated. Depth and color data are then integrated to provide each voxel $\mathbf{v}$ with its signed distance value $\mathbf{D}(\mathbf{v})$ and color value $\mathbf{C}(\mathbf{v})$. The refined distance value $\tilde{\mathbf{D}}(\mathbf{v})$ as well as the reflectance value $\mathbf{a}(\mathbf{v})$ of hierarchy level q are initialized by tri-linear interpolation of the refined values of level $q+1$, and then we optimize for $\tilde{\mathbf{D}}$ and $\mathbf{a}$ of level q. Pseudo-code for allocation and optimization on the sparse voxel hierarchy is given in Figure 3.

4.3 Integration of Depth Images

Similar to Kinect Fusion [Newcombe et al. 2011; Izadi et al. 2011], we incrementally compute the fused implicit signed distance field $\mathbf{D}$ by integrating depth data from the individual depth images $\mathcal{D}_i$. In addition, we integrate color data from input RGB images $\mathcal{C}_i$ to obtain the volumetric color function $\mathbf{C}$ in the same manner. A rigid camera pose $\mathcal{T}_i$ for frame i, bringing $\mathcal{D}_i$ and $\mathcal{C}_i$ to the first frame camera space, is computed in a pose optimization step (see Section 4.4). In this section, we assume that the alignment of an input sequence with n frames is given by a set of rigid transformation matrices $\mathcal{T} = \{\mathcal{T}_1, ..., \mathcal{T}_n\}$.

For a given voxel $\mathbf{v}$ in the fused scene model $\mathbf{D}$, the corresponding signed distance value $\mathbf{D}(\mathbf{v})$ can be computed as a weighted average with respect to n input frames of a given scanning sequence

$$\mathbf{D}(\mathbf{v}) = \frac{\sum_{i=1}^{n} w_i(\mathbf{v}) d_i(\mathbf{v})}{\mathbf{W}(\mathbf{v})}, \qquad \mathbf{W}(\mathbf{v}) = \sum_{i=1}^{n} w_i(\mathbf{v}),$$

where $d_i(\mathbf{v})$ is the projective distance (along the z axis) between a voxel and the i-th depth frame $\mathcal{D}_i$ and $w_i(\mathbf{v})$ is the integration weight for a sample of an input frame. For a given rigid transformation $\mathcal{T}_i$ between the current frame i and the first frame (cf. Section 4.4), depth camera projection matrix π_d, and a maximum truncation t_{trunc}, we define $d_i(\mathbf{v})$ as

$$\begin{aligned} d_i(\mathbf{v}) &= \text{sgn}(\hat{d}_i)(\mathbf{v}) \cdot \min(|\hat{d}_i(\mathbf{v})|, t_{\text{trunc}}), \quad \text{where} \\ \hat{d}_i(\mathbf{v}) &= (\mathcal{T}_i^{-1}\mathbf{v})_z - \mathcal{D}_i(\pi_d(\mathcal{T}_i^{-1}\mathbf{v}))_z. \end{aligned}$$

While most real-time systems (e.g., [Newcombe et al. 2011; Izadi et al. 2011]) use a uniform weighting ($w_i(\mathbf{v}) = 1,\ \forall i, \mathbf{v}$), we define our weighting function w based on the distance to the surface as well as the angle $\alpha_i(\mathbf{v})$ between the viewing direction and the surface normal [Curless and Levoy 1996; Bylow et al. 2013] in the depth image at the projected voxel location:

$$w_i(\mathbf{v}) = t_{\text{angle}} \cdot \phi(\alpha_i(\mathbf{v})) + t_{\text{dist}} \cdot \phi(|d_i(\mathbf{v})|),$$

```
▷ coarse-to-fine optimization
procedure OPTIMIZE(int numLevels)
    ▷ coarsest level follows [Nießner et al. 2013]
    ALLOCATECOARSEST(numLevels-1)
    INTEGRATECOARSEST(numLevels-1)

    for q = numLevels − 2 to 0 do
        ALLOCCOARSETOFINE(q, q+1)
        INTEGRATEDEPTHANDCOLOR(q)
        SAMPLECOARSETOFINE(q, q+1)
        REFINE(q)
    end for
end procedure
```

Figure 3: *Pseudo-code for coarse-to-fine optimization on the sparse voxel grid hierarchy.*

where t_{angle} and t_{dist} are user-defined constants, and $\phi(x)$ is a robust kernel with a user-defined constant t_{rob} ($2 \sim 5$ in our experiments)

$$\phi(x) = 1/(1 + t_{\text{rob}} \cdot x)^3.$$

The first term of w gives scene regions that are seen most head-on a higher priority. This reflects the lower measurement uncertainty of most depth cameras for such regions. The second term gives values of voxels that are further from the surface a lower weight during integration. We integrate color data analogously to depth: for all RGB input frames $\mathcal{C}_i$, we use the weighted average as shown above. Note that we keep the same weighting $w_i(\mathbf{v})$, since color integration benefits from this weighting strategy similar to depth data. Later, per-voxel color values are used as shading constraints to determine the refined distance field $\tilde{\mathbf{D}}$ (see Section 5.2).

4.4 Pose Optimization

As we are fusing data from many input frames into a compact surface representation, we must avoid drift while integrating the surface to guarantee sharp color data. This problem is stronger on larger scenes where the camera moves further. For such scenes, on the coarsest hierarchy level only, and before the coarse-to-fine refinement commences, we therefore perform a two-step global pose optimization in addition to the integration steps from Section 4.3. This step jointly solves for all camera poses $\mathcal{T} = \{\mathcal{T}_1, ..., \mathcal{T}_n\}$, with $\mathcal{T}_i$ as the rigid transform from the i-th frame to the first frame.

Sparse Bundle Adjustment Step First, we perform a sparse bundle adjustment step, similar to traditional bundle adjustment of RGB images [Triggs et al. 2000; Snavely et al. 2006; Agarwal et al. 2011]; however, we formulate our pose optimization to take advantage of the depth channel as well. The camera poses are initialized with frame-to-frame ICP on the depth data. Features are then detected in the color input using a SIFT keypoint detector [Lowe 2004], and for each pair of images which overlap (using the initial trajectory), feature correspondences are found based upon their SIFT descriptor distance. Using the depth data, we can project each feature to a 3D location $\mathbf{p}$ in the camera space of that frame. We then solve for the camera poses by minimizing the following alignment error:

$$E_{\text{sparse}}(\mathcal{T}) = \sum_{i,j}^{\#\text{frames}} \sum_{k}^{\#\text{corresp.}} \left\| \mathcal{T}_i \mathbf{p}_{ik} - \mathcal{T}_j \mathbf{p}_{jk} \right\|_2^2,$$

where $\mathbf{p}_{ik}$ is the 3D location of the k-th feature correspondence shared by frame i and frame j. This is a non-linear least-squares objective, which we minimize using the Levenberg-Marquardt algorithm. However, since the feature correspondences are fixed, the optimization results can only be as good as the feature correspondences, which may not be all perfect. As a mismatch of a few pixels may still lead to blurred color fusion, a dense bundle adjustment step is performed afterwards.

Dense Bundle Adjustment Step In the dense bundle adjustment step, instead of finding features and correspondences, we use the pixel information of $\mathcal{D}_i$ and $\mathcal{C}_i$ densely, and optimize for maximal photo-consistency and minimal re-projection error. Here, $\mathcal{I}_i$ is the luminance image corresponding to $\mathcal{C}_i$. We minimize the objective

$$E_{\text{dense}}(\mathcal{T}) = w_{\text{color}} E_{\text{color}}(\mathcal{T}) + w_{\text{geometric}} E_{\text{geometric}}(\mathcal{T}),$$

where w_{color} and $w_{\text{geometric}}$ weight the photo-consistency term and the geometric error term, respectively.

The photo-consistency term, similar to [Zhou and Koltun 2014], is defined as follows:

$$E_{\text{color}}(\mathcal{T}) = \sum_{i,j}^{\#\text{frames}} \sum_{k}^{\#\text{pixels}} \left\| \mathcal{I}_i(\pi_c(\mathbf{p}_{ik})) - \mathcal{I}_j(\pi_c(\mathcal{T}_j^{-1}\mathcal{T}_i \mathbf{p}_{ik})) \right\|_2^2.$$

Here, π_c (π_d) is the perspective transform of the color (depth) camera. That is, for each pixel k in each frame i with associated camera space position $\mathbf{p}_{ik}$ and color intensity $\mathcal{I}_i(\pi_c(\mathbf{p}_{ik}))$, the projection of $\mathbf{p}_{ik}$ into every other color image should produce a similar intensity. The geometric error term, a point-plane energy, is defined as:

$$E_{\text{geometric}}(\mathcal{T}) = \sum_{i,j}^{\#\text{frames}} \sum_{k}^{\#\text{pixels}} \left[\mathbf{n}_{ik}^T \cdot (\mathbf{p}_{ik} - \mathcal{T}_i^{-1}\mathcal{T}_j \pi_d^{-1} \left(\mathcal{D}_j \left(\pi_d(\mathcal{T}_j^{-1}\mathcal{T}_i \mathbf{p}_{ik}) \right) \right)) \right]^2.$$

That is, for each pixel k in frame i with associated camera space position $\mathbf{p}_{ik}$ and corresponding surface normal $\mathbf{n}_{ik}$, the projection of $\mathbf{p}_{ik}$ into every other depth image should produce a 3D position which (when projected into the camera space of frame i) agrees with $\mathbf{p}_{ik}$.

Note that for both terms we also discard invalid correspondences – e.g., if there is no associated depth value for a pixel or if $\pi_c(\mathbf{p}_{ik})$ is out of image bounds – and only project a pixel from frame i to frame j if the frames overlap (as computed by $\mathcal{T}$). We also subsample the depth and color images by a factor of 8 on pixel level for efficiency purposes. We minimize the non-linear least-squares objective E_{dense} using Gauss-Newton optimization, iteratively giving more weight to E_{color}. $\mathcal{T}$ is initialized with the results from the sparse bundling step.

5 Refinement of Signed Distance Functions

Our main goal is to refine $\mathbf{D}$ at the current hierarchy level, which lacks fine-scale detail, to reflect the fine-scale RGB detail in the geometry, and store this in $\tilde{\mathbf{D}}$. Previous shading-based refinement methods use mesh models which cause additional book keeping overhead on the GPU, and require more effort to ensure spatially regular sampling. In contrast, our volumetric data structure provides a uniform sampling of the underlying surface.

Prior to optimization, we initialize $\tilde{\mathbf{D}}$ with $\mathbf{D}$. As the surface is defined by the iso-surface $\mathbf{D}_0$, we aim to determine the refined geometry $\tilde{\mathbf{D}}_0$. To this end, we constrain points on $\mathbf{D}_0$ by an inverse rendering and shading assumption on the observed color data $\mathbf{C}$. This constraint relies on an efficient formulation of light transport to model the illumination in an environment and the reflection on the surface. Similar to previous approaches, we assume surfaces to be predominantly Lambertian, allowing us to estimate incident irradiance at a point $\mathbf{p}$ as a function of the (locally) parametrized surface normal $\mathbf{n}$. In the following, we explain how incident illumination in the scene can be directly computed from $\mathbf{D}$ and the fused aligned color images (Section 5.1). The estimated illumination is used in a combined non-linear optimization for refined geometry and dense spatially-varying albedo (Section 5.2). An overview of the shading-based refinement stage is shown in Fig. 4.

Lighting Model For Lambertian reflectance, the incident irradiance at a point $\mathbf{p}$ is known to be smooth, and can be efficiently represented using spherical harmonics [Ramamoorthi and Hanrahan 2001]. Typically, a good approximation is given by the first nine spherical harmonics basis functions; i.e., up to 2nd order. Similar to previous methods (e.g., [Wu et al. 2011; Wu et al. 2014]), we estimate the lighting based on a luminance attribute $\mathbf{I}(\mathbf{v})$ for each voxel,

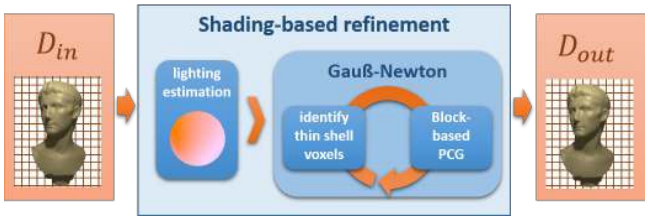

Figure 4: *At each hierarchy level, lighting is estimated based on the fused model and the spatially-varying albedo estimates. Then, albedos and geometry of voxels within a thin-shell are optimized.*

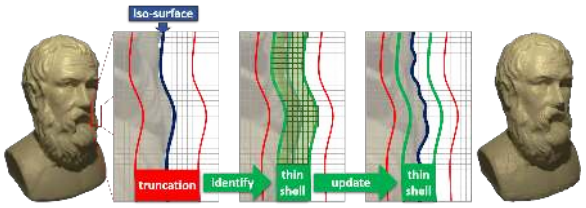

Figure 5: *Refinement overview: input model (left) and output (right). Memory for voxels is only allocated within the truncation region. Refinement is performed in the thin-shell region.*

which we compute on-the-fly from $\mathbf{C}(\mathbf{v})$. The reflected irradiance B at a voxel $\mathbf{v}$ on (or close to) $\mathbf{D}_0$ is given as

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

This is a parameterization of the reflectance equation, where $\mathbf{a}(\mathbf{v})$ is the albedo at $\mathbf{v}$, H_m are the spherical harmonics (SH) basis functions, and $\mathbf{l} = (l_1, ..., l_{b^2})$ are the corresponding spherical harmonics coefficients of the environment map; i.e., the incident illumination represented in the SH basis. We do not infer more than $b = 3$ (i.e., up to 2nd order) spherical harmonics bands from images of Lambertian surfaces [Hasinoff et al. 2011]. For efficiency reasons, we also do not consider visibility at each voxel. In order to refine a surface based on the shading constraint, we need to solve for the per-voxel albedo $\mathbf{a}$, per-scene lighting $\mathbf{l}$, and most importantly, the per-voxel normal $\mathbf{n}$, which is directly coupled to the underlying signed distance function (see next paragraph).

The spherical harmonics basis functions H_m are parametrized by a unit normal $\mathbf{n}$, and are defined as

$$H_0 = 1.0, H_1 = n_y, H_2 = n_z, H_3 = n_x, H_4 = n_x n_y,$$
$$H_5 = n_y n_z, H_6 = -n_x n_x - n_y n_y + 2 n_z n_z,$$
$$H_7 = n_z n_x, H_8 = n_x n_x - n_y n_y.$$

Normal Field On a signed distance field, surface normals $\mathbf{n} \in \mathbb{R}^3$ are given by the gradient operator. In our case, we express the normals by the gradient of the refined signed distance function $\tilde{\mathbf{D}}$, which for a continuous 3D location $\mathbf{p} = (x, y, z)$ is defined as:

$$\nabla\tilde{\mathbf{D}}(x,y,z) = \lim_{\delta \to 0} \frac{1}{\delta} \begin{pmatrix} \tilde{\mathbf{D}}(x+\delta, y, z) - \tilde{\mathbf{D}}(x,y,z) \\ \tilde{\mathbf{D}}(x, y+\delta, z) - \tilde{\mathbf{D}}(x,y,z) \\ \tilde{\mathbf{D}}(x, y, z+\delta) - \tilde{\mathbf{D}}(x,y,z) \end{pmatrix}.$$

The surface normal $\mathbf{n}(\mathbf{p}) = (n_x, n_y, n_z)$ is then given by

$$\mathbf{n}(\mathbf{p}) = \nabla\tilde{\mathbf{D}}(\mathbf{p}) / \|\nabla\tilde{\mathbf{D}}(\mathbf{p})\|_2.$$

In the case of a discrete voxel $\mathbf{v} = (i, j, k)$, we obtain the corresponding surface normal $\mathbf{n}(i, j, k)$ through $\nabla\tilde{\mathbf{D}}(i, j, k)$ with $\delta = 1$; i.e., a numerical forward difference between adjacent voxels. Note again that we initialize $\tilde{\mathbf{D}}$ with $\mathbf{D}$, and as the optimization commences, $\mathbf{n}$ changes along with the underlying $\tilde{\mathbf{D}}$.

5.1 Lighting Estimation with Signed Distance Fields

We estimate the illumination coefficients $\mathbf{l}$ by minimizing the differences between the computed shading B on the iso-surface $\mathbf{D}_0$ and the luminance data $\mathbf{I}$, which is computed from the captured color data $\mathbf{C}$:

$$E_{\text{light}}(\mathbf{l}) = \sum_{\mathbf{v} \in \mathbf{D}_0} (B(\mathbf{v}) - \mathbf{I}(\mathbf{v}))^2.$$

In practice, we consider voxels located inside a thin shell region t_{shell} – i.e., where $|\tilde{\mathbf{D}}| < t_{\text{shell}}$ – as one sample point (cf. Fig. 5). In our experiments, we set t_{shell} to twice of the size of a voxel. The minimization of $E_{\text{light}}(\mathbf{l})$ is performed using the current albedo and geometry estimate. In our case, $E_{\text{light}}(\mathbf{l})$ is a linear least-squares problem whose solution is equivalent to the following over-constrained linear system, where $\{\mathbf{v}_0, ..., \mathbf{v}_k\}$ is the set of voxels with $|\tilde{\mathbf{D}}| < t_{\text{shell}}$:

$$\underbrace{\begin{pmatrix} H_1(\mathbf{n}(\mathbf{v_0})) & \dots & H_{b^2}(\mathbf{n}(\mathbf{v_0})) \\ H_1(\mathbf{n}(\mathbf{v_1})) & \dots & H_{b^2}(\mathbf{n}(\mathbf{v_1})) \\ \vdots & \ddots & \vdots \\ H_1(\mathbf{n}(\mathbf{v_k})) & \dots & H_{b^2}(\mathbf{n}(\mathbf{v_k})) \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} l_1 \\ l_2 \\ \vdots \\ l_{b^2} \end{pmatrix}}_{\mathbf{l}} = \underbrace{\begin{pmatrix} \mathbf{I}(\mathbf{v}_0)/\mathbf{a}(\mathbf{v}_0) \\ \mathbf{I}(\mathbf{v}_1)/\mathbf{a}(\mathbf{v}_1) \\ \vdots \\ \mathbf{I}(\mathbf{v}_k)/\mathbf{a}(\mathbf{v}_k) \end{pmatrix}}_{\mathbf{b}}.$$

Note that the dimensionality of $\mathbf{A}$ is $|\{\mathbf{v}_0, ..., \mathbf{v}_k\}| \times b^2$, where $k >> b^2$. To obtain the least-squares solution, we use the normal equation $\mathbf{A}^T\mathbf{A} \cdot \mathbf{l} = \mathbf{A}^T\mathbf{b}$. Instead of storing $\mathbf{A}$ or $\mathbf{b}$ explicitly, we directly compute $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{b}$ using a parallel reduction on the GPU. As a result, we obtain $(\mathbf{A}^T\mathbf{A}) \cdot \mathbf{l} = (\mathbf{A}^T\mathbf{b})$, a $b^2 \times b^2$ linear system, which is solved using a singular value decomposition. As $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{b}$ are low-dimensional, the SVD solve (not the reduction, which runs on the GPU) is sufficiently fast on the CPU.

We assume distant illumination, which makes the lighting coefficients $\mathbf{l}$ spatially invariant; however, they are computed under the consideration of local albedo and surface variation. Before we start optimizing for $\tilde{\mathbf{D}}$ and $\mathbf{a}$ (see Section 6) on the coarsest level, we assume an initially uniform albedo to compute $\mathbf{l}$. Since we update the lighting with each hierarchy level (see Section 3), we use the optimized albedo distribution from the previous level for subsequent lighting estimations.

Unlike single image-based methods (e.g., [Wu et al. 2014]) which yield inconsistent lighting estimates for different views, our estimation of lighting on a volume provides a consistent and much more robust solution. Using the information of all fused color images in tandem also efficiently regularizes out noise prior to energy minimization. Directly considering all images simultaneously would make the optimization more complex. In contrast, we can exploit our efficient SDF representation for gaining higher efficiency.

5.2 Geometry Refinement and Albedo Estimation on Signed Distance Fields

The core innovation in our method is the shape-from-shading-based refinement of $\mathbf{D}$ to obtain $\tilde{\mathbf{D}}$. The goal is to refine $\mathbf{D}$ such that the

computed appearance B of the refined surface matches the high-resolution image data, under the estimated lighting $\mathbf{l}$. This involves solving a non-linear optimization problem to minimize a similarity measure. The problem is further compounded by the fact that albedos $\mathbf{a}(\mathbf{v})$ are needed to predict the appearance, but are also unknown. Previous shading-based refinement methods often approached this chicken-and-egg problem by assuming constant albedo, or by clustering a fixed set of discrete albedos before optimizing geometry. A better, yet more complex strategy, is to simultaneously optimize for unknown albedos and refined geometry. In order to solve this ill-conditioned inverse rendering problem, prior assumptions about materials are usually made, such as a discrete set of albedos [Wu et al. 2011], or a data-driven BRDF prior [Haber et al. 2009].

Our method solves for refined geometry $\tilde{\mathbf{D}}$ and unknown albedos $\mathbf{a}$ in a combined global optimization problem. However, we formulate this optimization in a fundamentally different manner from many previous shading-based refinement approaches that solve for surface displacements along normals of a mesh, or orientations of points or patches. Instead, we directly solve for distance values in a volumetric signed distance field such that the shading constraints are fulfilled. This implicit formulation enables fast hierarchical processing, efficient regularization, and avoids commitment to an explicit surface model with costly topology handling. It also differs by simultaneously solving for dense spatially-varying albedos under the control of a robust chromaticity-based regularizer.

The unknowns are optimized by solving the following non-linear least squares problem:

$$E_{\text{refine}}(\tilde{\mathbf{D}}, \mathbf{a}) = \sum_{\mathbf{v} \text{ s.t. } |\tilde{\mathbf{D}}(\mathbf{v})|<t_{\text{shell}}} w_g E_g(\mathbf{v}) + w_r E_r(\mathbf{v}) + w_s E_s(\mathbf{v}) + w_a E_a(\mathbf{v}),$$

where E_g is a shading gradient constraint, E_r is a volumetric regularizer, E_s is a surface stabilization constraint, and E_a is a constraint on albedos. w_g, w_r, w_s, w_a are corresponding optimization weights. We solve for the unknowns of the voxels in a thin shell band of width t_{shell} around the surface, where $|\tilde{\mathbf{D}}| < t_{\text{shell}}$, as depicted in Fig. 5.

Gradient-based Shading Constraint Our data term is based on the assumption that high-frequency shape detail leads to shading variations visible in the RGB images. For simplicity, we only consider grayscale input $\mathbf{I}$ (computed from $\mathbf{C}$), which should agree with the computed shading intensity B. At a voxel $\mathbf{v}$, E_g thus penalizes differences between gradients of captured grayscale input and the predicted intensity of our lighting model

$$E_g(\mathbf{v}) = \|\nabla B(\mathbf{v}) - \nabla \mathbf{I}(\mathbf{v})\|_2^2.$$

Note that the gradient difference metric is more robust than direct appearance difference against inaccuracies of our hypothesized shading model.

In order to evaluate E_g, we directly couple the gradients with the data of the relevant voxels. That is, ∇B is directly linked to the normals $\mathbf{n}$ and albedos $\mathbf{a}$. As $\mathbf{n}$ is not explicitly stored, our data term E_g directly constrains the underlying distance values. Note that the normals are defined with respect to the refined signed distance function $\tilde{\mathbf{D}}$.

As we optimize at discrete voxel locations (i, j, k), we approximate the gradients of the computed shading and observed image intensity using finite forward differences, similar to the computation of $\mathbf{n}$.

Volumetric Regularizer As shading-based refinement is generally an ill-posed problem and sensitive to noisy input, we need to regularize the refined signed distance function. To this end, we enforce a smoothness constraint at every voxel $\mathbf{v}$, which is defined as

$$E_r(\mathbf{v}) = (\Delta \tilde{\mathbf{D}}(\mathbf{v}))^2.$$

In practice, we discretize the volumetric Laplacian operator, considering direct voxel neighbors with uniform weighting.

Figure 6: *From left to right: fused model color, chromaticity, and estimated per voxel albedo luminance estimate* $\mathbf{a}(\mathbf{v})$.

Surface Stabilization We also define a surface stabilization constraint, penalizing the deviation of $\tilde{\mathbf{D}}$ from the original unrefined input distances $\mathbf{D}$:

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

This constraint is particularly important in the context of noisy input data, and mitigates the problem of getting stuck in local minima.

Albedo Regularizer The super-linear convergence of our Gauss-Newton solver (Section 6) enables efficient simultaneous optimization for both albedo and geometry. However, this may also introduce instabilities, as the decision of whether to change material or surface geometry is non-trivial. To reduce these ambiguities, we regularize albedo variations by introducing a consistency constraint based on chromaticity $\mathbf{\Gamma}$; see Fig 6. In spirit, this is similar to the regularization employed by Chen and Koltun [2013] to tackle the intrinsic image problem. The albedo regularizer at a voxel $\mathbf{v}$ considers its 1-ring neighborhood $\mathcal{N}_\mathbf{v}$ and is defined as

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

where the chromaticity $\mathbf{\Gamma}(\mathbf{v})$ is directly computed from $\mathbf{C}(\mathbf{v})$ as $\mathbf{\Gamma}(\mathbf{v}) = \mathbf{C}(\mathbf{v}) \,/\, \mathbf{I}(\mathbf{v})$, and $\phi(x)$ is the robust kernel introduced in Section 4.3. This regularization can be seen as a Laplacian smoothness constraint with anisotropic weights based on local chromaticity variations. This idea follows the assumption that surfaces with a similar chromaticity share a similar albedo. While this assumption does not hold everywhere in a scene, our experiments show that it is a good compromise between generality and resolving ambiguity.

6 Parallel Energy Optimization

Our objective $E_{\text{refine}}(\tilde{\mathbf{D}}, \mathbf{a})$ has a total of $2N$ unknowns, where N is the number of voxels (one unknown for $\tilde{\mathbf{D}}$ and one for $\mathbf{a}$ per voxel) inside of the thin shell region $|\tilde{\mathbf{D}}| < t_{\text{shell}}$. In our example scenes, N lies in the tens of millions of unknowns, leading to a high-dimensional non-linear optimization problem. In order to minimize the objective efficiently, we introduce a specifically tailored GPU-based Gauss-Newton solver that exploits the sparse structure of our data representation. Note that we assume the lighting is known here, as it is estimated at each hierarchy level before minimizing $E_{\text{refine}}(\tilde{\mathbf{D}}, \mathbf{a})$.

As our energy $E_{\text{refine}}(\tilde{\mathbf{D}}, \mathbf{a}) : \mathbb{R}^{2N} \rightarrow \mathbb{R}$ is a sum of squares, we

reformulate E_{refine} as a non-linear least-squares problem:

$$E_{\text{refine}}(\tilde{\mathbf{D}}, \mathbf{a}) = \sum_{k=1}^{M} f_k(\tilde{\mathbf{D}}, \mathbf{a})^2.$$

Then the number M of residual terms f_k is $M = 11N$. The number of residuals is composed of the following terms: E_g evaluates a difference of gradients $\rightarrow 3N$ terms; E_r is the Laplacian operator evaluated as a set of scalar values $\rightarrow N$ terms; E_s is a difference of scalar values $\rightarrow N$ terms; E_a is a sum iterating over the six direct neighbors of a voxel $\rightarrow 6N$ terms.

6.1 Parallel Gauss-Newton Optimization

For simplicity, we define the unknowns of E_{refine} as $\mathbf{x} = \{\tilde{\mathbf{D}}, \mathbf{a}\}$. Minimizing $E_{\text{refine}}(\mathbf{x})$ is a non-linear least-squares problem, which is reformulated in terms of its residual vector $\mathbf{F} : \mathbb{R}^{2N} \rightarrow \mathbb{R}^M$. This leads to the traditional Gauss-Newton definition

$$E_{\text{refine}}(\mathbf{x}) = \|\mathbf{F}(\mathbf{x})\|_2^2, \qquad \mathbf{F}(\mathbf{x}) = [f_1(\mathbf{x}), ..., f_M(\mathbf{x})]^T.$$

The optimal parameters $\mathbf{x}^*$ are obtained by solving the minimization problem

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{F}(\mathbf{x})\|_2^2.$$

To this end, we linearize the vector field $\mathbf{F}(\mathbf{x})$ around $\mathbf{x}_k$ using a first-order Taylor expansion to obtain an approximation of $\mathbf{F}(\mathbf{x}_{k+1})$:

$$\mathbf{F}(\mathbf{x}_{k+1}) \approx \mathbf{F}(\mathbf{x}_k) + \mathbf{J}(\mathbf{x}_k)\delta_k, \qquad \delta_k = \mathbf{x}_{k+1} - \mathbf{x}_k,$$

where $\mathbf{J}(\mathbf{x}_k)$ is the Jacobian matrix of $\mathbf{F}$ evaluated at $\mathbf{x}_k$. This approximation transforms the original non-linear optimization problem into a linear minimization problem:

$$\delta_k^* = \underset{\delta_k}{\operatorname{argmin}} ||\mathbf{F}(\mathbf{x}_k) + \mathbf{J}(\mathbf{x}_k)\delta_k||^2.$$

This is a highly over-constrained linear system for which we obtain the optimal least-squares solution δ_k^* by solving the corresponding normal equations:

$$\mathbf{J}(\mathbf{x}_k)^T\mathbf{J}(\mathbf{x}_k)\delta_k^* = -\mathbf{J}(\mathbf{x}_k)^T\mathbf{F}(\mathbf{x}_k).$$

To solve the original non-linear minimization problem, we thus need to solve a sequence of linearized problems; i.e., we initialize $\mathbf{x}_0$ with the scanned input $\mathbf{D}$ and uniform albedo, and successively compute the update δ_k^* from $\mathbf{x}_k$ to $\mathbf{x}_{k+1}$.

In order to solve for the linear update δ_k^*, we jointly optimize for all unknowns using a preconditioned conjugate gradient (PCG) solver. Similar to previous methods [Weber et al. 2013; Zollhöfer et al. 2014; Wu et al. 2014], we run the linear PCG steps using a set of CUDA kernels on the GPU.

6.1.1 Block-based PCG Solver

Note that we optimize on the sparse voxel hashing data structure (Section 4.1). Since our data structure is sparse and we want to refine around the iso-surface, we refine a voxel block if and only if it contains at least one voxel within the thin shell region ($|\tilde{\mathbf{D}}(\mathbf{v})| < t_{shell}$). To identify these blocks, we run a compactification on the spatial hashing data structure. The resulting linear index set is used for thread allocation. Note that the thin shell region is defined with respect to the refined surface, and is thus moving with the refined $\tilde{\mathbf{D}}$ in every Gauss-Newton step. In order to efficiently map the linear solve to the GPU, we have to exploit shared memory, which is an order of magnitude faster than global memory. To this end, we divide the entire voxel grid domain into a set of 4^3 voxel blocks, and use a variant of the *Schwarz Alternating Procedure*. Each voxel block and its corresponding 2-ring voxel boundary are loaded into shared memory and processed by a 64-thread CUDA block (i.e., $(2+4+2)^3 = 8^3$ voxels loaded per block). Note that the 2-ring voxel boundary is required due to the gradient and volumetric regularization constraints, E_g and E_r; the albedo constraint involves only a 1-ring neighborhood. We decide once per 5×5 voxel neighborhood if it can be optimized; i.e., we check if all voxels are allocated, have non-zero weight $\mathbf{W}(\mathbf{v}) > 0$, and provide a valid normal $||\mathbf{n}(\mathbf{v})||_2 \neq 0$. We treat the sub-domains (i.e., voxel blocks) as independent linear systems by imposing Neumann constraints at the boundaries. The sub-problems are solved using a data-parallel Preconditioned Conjugate Gradient (PCG) solver. After a sub-problem has been solved, we directly apply the obtained updates to the corresponding variables in global memory. Note that this leads to a mixture of *additive-* and *multiplicative-Schwarz*, since results of already finished blocks may be read by other blocks within the same Gauss-Newton step. We apply a virtual sub-block shift when loading data to shared memory. This shift is based on multiple Halton sequences with respect to the biases $2, 3, 5$ in the x-,y-, and z-directions, respectively. This moves the block boundaries around in space, allowing data to propagate faster, and thus improving convergence. In our examples, we run 10 local PCG iterations before we propagate updates to global memory. We monitor convergence of the Gauss-Newton solver by evaluating the non-linear residual error after each iteration step. If the change in residual error is smaller than a threshold, convergence is assumed.

6.1.2 Coarse-to-fine Optimization

The presented non-linear optimization strategy works well for estimating geometric detail at the given voxel resolution, since information can be easily propagated in a local neighborhood. Due to the sparse structure of the optimization problem and the used iterative optimization strategy, the information propagation distance is directly dependent on the number of iteration steps performed. Therefore, a high number of iterations would be required to propagate the voxel attributes over large spatial distances. To alleviate this problem, we use a coarse-to-fine nested optimization strategy (see Fig. 3) that leverages our sparse hierarchical voxel hashing data structure, thus effectively reducing the number of iterations and improving convergence. The objective is first minimized on the coarsest level; upon convergence, the optimized parameters $\tilde{\mathbf{D}}, \mathbf{a}$ are prolongated to the next finer level, where they serve as an initial estimate. We traverse the complete hierarchy in a coarse-to-fine fashion, from centimeter to sub-millimeter voxel resolution.

7 Results

We tested our volumetric shading-based refinement method on different reconstruction settings, primarily for RGB-D scanning but also with purely image-based reconstruction. In both cases, the cameras were moved by hand. In the case of RGB-D scanning, we capture 5 scenes using a PrimeSense Carmine 1.09 (Short Range) sensor. We use two modes of the sensor: 1) 640×480 depth and 640×480 RGB at 30 fps, and 2) 640×480 depth and 1280×1024 RGB at ≈ 12 fps. Note that this sensor uses a structured light pattern, only obtaining independent depth values at visible projected IR dot locations. Thus, the effective depth resolution is much lower than the depth stream resolution. Mode 1 is used for all sequences, except for the Augustus and the Relief. We used the weights $w_g = 0.2$, $w_r = 20 \rightarrow 160$, $w_s = 10 \rightarrow 120$, $w_a = 0.1$ for our test. Here, $a \rightarrow b$ means an increase of the weight from a to b during optimization. For objects with uniform albedo – i.e., the Augustus data set –, we use $w_a = \infty$ to keep the albedo constant.

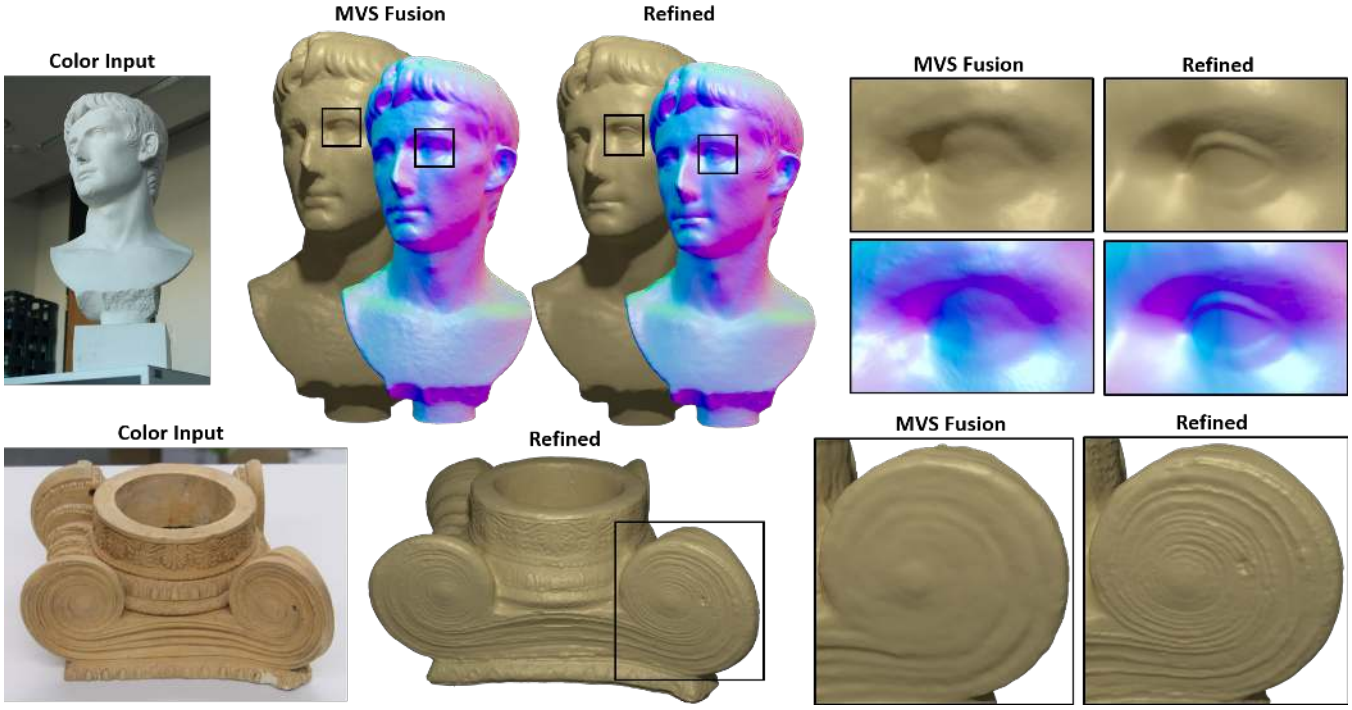

Figure 7: *Refinement results of our method using data obtained from multi-view stereo. In order to obtain the initial TSDF reconstruction, we input* 1139×1709 *pixel RGB images to a multi-view stereo reconstruction method [AgiSoft 2014].*

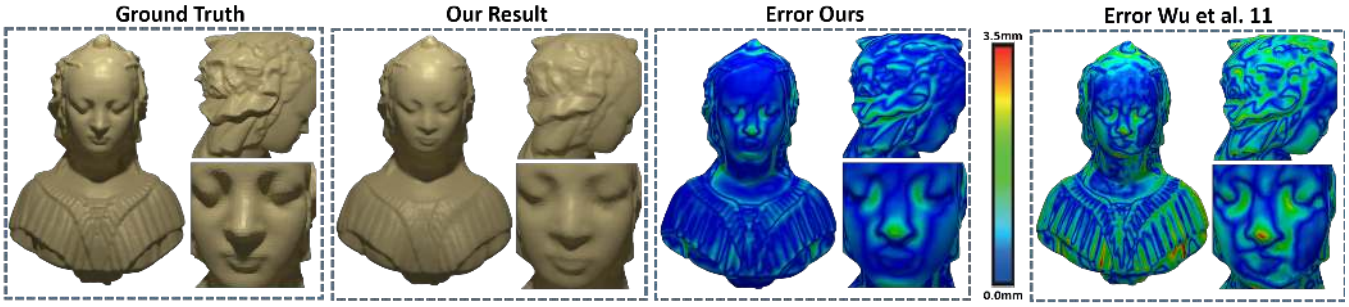

Figure 8: *Comparison between our approach and previous work [Wu et al. 2011] on synthetically-generated data, where the surface albedo is set to a uniform value.*

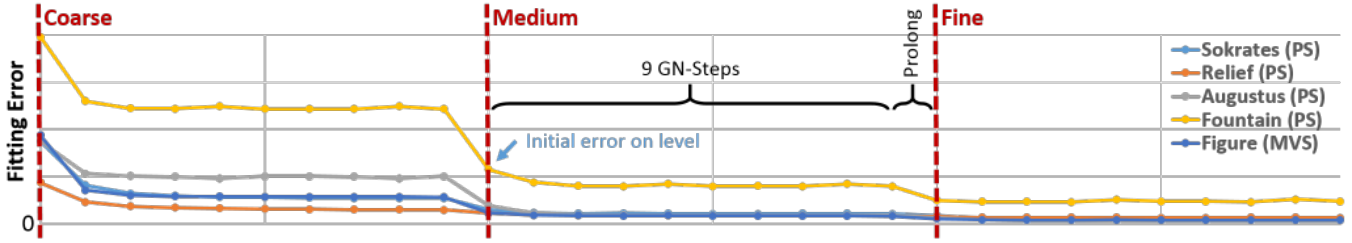

Figure 9: *Convergence analysis of our energy minimization using our Gauss-Newton solver for different scenes, where PS denotes the PrimeSense sensor, and MVS, multi-view stereo. We iterate over* 3 *hierarchy levels and run* 9 *Gauss-Newton steps at each level. Within a Gauss-Newton iteration,* 10 *PCG iterations minimize the linear system.*

First, we fuse noisy input depth data in order to obtain an initial shape represented as a TSDF. While fusion regularizes out noise, it also leads to severe over-smoothing, resulting in reconstructions which lack high-frequency geometric detail. We then run our new shading-based refinement approach directly on the implicit function to obtain the refined surface geometry along with a spatially-varying albedo estimate for each voxel. Fig. 1 and Fig. 12 show reconstruction results obtained with a PrimeSense Carmine 1.09 (Short Range) sensor before and after our refinement step. While real-time frame-to-frame tracking suffices for small scenes (i.e., the Augustus (PS) and Sokrates (PS)), we need to perform an offline pose optimization step (see Sec. 4.4) for larger scans. All scenes were captured under different uncontrolled lighting conditions. The number of variables that need to be optimized during refinement, and the typical amount of iterations used in the refinement optimization are listed in Table 1. Effectively, that means we show results with a 0.5mm voxel resolution on the finest level for the fountain scene, and all other scenes have a 1mm resolution. Note however, that the continuous signed distance field $\tilde{\mathbf{D}}$ provides a higher surface resolution than the underlying voxel grid. We compare a refined reconstruction with a ground truth laser scan in Fig 14. As we can see in the final reconstructions, we are able to capture a significant amount of surface detail absent from the unrefined result. For instance, note the fine strands in the hair of the Augustus bust, or the fine details and writing in the Relief. For both weakly and strongly textured objects, our reconstructions are of high accuracy.

In addition to RGB-D scanning, we also show the versatility of our method by using it in a purely image-based reconstruction context. We capture 4 scenes using a multi-view stereo setup, where we reconstruct an initial shape model using a state-of-the-art stereo reconstruction method [AgiSoft 2014]. For each result, 30 images are captured with a handheld commodity Canon EOS 1100D camera and downsampled to 1139×1709 pixels. From the multi-view stereo reconstruction, we obtain a initial TSDF and then run our volumetric shading-based refinement to obtain fine-scale geometric detail. Fig. 7 shows the reconstruction results before and after our refinement step. Note that we are able to show significant improvements, even on these high-quality multi-view stereo reconstructions.

Our method is very stable for different parameter settings; no scene-specific tweaking is needed. There are a few controllable parameters, such as the number of hierarchy levels, or the number of iterations of the optimizer. These parameters are determined by the difference in input depth and color resolutions; e.g., low-resolution depth and high-resolution color will require more hierarchy levels. Results are also stable on a range of the weights of E_{refine}; we use a coherent set for all our results. For input depth with strong noise (this is currently the norm for commodity RGB-D data), better results are obtained when using a higher weight for the volumetric regularizer E_r.

7.1 Evaluation

We quantitatively evaluate the accuracy of our method on synthetically-generated depth and color data (see Fig. 8). The data is generated by rendering a mesh from 28 virtual view points with a given camera trajectory. Through ray casting, we obtain a quantized depth map from the ground truth model, and add Gaussian noise to mimic a real depth sensor's characteristics. For color data generation, we set the surface reflectance to a uniform value and use the same spherical harmonics lighting model as in the method of [Ramamoorthi and Hanrahan 2001]. We run our algorithm on these simulated depth and RGB images, first obtaining an initial model by fusing the depth and then applying our shading-based refinement technique. The length of the bounding box diagonal of the mesh corresponds to 30cm, and the root-mean-square error is 0.47mm. The error plot on the surface is shown in Fig. 8.

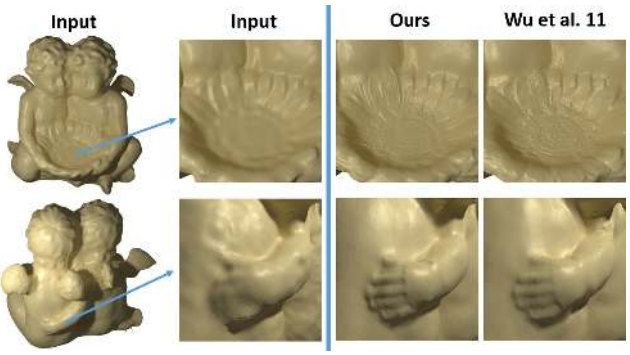

Figure 10: *Comparison between our algorithm and Wu et al. [2011] on a multi-view stereo sequence.*

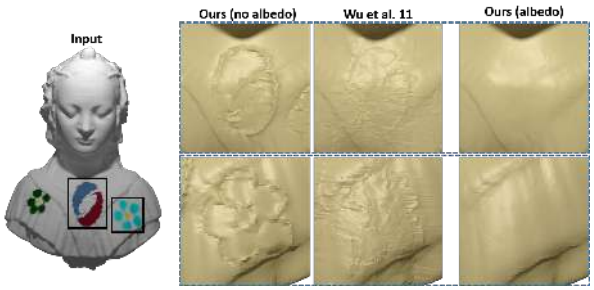

Figure 11: *Comparison between our approach and Wu et al. [2011] on synthetic data with varying surface albedo.*

In addition, we quantitatively compare our method with previous work, a shading-based refinement method which operates on meshes (MESHRef) [Wu et al. 2011]. As MESHRef requires an input mesh instead of a TSDF, we extract the initial mesh from the implicit function. In comparison to our method, MESHRef only obtained a reconstruction accuracy with root-mean-square error of 0.86mm. Our approach not only improves upon quality, but moreover, significantly improves computational efficiency. While our method requires less than 7.8 seconds, MESHRef takes 20 minutes to obtain the refined result. This speedup is largely due to the fact that minimization of an objective function on a mesh data structure is fundamentally harder than using our hierarchical and sparse grid structure. In Fig. 11, we provide another comparison on the same dataset, except that we now consider spatially-varying albedo from a painted texture. If we disable albedo optimization – i.e., assume a fixed uniform albedo –, we would misinterpret material transitions as shading cues and hallucinate wrong surface detail; previous work reduce such artifacts by assuming a set of clusters of albedos [Wu et al. 2011], but these artifacts are still very notable. Jointly optimizing for both geometric detail and albedo variation with our new chromaticity-based albedo regulizer further mitigates these artifacts and provides a much more realistic solution. We also compare our method with MESHRef on real-world data in Fig. 10. Our approach not only has a runtime advantage (6.5 seconds against about 1 hour), but also produces higher quality results. Fig. 13 shows a comparison to the method of [Wu et al. 2014] including a final fusion step. This final fusion smoothes out some of the previously reconstructed detail. In contrast, our approach works in the reverse order and obtains higher quality results (voxel resolution of 0.5mm in both cases). The cumulative runtime on the complete sequence is a few seconds for our approach as well as the method of [Wu et al. 2014].

7.2 Runtime and Convergence

We analyze the effectiveness of our method in Fig. 9, which shows convergence plots for optimizing our objective function. In this evaluation, we use 3 hierarchy levels and run 9 Gauss-Newton steps on

Seq.	Level 3			Level 2			Level 1			Level 0			Total	
	Fuse	Opt	#Vars	Fuse	Opt	#Vars	Fuse	Opt	#Vars	Fuse	Opt	#Vars	#Iter	Time
Sokrates (PS)	0.5s	85ms	200k	1.3s	0.1s	520k	1.6s	0.5s	2.0M	1.9s	3.9s	16M	10	9.9s
Relief (PS)	0.9s	0.6s	1.2M	1.3s	0.7s	2.5M	1.0s	1.4s	4.0M	1.2s	2.6s	12M	11	9.7s
Augustus (PS)	0.4s	0.1s	200k	1.8s	0.2s	1.5M	2.1s	1.2s	8.5M	2.4s	4.9s	26M	12	13.1s
Fountain (PS)	0.1s	0.1s	500k	0.2s	0.8s	2.5M	0.3s	1.1s	6.0M	0.5s	2.7s	19M	10	5.8s
Figure (MVS)	0.4s	0.8s	600k	1.4s	1.0s	2.7M	2.1s	2.1s	11M	1.9s	2.3s	16M	10	12s

Table 1: *Timing measurements for different test scenes, where PS denotes the PrimeSense sensor, and MVS, multi-view stereo.*

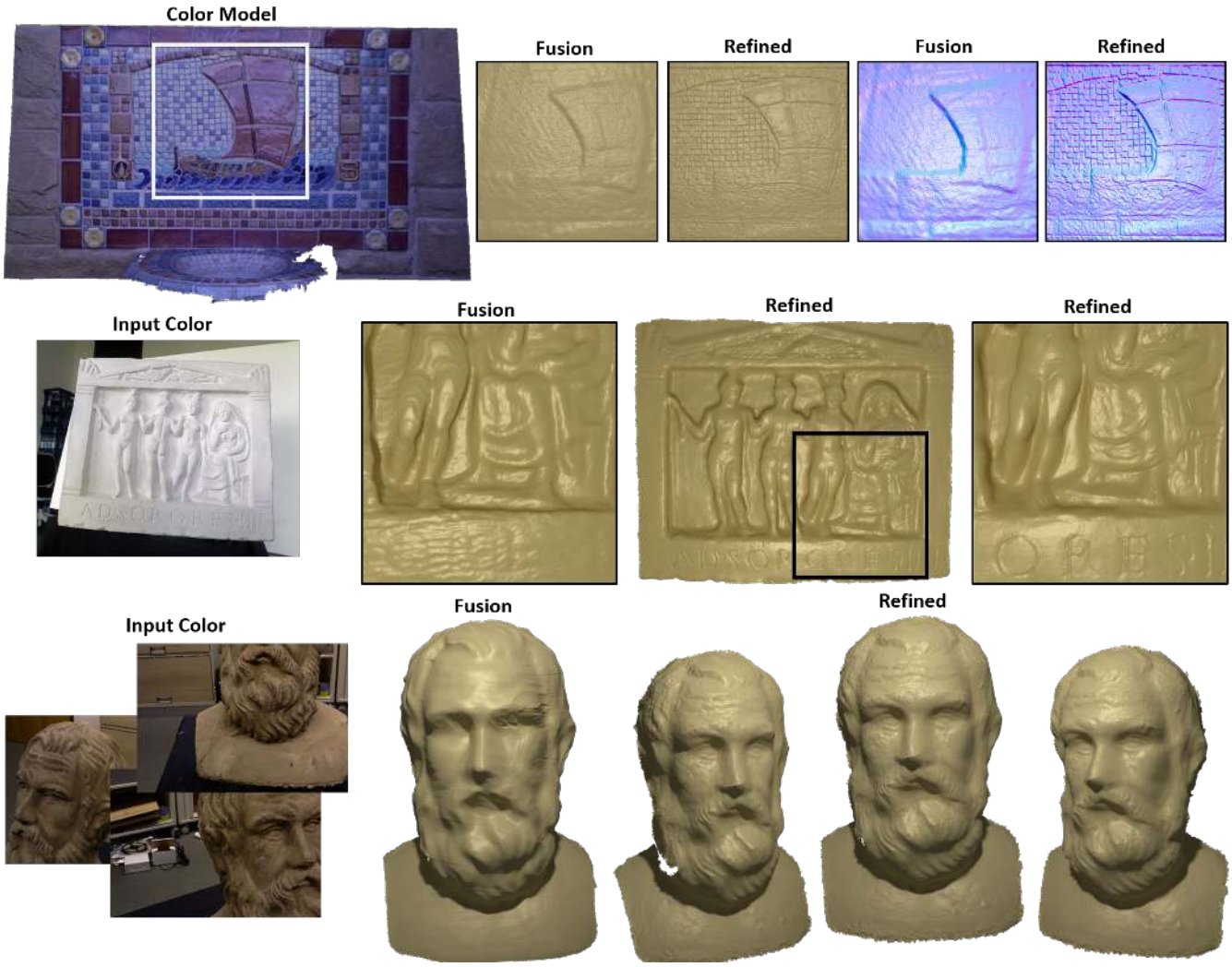

Figure 12: *Refinement results for different scenes captured with a PrimeSense Carmine 1.09 (Short Range) sensor.*

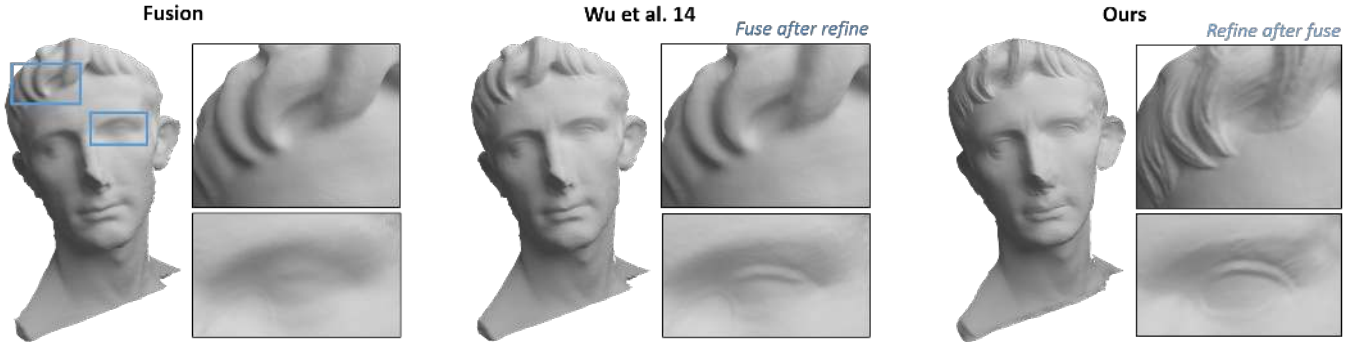

Figure 13: *The method of Wu et al. [2014] refines independent depth maps, causing detail to smooth out in the subsequent fusion process; i.e., fuse after refine. Our method refines the geometry after surface fusion, i.e., refine after fuse, thus capturing substantially more detail.*

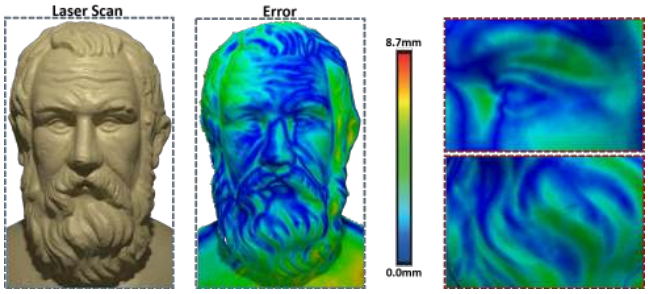

Figure 14: *Comparison with a laser scan: laser scan (left), error of our refined reconstruction (right) based on PrimeSense data.*

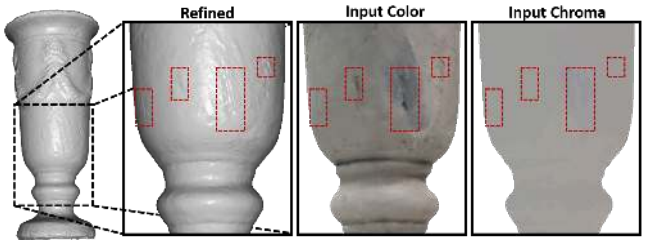

Figure 15: *Limitation of the albedo regularization strategy: spatially close albedo variations in monochromatic material lead to hallucinated geometry in the final reconstruction.*

each level of the hierarchy; within every non-linear Gauss-Newton iteration, we run 10 PCG steps. As our method enjoys super-linear convergence from jointly solving for all unknowns of our objective, the optimizer converges on every hierarchy level after only few iterations. At runtime, we detect convergence by monitoring the change in residual error after each Gauss-Newton iteration, thus allowing for an early out and keeping computational costs low.

The resulting timings are summarized in Table 1, measured on an Intel Core i7-3770 CPU with 3.4 GHz (16 GB Ram) and Nvidia Geforce GTX Titan GPU. The timings are generated using the following parameters: 4 hierarchy levels, a maximum of 9 outer Gauss-Newton iterations (early-out enabled), and 10 PCG iterations (coarse-to-fine). We measure timings for fusing input data (Fusion) and optimizing the TSDF (Opt) at each of the 4 hierarchy levels. In addition, we state the number of variables involved per level (#Vars), and provide the total number of Gauss-Newton steps over the complete hierarchy (#Iter). We also give the total compute time (Time). As shown, we are able to refine an implicit function with 26M free variables in only a few seconds. The highest resolution we used during our tests was 115M variables based on a 5 level hierarchy, which took about 25 seconds to compute (Fountain). Note, that this includes fusion and refinement of the complete input data.

8 Limitations

While we are able to generate compelling reconstruction results, our method still suffers from several limitations. Similar to previous shape-from-shading methods, our lighting model is an approximation to reality, as we employ spherical harmonics, assuming a distant, monochromatic light source. Second, we assume Lambertian surfaces, and cannot handle specularities, self-occlusions, spatially-varying illumination, or high-frequency textures. While our simple lighting model permits the reconstruction of a subset of materials, we hope that our volumetric formulation will support more complex models in the future. An equally hard problem is distinguishing between albedo and shading variation, particularly for high-frequency material. Our joint optimization for geometry and albedo is a first step towards such disambiguation, for materials with significant changes in chromaticity. However, for monochromatic material, our method might erroneously hallucinate geometric detail if the surface albedo changes abruptly (see Fig. 15). This error is a result of misinterpreting local albedo changes as variation in surface shading. A crucial requirement for shading-based refinement is the precise alignment of RGB-D input frames. Unfortunately, for larger scans, this requires several minutes in our current unoptimized implementation, limiting our method to run offline. If we were able to obtain accurate poses in real-time, our method would be well-suited to run in the background of a real-time scanning framework.

9 Conclusion

We have presented a new method for shading-based refinement directly on truncated signed distance fields to mitigate one of their significant disadvantages; i.e., over-smoothing. We leverage RGB data captured in general uncontrolled environments to refine the implicit surface by exploiting the shading cues. Our hierarchical and sparse TSDF representation easily allows for high spatial resolution, and facilitates efficient optimization, resulting in compute times over an order of magnitude faster than previous methods, while attaining more precision. While TSDFs are commonly used in real-time reconstruction methods, we still require an offline pose optimization step for larger scenes. However, we believe that future research along the lines of real-time bundle adjustment will close this gap. With this in mind, we hope that achieving highly-detailed reconstructions in real-time using consumer setups will become possible in the future.

A List of Mathematical Symbols

Symbol	Description
$\mathbf{v}$	position of voxel in $\mathbb{R}^3$
$\mathbf{p}$	continuous point in $\mathbb{R}^3$
$\mathbf{n}$	surface normal in $\mathbb{R}^3$
$\mathbf{D}(\mathbf{v})$	signed distance value at $\mathbf{v}$
$\mathbf{C}(\mathbf{v})$	color value (RGB) at $\mathbf{v}$
$\mathbf{W}(\mathbf{v})$	integration weight at $\mathbf{v}$
$\mathbf{a}(\mathbf{v})$	albedo at $\mathbf{v}$ (unknown variable)
$\tilde{\mathbf{D}}(\mathbf{v})$	refined TSDF (unknown variable)
$w_i(\mathbf{v})$	integration weight of frame i
$d_i(\mathbf{v})$	projective distance to voxel center in frame i
$\mathbf{D}_0$	iso-surface of the TSDF
π_c	projection of the color camera
π_d	projection of the depth camera
$\mathbf{I}(\mathbf{v})$	intensity at $\mathbf{v}$
$\mathbf{\Gamma}(\mathbf{v})$	chromaticity at $\mathbf{v}$
n	number of input frames
$\mathcal{D}_i$	depth image of frame i
$\mathcal{C}_i$	color image of frame i
$\mathcal{I}_i$	intensity of $\mathcal{C}_i$
$\mathcal{T}_i$	transformation from frame i to the base frame
q	hierarchy level
H_m	m-th spherical harmonics basis
$\mathbf{l}$	vector of all lighting coefficients l_m
$B(\mathbf{v})$	reflected irradiance at $\mathbf{v}$
N	number of voxels inside the thin shell region

Acknowledgements

We would like to thank Qian-Yi Zhou and Vladlen Koltun for the Fountain data. This work was funded by the Max Planck Center for Visual Computing and Communications, the German Research Foundation (DFG), grant GRK-1773 Heterogeneous Image Systems, and the ERC Starting Grant 335545 CapReal. We are grateful for the support of a Stanford Graduate Fellowship and gifts from NVIDIA Corporation.

References

AGARWAL, S., FURUKAWA, Y., SNAVELY, N., SIMON, I., CURLESS, B., SEITZ, S. M., AND SZELISKI, R. 2011. Building rome in a day. *Communications of the ACM 54*, 10, 105–112.

AGISOFT, L. 2014. Agisoft photoscan. *Professional Edition*.

BEELER, T., BRADLEY, D., ZIMMER, H., AND GROSS, M. 2012. Improved reconstruction of deforming surfaces by cancelling ambient occlusion. In *Proc. ECCV*, 30–43.

BYLOW, E., STURM, J., KERL, C., KAHL, F., AND CREMERS, D. 2013. Real-time camera tracking and 3d reconstruction using signed distance functions. In *Robotics: Science and Systems (RSS) Conference 2013*, vol. 9.

CARR, J. C., BEATSON, R. K., CHERRIE, J. B., MITCHELL, T. J., FRIGHT, W. R., MCCALLUM, B. C., AND EVANS, T. R. 2001. Reconstruction and representation of 3d objects with radial basis functions. In *Proc. SIGGRAPH*, ACM, 67–76.

CHAN, D., BUISMAN, H., THEOBALT, C., THRUN, S., ET AL. 2008. A noise-aware filter for real-time depth upsampling. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications-M2SFA2 2008*.

CHEN, Q., AND KOLTUN, V. 2013. A simple model for intrinsic image decomposition with depth cues. In *The IEEE International Conference on Computer Vision (ICCV)*.

CHEN, J., BAUTEMBACH, D., AND IZADI, S. 2013. Scalable real-time volumetric surface reconstruction. *ACM TOG 32*, 4, 113.

CUI, Y., SCHUON, S., CHAN, D., THRUN, S., AND THEOBALT, C. 2010. 3d shape scanning with a time-of-flight camera. In *Proc. CVPR*, 1173–1180.

CURLESS, B., AND LEVOY, M. 1996. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM, 303–312.

DEBEVEC, P. 2012. The light stages and their applications to photoreal digital actors. In *SIGGRAPH Asia Technical Briefs*.

DIEBEL, J., AND THRUN, S. 2006. An application of Markov Random Fields to range sensing. 291–298.

DOLSON, J., BAEK, J., PLAGEMANN, C., AND THRUN, S. 2010. Upsampling range data in dynamic environments. In *Proc. CVPR*, IEEE, 1141–1148.

FUHRMANN, S., AND GOESELE, M. 2014. Floating scale surface reconstruction. *ACM TOG 33*, 4, 46.

GHOSH, A., FYFFE, G., TUNWATTANAPONG, B., BUSCH, J., YU, X., AND DEBEVEC, P. 2011. Multiview face capture using polarized spherical gradient illumination. *ACM TOG 30*.

GOLDLUECKE, B., AUBRY, M., KOLEV, K., AND CREMERS, D. 2014. A super-resolution framework for high-accuracy multiview reconstruction. *ijcv 106*, 2 (jan), 172–191.

HABER, T., FUCHS, C., BEKAER, P., SEIDEL, H.-P., GOESELE, M., AND LENSCH, H. 2009. Relighting objects from image collections. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 627–634.

HAN, Y., LEE, J.-Y., AND KWEON, I. S. 2013. High quality shape from a single rgb-d image under uncalibrated natural illumination. In *Proc. ICCV*.

HASINOFF, S., LEVIN, A., GOODE, P., AND FREEMAN, W. 2011. Diffuse reflectance imaging with astronomical applications. In *Proc. ICCV*, 185–192.

HENRY, P., KRAININ, M., HERBST, E., REN, X., AND FOX, D. 2012. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *Int. J. Robotics Research 31* (apr), 647–663.

HERNÁNDEZ, C., VOGIATZIS, G., AND CIPOLLA, R. 2008. Multiview photometric stereo. *IEEE PAMI 30*, 3, 548–554.

HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. 1992. Surface reconstruction from unorganized points. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '92, 71–78.

HORN, B. K. 1975. Obtaining shape from shading information. *The psychology of computer vision*, 115–155.

IZADI, S., KIM, D., HILLIGES, O., MOLYNEAUX, D., NEWCOMBE, R., KOHLI, P., SHOTTON, J., HODGES, S., FREEMAN, D., DAVISON, A., ET AL. 2011. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proc. UIST*, ACM, 559–568.

KAZHDAN, M., BOLITHO, M., AND HOPPE, H. 2006. Poisson surface reconstruction. In *Proc. SGP*.

KEHL, W., NAVAB, N., AND ILIC, S. 2014. Coloured signed distance fields for full 3d object reconstruction. In *Proc. BMVC*.

KELLER, M., LEFLOCH, D., LAMBERS, M., IZADI, S., WEYRICH, T., AND KOLB, A. 2013. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *Proc. 3DV*, 1–8.

KOPF, J., COHEN, M. F., LISCHINSKI, D., AND UYTTENDAELE, M. 2007. Joint bilateral upsampling. *ACM TOG 26*, 3.

LEE, K. J., ZHAO, Q., TONG, X., GONG, M., IZADI, S., LEE, S. U., TAN, P., AND LIN, S. 2012. Estimation of intrinsic image sequences from image+depth video. In *Proc. ECCV*, 327–340.

LEVOY, M., PULLI, K., CURLESS, B., RUSINKIEWICZ, S., KOLLER, D., PEREIRA, L., GINZTON, M., ANDERSON, S., DAVIS, J., GINSBERG, J., SHADE, J., AND FULK, D. 2000. The digital michelangelo project: 3d scanning of large statues. In *Proc. SIGGRAPH*, 131–144.

LINDNER, M., KOLB, A., AND HARTMANN, K. 2007. Data-fusion of PMD-based distance-information and high-resolution RGB-images. In *Proc. ISSCS*, 121–124.

LOWE, D. G. 2004. Distinctive image features from scale-invariant keypoints. *IJCV 60*, 2, 91–110.

MULLIGAN, J., AND BROLLY, X. 2004. Surface determination by photometric ranging. In *Proc. CVPR Workshops*.

NAIR, R., RUHL, K., LENZEN, F., MEISTER, S., SCHÄFER, H., GARBE, C. S., EISEMANN, M., MAGNOR, M., AND KONDERMANN, D. 2013. A survey on time-of-flight stereo fusion. In *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*. Springer, 105–127.

NEHAB, D., RUSINKIEWICZ, S., DAVIS, J., AND RAMAMOORTHI, R. 2005. Efficiently combining positions and normals for precise 3D geometry. *Proc. SIGGRAPH 24*, 3.

NEWCOMBE, R. A., AND DAVISON, A. J. 2010. Live dense reconstruction with a single moving camera. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, IEEE, 1498–1505.

NEWCOMBE, R. A., DAVISON, A. J., IZADI, S., KOHLI, P., HILLIGES, O., SHOTTON, J., MOLYNEAUX, D., HODGES, S., KIM, D., AND FITZGIBBON, A. 2011. Kinectfusion: Real-time dense surface mapping and tracking. In *Proc. ISMAR*, IEEE, 127–136.

NIESSNER, M., ZOLLHÖFER, M., IZADI, S., AND STAMMINGER, M. 2013. Real-time 3d reconstruction at scale using voxel hashing. *ACM TOG 32*, 6, 169.

PARK, J., KIM, H., TAI, Y.-W., BROWN, M. S., AND KWEON, I.-S. 2011. High quality depth map upsampling for 3d-tof cameras. In *Proc. ICCV*, IEEE, 1623–1630.

PRADEEP, V., RHEMANN, C., IZADI, S., ZACH, C., BLEYER, M., AND BATHICHE, S. 2013. Monofusion: Real-time 3d reconstruction of small scenes with a single web camera. In *Proc. ISMAR*.

PRADOS, E., AND FAUGERAS, O. 2005. Shape from shading: a well-posed problem? In *Proc. CVPR*.

RAMAMOORTHI, R., AND HANRAHAN, P. 2001. A signal-processing framework for inverse rendering. In *Proc. SIGGRAPH*, ACM, 117–128.

RICHARDT, C., STOLL, C., DODGSON, N. A., SEIDEL, H.-P., AND THEOBALT, C. 2012. Coherent spatiotemporal filtering, upsampling and rendering of RGBZ videos. *CGF (Proceedings of Eurographics) 31*, 2 (May).

ROTH, H., AND VONA, M. 2012. Moving volume kinectfusion. In *BMVC*, 1–11.

RUSINKIEWICZ, S., HALL-HOLT, O., AND LEVOY, M. 2002. Real-time 3D model acquisition. *ACM TOG 21*, 3, 438–446.

SCHARSTEIN, D., HIRSCHMÜLLER, H., KITAJIMA, Y., KRATHWOHL, G., NEŠIĆ, N., WANG, X., AND WESTLING, P. 2014. High-resolution stereo datasets with subpixel-accurate ground truth. In *Pattern Recognition*. Springer, 31–42.

SCHUON, S., THEOBALT, C., DAVIS, J., AND THRUN, S. 2009. Lidarboost: Depth superresolution for tof 3d shape scanning. In *Proc. CVPR*, IEEE, 343–350.

SEITZ, S., CURLESS, B., DIEBEL, J., SCHARSTEIN, D., AND SZELISKI, R. 2006. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. CVPR*, vol. 1, 519–528.

SNAVELY, N., SEITZ, S. M., AND SZELISKI, R. 2006. Photo tourism: exploring photo collections in 3d. *ACM TOG 25*, 3, 835–846.

TRIGGS, B., MCLAUCHLAN, P. F., HARTLEY, R. I., AND FITZGIBBON, A. W. 2000. Bundle adjustment–a modern synthesis. In *Vision algorithms: theory and practice*. Springer, 298–372.

WEBER, D., BENDER, J., SCHNOES, M., STORK, A., AND FELLNER, D. 2013. Efficient gpu data structures and methods to solve sparse linear systems in dynamics applications. In *CGF*, vol. 32, Wiley Online Library, 16–26.

WEISE, T., WISMER, T., LEIBE, B., AND VAN GOOL, L. 2009. In-hand scanning with online loop closure. In *ICCV Workshops*, 1630–1637.

WHELAN, T., KAESS, M., FALLON, M., JOHANNSSON, H., LEONARD, J., AND MCDONALD, J. 2012. Kintinuous: Spatially extended kinectfusion.

WU, C., VARANASI, K., LIU, Y., SEIDEL, H.-P., AND THEOBALT, C. 2011. Shading-based dynamic shape refinement from multi-view video under general illumination. In *Proc. ICCV*, IEEE, 1108–1115.

WU, C., STOLL, C., VALGAERTS, L., AND THEOBALT, C. 2013. On-set performance capture of multiple actors with a stereo camera. *ACM TOG (Proc. SIGGRAPh Asia) 32*, 6, 161.

WU, C., ZOLLHÖFER, M., NIESSNER, M., STAMMINGER, M., IZADI, S., AND THEOBALT, C. 2014. Real-time shading-based refinement for consumer depth cameras. *ACM TOG (Proc. SIGGRAPH Asia) 33*, 6, 200:1–200:10.

YU, L.-F., YEUNG, S.-K., TAI, Y.-W., AND LIN, S. 2013. Shading-based shape refinement of rgb-d images. In *Proc. CVPR*.

ZHANG, Z., TSA, P.-S., CRYER, J. E., AND SHAH, M. 1999. Shape from shading: A survey. *IEEE PAMI 21*, 8, 690–706.

ZHOU, Q.-Y., AND KOLTUN, V. 2014. Color map optimization for 3d reconstruction with consumer depth cameras. *ACM Transactions on Graphics (TOG) 33*, 4, 155.

ZOLLHÖFER, M., NIESSNER, M., IZADI, S., REHMANN, C., ZACH, C., FISHER, M., WU, C., FITZGIBBON, A., LOOP, C., THEOBALT, C., ET AL. 2014. Real-time non-rigid reconstruction using an rgb-d camera. *ACM TOG (Proc. SIGGRAPH) 4*.