Real-time Shading-based Refinement for Consumer Depth Cameras

Chenglei Wu[1] Michael Zollhöfer[2] Matthias Nießner[3] Marc Stamminger[2] Shahram Izadi[4] Christian Theobalt[1]
[1]Max-Planck-Institute for Informatics [2]University of Erlangen-Nuremberg [3]Stanford University [4]Microsoft Research

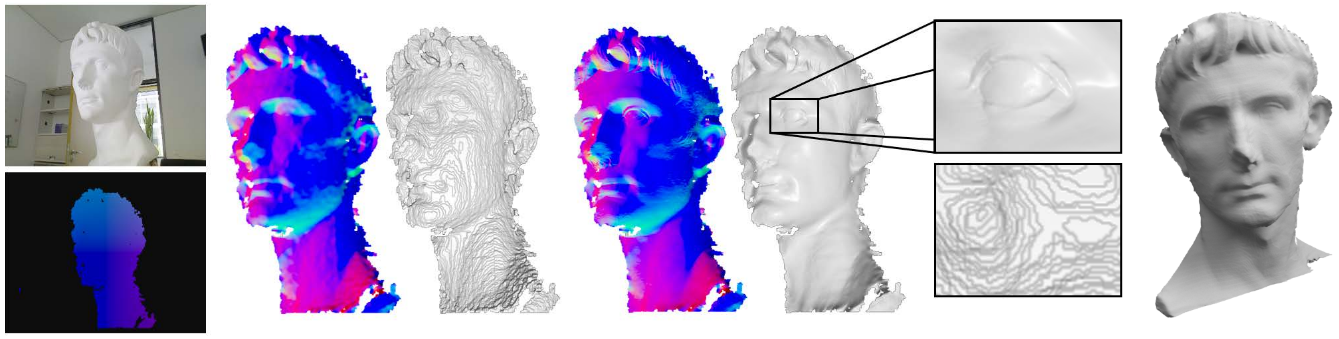


Figure 1: *Our method takes as input depth and aligned RGB images from any consumer depth camera (here a PrimeSense Carmine 1.09). Per-frame and in real-time we approximate the incident lighting and albedo, and use these for geometry refinement. From left: Example input depth and RGB image; raw depth input prior to refinement (rendered with normals and phong shading, respectively); our refined result, note detail on the eye (top right) compared to original depth map (bottom right); full 3D reconstruction using our refined depth maps in the real-time scan integration method of [Nießner et al. 2013] (far right)*

Abstract

We present the first real-time method for refinement of depth data using shape-from-shading in general uncontrolled scenes. Per frame, our real-time algorithm takes raw noisy depth data and an aligned RGB image as input, and approximates the time-varying incident lighting, which is then used for geometry refinement. This leads to dramatically enhanced depth maps at 30Hz. Our algorithm makes few scene assumptions, handling arbitrary scene objects even under motion. To enable this type of real-time depth map enhancement, we contribute a new highly parallel algorithm that reformulates the inverse rendering optimization problem in prior work, allowing us to estimate lighting and shape in a temporally coherent way at video frame-rates. Our optimization problem is minimized using a new regular grid Gauss-Newton solver implemented fully on the GPU. We demonstrate results showing enhanced depth maps, which are comparable to offline methods but are computed orders of magnitude faster, as well as baseline comparisons with online filtering-based methods. We conclude with applications of our higher quality depth maps for improved real-time surface reconstruction and performance capture.

CR Categories: I.3.7 [Computer Graphics]: Digitization and Image Capture—Applications I.4.8 [Image Processing and Computer Vision]: Scene Analysis—Range Data

Keywords: shading-based refinement, real-time, depth camera

Links: DL PDF WEB VIDEO CODE

1 Introduction

Consumer depth cameras have opened up many new real-time applications in the field of computer graphics and vision, robotics and human-computer interaction; including gestural interfaces, live 3D scanning, augmented reality, and robot navigation. However, the noise and resolution limitations of even recent depth cameras, result in only coarse geometry acquisition per frame. The ability to capture higher fidelity geometry in *real-time* could open up many new scenarios, such as tracking detailed features of the user (e.g., facial expressions, clothing etc.) for real-time performance capture or other interactive scenarios, as well as the ability to scan higher quality 3D models of real-world objects.

As shown previously, input from a stereo camera and shape-from-shading (SfS) can be used to capture detailed models with results approaching laser scan quality [Wu et al. 2011; Han et al. 2013; Yu et al. 2013; Beeler et al. 2010]. This raises the question: can this type of shading-based refinement be used to improve depth camera data, only by leveraging an additional RGB camera, which most sensors typically provide. Unfortunately, shading-based refinement techniques require information about the incident lighting and surface material in the scene. In most cases this requirement is fulfilled by making assumptions about albedo, and by working with controlled lighting [Hernández et al. 2008; Fanello et al. 2014], and studio setups [Ghosh et al. 2011; Debevec 2012; Bermano et al. 2014]. When moving to general uncontrolled scenes, SfS methods thus need to estimate albedo and illumination along with the geometry by solving a complex inverse rendering problem [Wu et al. 2011; Wu et al. 2013; Han et al. 2013; Yu et al. 2013]. So far, this was not possible in real time, and as such refinement techniques have yet to be used interactively.

Due to this performance bottleneck, researchers have developed alternative heuristic fusion strategies to enhance depth camera data in real time [Richardt et al. 2012]. Many of them use variants of joint bilateral upsampling [Kopf et al. 2007] to lift the depth data to the pixel grid resolution of a concurrently acquired and aligned RGB image. While computation is fast, the results are based on a purely heuristic assumption about the co-occurrence of discontinuities in RGB and depth data. In consequence, reconstructions may look plausible but estimated detail may not be metrically accurate. Further, the heuristic underpinning leads to commonly known artifacts, such as *texture copying*, where spatial albedo variations are mistaken for geometric detail.

In this paper, we propose a new real-time method for enhancement of depth data using SfS in general uncontrolled scenes. Starting from the raw depth data and an aligned RGB image, the algorithm estimates – in real time – the time-varying incident lighting distribution, which is then used to considerably enhance the reconstructed geometric detail. In contrast to previous fusion-based enhancement approaches, our reconstructions are not only plausible but more metrically faithful, and avoid some of the texture-copy artifacts seen previously.

In order to refine a depth map based on the shading in real-time, orders of magnitude faster than state-of-the-art offline systems [Wu et al. 2011], we must redesign the shading-based energy function as well as its optimization method. As such, we rephrase the shading-based refinement problem to fully exploit the regular connectivity of image grids. Instead of using an off-the-shelf conventional solver, we introduce a novel patch-based Gauss-Newton solver running on the GPU, which is specifically designed for our energy function. This careful design choice enables the refinement of depth maps in real-time, making it ideally suited to modern commodity range sensors that run at $\geq$ 30Hz. Specifically, our algorithm provides the following contributions:

- rephrasing the inverse rendering optimization problems used in offline methods [Wu et al. 2011] in a highly parallelized manner to enable real-time lighting estimation through spherical harmonics, and direct solving for refined depth rather than displacements on 3D meshes.
- space-time coherent estimation of shape and lighting using temporal correspondences derived from a real-time alignment of depth maps.
- an adaptive shape refinement strategy that reduces texture-copy artifacts by analyzing an approximate albedo image.
- a novel patch-based Gauss-Newton solver on the GPU to compute metrically faithful geometry at real-time frame-rates.

Beyond these technical contributions, we show the versatility of our method for reconstructing arbitrary scenes, even under motion, and demonstrate improved accuracy compared to filtering based refinement methods. We show integration into a real-time scanning framework akin to KinectFusion [Newcombe et al. 2011; Izadi et al. 2011; Nießner et al. 2013], and show improved quality during real-time capture. Finally, we demonstrate how our method enables improvement of the spatio-temporal reconstructions of a recent live non-rigid performance capture system [Zollhöfer et al. 2014a].

2 Related Work

Range Image Enhancement and Sensor Fusion Several methods to denoise and enhance depth data leverage the higher pixel resolution of one or two concurrently captured RGB images. Most of these methods rely on heuristic assumptions about the correlation of color and depth, e.g., that edges in both channels likely coincide.

Diebel and Thrun [2006] compute the upsampled depth using a Markov-Random Field. Park et al. [2011] formulate depth upsampling to color image resolution as an optimization problem enforcing the discontinuity similarity mentioned earlier, as well as additional regularization terms. Implementing the above heuristics through filtering is also feasible [Lindner et al. 2007], for instance by using joint bilateral upsampling [Kopf et al. 2007]. Yang et al [2007] create a cost space from the depth map, and filter it joint-bilaterally using a stereo image to raise resolution. Similar ideas have been explored for joint reconstruction using stereo images and depth data, where photometric constraints from stereo can be exploited for further data refinement [Beder et al. 2007; Zhu et al. 2008; Gudmundsson et al. 2008].

While the above methods run offline, variants of joint-bilateral or multilateral filtering for depth upsampling can run in real-time [Chan et al. 2008; Dolson et al. 2010; Richardt et al. 2012]. Their results, however, are merely plausible and not metrically accurate, and texture-copy artifacts frequently occur when texture variations are mistaken for geometric detail.

Multi-frame superresolution techniques estimate higher resolution depth images from a stack of aligned low resolution images captured under slight lateral displacement [Cui et al. 2013], but real-time computation has not been possible so far. One final set of methods increases the resolution of a single depth image offline using a learned database of local patches [Aodha et al. 2012].

Shape-from-Shading and Photometric Stereo A related topic acquires the 3D shape of an object using shape-from-shading (SfS) where the naturally occurring intensity patterns across an image are used to extract the 3D geometry from a single image [Horn 1975; Zhang et al. 1999]. The mathematics of SfS is well-understood, particularly when surface reflectance and light source positions are known. Prados and Faugeras [2005] and Fanello et al. [2014] reconstruct various objects including faces, using controlled light sources near the camera center. Ahmed and Farag [2007] demonstrate geometry estimation for non-Lambertian surfaces and varying illumination conditions, but make strong scene assumptions. Böhme et al. [2008] use the near infrared image available on time-of-flight (ToF) cameras to relate depth to intensity for filtering. However, unlike our method, their approach is limited to only ToF cameras with collocation of light source and camera, runs offline, and does not increase the X/Y resolution of images.

Recent methods have shown that SfS can refine coarse image-based shape models [Beeler et al. 2012], even if they were captured under general uncontrolled lighting with several cameras [Wu et al. 2011; Wu et al. 2013] or an RGB-D camera [Han et al. 2013; Yu et al. 2013]. To this end, illumination and albedo distributions, as well as refined geometry are found via inverse rendering optimization.

While the physics of SfS is well known, the problem is inherently ill-posed, and achieving compelling results requires strong scene and lighting assumptions, and computationally complex algorithms, particularly to solve hard inverse rendering optimizations. As such, real-time performance has rarely been demonstrated. This has led to work on *photometric stereo* where *multiple* images of a scene are captured under different controlled illumination to compute geometry. Photometric stereo has demonstrated compelling reconstructions of surfaces with complex reflectance properties [Mulligan and Brolly 2004; Hernández et al. 2008; Ghosh et al. 2011; Tunwattanapong et al. 2013; Debevec 2012; Bermano et al. 2014; Nehab et al. 2005]. However, these approaches require complex controlled lighting setups, which are not available in many standard scenarios.

More data-driven approaches for solving the SfS problem have also been proposed. Barron and Malik [2013b] jointly solve for reflectance, shape and illumination, based on priors derived statistically from images. Similar concepts were also used for offline intrinsic image decomposition of RGB-D data [Barron and Malik 2013a]. Zollhöfer et al. [2014b] use SfS to fit a morphable face model to an RGB input stream. Our approach does not impose strong priors on shape recovery. Khan et al. [2009] learn weighting parameters for complex SfS models to aid facial reconstruction.

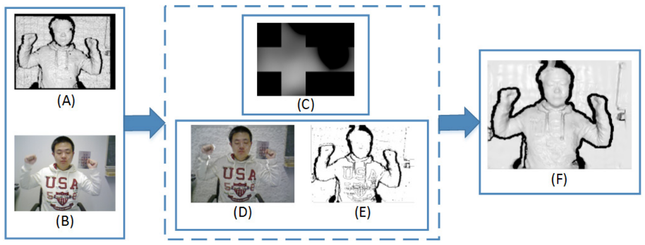


Figure 2: *Overview of our main pipeline. From left to right: Input to our algorithm is a noisy low resolution depth map (A) and an aligned RGB image (B). From these, an initial estimate of illumination is found (C), and subsequently an albedo image is computed (D), which is then used to detect an albedo edge map (E). Thereafter, the coarse geometry is refined using shading information (F). The high-dimensional non-linear optimization is solved using a new GPU-based iterative Gauss-Newton solver.*

Wei and Hirzinger [1996] use deep neural networks to learn aspects of the physical model for SfS, demonstrating moderate results for very constrained scenes. Again, none of these approaches achieves real-time performance.

3 Overview

In our work, we demonstrate *real-time* shading-based refinement of RGB-D data, captured in general scenes with unknown and time-varying lighting, using only commodity hardware. To achieve this goal, we reformulate the complex inverse problem for estimating illumination, albedo and refined geometry, which so far has only been solved offline, into a highly parallelized non-linear optimization problem, which we solve efficiently on the GPU using a new patch-based Gauss-Newton solver. We further employ new effective approximations and parameterizations, as well as fast geometric correspondence search on the GPU, which enables us to even enforce temporal priors in our reconstructions.

Input to our algorithm is a noisy low resolution depth map D^r from a depth camera and an aligned RGB image I. Unlike previous offline methods that used multi-camera input to refine full 3D meshes, we rephrase shading-based refinement as a depth map enhancement process. We solve the inverse rendering problem using an effective parameterization of the shading equation (Sect. 4). From the coarse depth and the RGB data, an initial estimate of illumination is found (Sect. 4.1), and subsequently an albedo image is computed. Thereafter, the coarse geometry is refined using shading information (Sect. 4.2). The high-dimensional non-linear optimization problem for depth refinement is solved using a new GPU-based iterative Gauss-Newton solver (Sect. 5). Fig. 2 highlights these main steps in the pipeline.

4 Shading-based Refinement of RGB-D Data

Real-time estimation of illumination and refined geometry necessitates an efficient formulation of the light transport model, i.e., the shading equation. Similar to previous offline methods, we assume that surfaces in a scene are Lambertian, and we parameterize the incident lighting with spherical harmonics (SH) [Wu et al. 2011]. In fact, we estimate incident irradiance as a function of the surface normal, that is the incident light, filtered by the cosine with the normal. For Lambertian reflectance, the incident irradiance function is known to be smooth, and can be represented with only little error using the first nine spherical harmonics basis functions up to 2nd order [Ramamoorthi and Hanrahan 2001].

As with previous approaches, we henceforth estimate lighting from a grayscale version of I, and thus assume gray lighting with equal values in each RGB channel. In some steps, full RGB images are used, which we denote I_c. Unlike offline multi-view methods, we employ a triangulated depth map as geometry parameterization. This means there is a fixed depth pixel to mesh vertex relation, and we can express the reflected irradiance $B(i, j)$ of a depth pixel (i, j) with normal $n(i, j)$ and albedo $k(i, j)$ as:

$$B(i,j) = k(i,j) \sum_{k=0}^{8} l_k H_k(\boldsymbol{n}(i,j)), \tag{1}$$

where l_k are the nine 2nd order spherical harmonics coefficients of the incident illumination. Note that in our real-time setting, we cannot afford local visibility computation, so illumination depends only on the normal direction.

The spherical harmonics basis functions $H_k(\boldsymbol{n})$ take a unit surface normal $\boldsymbol{n} = (n_x, n_y, n_z)$ as input, and evaluate to:

$$\begin{aligned} &H_0 = 1.0, H_1 = n_y, H_2 = n_z, H_3 = n_x, H_4 = n_x n_y, \\ &H_5 = n_y n_z, H_6 = -n_x n_x - n_y n_y + 2 n_z n_z, \\ &H_7 = n_z n_x, H_8 = n_x n_x - n_y n_y. \end{aligned} \tag{2}$$

Solving for geometry, lighting, and albedo from a single RGB-D image is highly underconstrained. During lighting estimation (Sect. 4.1), we therefore initially assume that the scene has uniform albedo. Subsequently, a dense albedo image is computed by dividing the RGB values through the lighting term. High-frequency detail in the depth map is then computed by shading-based refinement of the per-pixel depth values (Sect. 4.2). Unlike previous SfS methods that solve for surface normal orientations, we directly optimize the depth by linking the depth to the normal. This is not only computationally much more efficient, but also allows us to implicitly enforce surface integrability during depth optimization.

4.1 Lighting Estimation

The illumination coefficients l_k are computed by minimizing the difference between the rendered image B (given our current lighting estimate and geometry) and the captured RGB image I:

$$E_L(l) = \sum_{1\le i\le N_x, 1\le j\le N_y} \left(B(i,j) - I(i,j)\right)^2, \tag{3}$$

where (N_x, N_y) is the image size. Solving this least-squares problem is equivalent to solving the following system of linear equations:

$$\begin{pmatrix} H_0(\boldsymbol{n}(1,1)) & \dots & H_8(\boldsymbol{n}(1,1)) \\ H_0(\boldsymbol{n}(1,2)) & \dots & H_8(\boldsymbol{n}(1,2)) \\ \dots & \dots & \dots \\ H_0(\boldsymbol{n}(N_x,N_y)) & \dots & H_8(\boldsymbol{n}(N_x,N_y)) \end{pmatrix} \cdot l = A \cdot l = I. \tag{4}$$

The surface normals $n(i, j)$ are computed from the depth map after applying a Gaussian filter to remove noise. We exclude pixels at grazing angles for lighting estimation, as both shading and depth are unreliable in these regions. We detect these by checking if the angle between normal and viewing direction is greater than $78°$. For performance reasons, when the input RGB image resolution is higher than 640×480, we downsample the image by a factor of three in the lighting estimation stage.

The SH lighting coefficients are then obtained as $l = (A^T A)^{-1} A^T I$. For the calculation of $A^T A$ and $A^T I$, we use a parallel reduction and solve for the l_k on the CPU. In order to stabilize the lighting

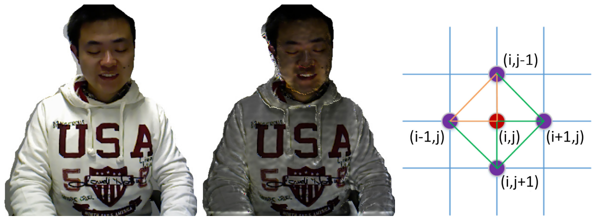


Figure 3: *Left: Albedo estimation. input image and estimated albedo map. Right: spatial neighborhood of geometric regularizer.*

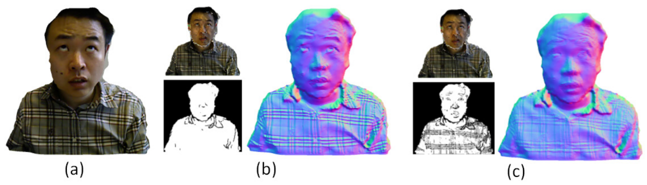


Figure 4: *Adaptive refinement helps to reduce texture-copy artifacts: the input frame (a) is refined and texture in the geometry of the shirt may lead to erroneous detail. Using a high (b) or low (c) threshold for albedo edge detection controls the reduction of this artifact.*

estimation, we optionally add a temporal prior term $\lambda_L(l - l^p)^2$ to Eq. (3), weighted by λ_L, which constrains the estimated lighting l to be similar to the lighting l^p in the previous frame. Then, the linear system we need to solve is as follows:

$$A^T A \cdot l + \lambda_L M_I = A^T \cdot I + \lambda_L l^p, \tag{5}$$

where $M_I \in \mathbb{R}^{9\times 9}$ is an identity matrix. An example illumination environment map corresponding to l is shown in Fig. 2. Given l, an estimate of a dense albedo image I_a with $I_a(i,j) = k(i,j)$ is computed on the GPU by dividing $I_c(i,j)$ by $\sum_{k=0}^{8} l_k H_k(\boldsymbol{n}(i,j))$, see Eq. (1). Example albedo images are shown in Fig. 3.

4.2 Shading-based Depth Map Refinement

Given the estimated lighting and albedo image, we refine the coarse depth through a second error minimization that uses shading cues from the intensity image. Previous methods for shading-based RGB-D refinement [Han et al. 2013] follow the traditional two-step SfS strategy, i.e., they first estimate the normal field, and then use it to refine the depth. Normal field computation for an image with N pixels requires optimizing an energy in $2N$ unknowns, and refining the depth based on the normal constraint means solving another sparse linear system with N variables. To achieve real-time performance we choose a more efficient strategy, and directly optimize for the depth value of each of the N pixels in I. This enables us to use the regular image structure for efficient parallelism of our optimization. Note that, depending on the camera, the physical depth resolution may be lower than the RGB resolution; we always sample depth and color at the same higher resolution.

To obtain the refined depth map D^*, we minimize:

$$E(D) = \sum_{(i,j)} w_g E_g(i,j) + w_s E_s(i,j) + w_p E_p(i,j) + w_r E_r(i,j), \tag{6}$$

where D is the vector of depth values. E_g is the shading gradient constraint, E_s is the smoothness constraint, E_p is the depth constraint, and E_r is a temporal smoothness prior. This is broken down into the following four terms:

Shading Gradient Constraint Our data term penalizes differences between rendered shading gradients and intensity image gradients:

$$\begin{aligned} E_g(i,j) = {} & [B(i,j) - B(i+1,j) - (I(i,j) - I(i+1,j))]^2 \\ & + [B(i,j) - B(i,j+1) - (I(i,j) - I(i,j+1))]^2, \end{aligned} \tag{7}$$

This gradient-based metric is more robust against inaccuracies of our approximate shading model which does not account for all lighting effects in a real scene. In order to evaluate the shading constraint w.r.t. $D(i,j)$, we first establish the link between $D(i,j)$ and $\boldsymbol{n}(i,j)$. The 3D position $\boldsymbol{p}(i,j)$ (in camera coordinates) of a depth point at distance $D(i,j)$ from the camera is:

$$\boldsymbol{p}(i,j) = \begin{pmatrix} (i-u_x)/f_x \\ (j-u_y)/f_y \\ 1 \end{pmatrix} D(i,j), \tag{8}$$

where (u_x, u_y) is the camera's principal point, and f_x and f_y are the focal lengths in x and y direction. The unnormalized surface normal at (i,j) can be computed from the 3D points of the neighboring depth pixels (Fig. 3):

$$\tilde{\boldsymbol{n}}(i,j) = (\boldsymbol{p}(i,j-1) - \boldsymbol{p}(i,j)) \times (\boldsymbol{p}(i-1,j) - \boldsymbol{p}(i,j)). \tag{9}$$

After substituting Eq. (8), this evaluates to:

$$\tilde{\boldsymbol{n}}(i,j) = \begin{pmatrix} \frac{D(i,j-1)(D(i,j)-D(i-1,j))}{f_y} \\ \frac{D(i-1,j)(D(i,j)-D(i,j-1))}{f_x} \\ \frac{\tilde{n}_x(i,j)(u_x-i)}{f_x} + \frac{\tilde{n}_y(i,j)(u_y-j)}{f_y} - \frac{D(i-1,j)D(i,j-1)}{f_x f_y} \end{pmatrix}. \tag{10}$$

Smoothness Constraint As shading-based refinement from a single image is ill-posed, we employ geometric regularization to constrain the solution. We enforce a Laplacian smoothness constraint for each pixel, which is computed as:

$$\begin{aligned} E_s(i,j) = {} & \|\boldsymbol{p}(i,j) - w_s(\boldsymbol{p}(i-1,j) + \boldsymbol{p}(i,j-1) + \\ & \boldsymbol{p}(i+1,j) + \boldsymbol{p}(i,j+1))\|_2^2, \end{aligned} \tag{11}$$

where $w_s = 0.25$ is the Graph Laplacian weight for the 1-ring neighborhood on the regular image triangle grid (Fig. 3), and $\boldsymbol{p}(i,j)$ is computed according to Eq. (8). Fig. 3 shows the neighborhood of this geometric regularizer.

Depth Constraint We also define a depth constraint, which enforces that the refined depth stays close to the initial depth before refinement D^i:

$$E_p(i,j) = (D(i,j) - D^i(i,j))^2, \tag{12}$$

Temporal Constraint To reduce temporal aliasing in our reconstructions, for static scenes we employ a temporal constraint to stabilize the refined depth. This uses the normals from the previous frame to constrain the depth in the current frame, and is defined as:

$$\begin{aligned} E_r(i,j) = {} & (\boldsymbol{n}^p(c(i,j)) \cdot (\boldsymbol{p}(i,j) - \boldsymbol{p}(i-1,j)))^2 \\ & + (\boldsymbol{n}^p(c(i,j)) \cdot (\boldsymbol{p}(i,j) - \boldsymbol{p}(i,j-1)))^2 \\ & + (\boldsymbol{n}^p(c(i,j)) \cdot (\boldsymbol{p}(i-1,j) - \boldsymbol{p}(i,j-1)))^2, \end{aligned} \tag{13}$$

where $\boldsymbol{n}^p$ is the refined normal in the previous frame, and $c(i,j)$ is the pixel in the previous frame corresponding to pixel (i,j) in the current frame. Unlike offline model-based reconstruction approaches, where pixel correspondences are implicitly given through a tracked template [Wu et al. 2013], our correspondences $c(i,j)$ are computed using a GPU-based iterative closest point (ICP) [Besl and McKay 1992] alignment between current and previous depth maps.

4.3 Adaptive Refinement

As our image formation model has not taken albedo variation into account, our method may interpret albedo changes as shading variation and produce artificial details around albedo boundaries. In order to reduce these texture-copy artifacts, we modified our shading energy term in Eq. (7) to be weighted by a binary mask, which decides if the corresponding image gradient comes from shading variation or albedo change. So the modified shading energy is defined as:

$$\begin{aligned} E_g(i,j) = w^r_{ij}[B(i,j) - B(i+1,j) - (I(i,j) - I(i+1,j))]^2 \\ + w^c_{ij}[B(i,j) - B(i,j+1) - (I(i,j) - I(i,j+1))]^2, \end{aligned} \tag{14}$$

where $w^r_{ij}, w^c_{ij} \in \{0,1\}$ are binary weights for each row and column, which are set to zero for albedo boundary edges. Albedo changes usually result in large difference in RGB color space [Horn 1974]. Therefore, we detect these by applying a user-defined threshold to an edge map computed on the albedo image I_a. Fig. 4 shows an example of how this strategy can reduce the texture-copy artifacts with varying thresholds. As the shading constraint is not reliable along silhouettes, we also search for depth discontinuities and set the corresponding weights to zero.

Solving the non-linear energy (Eq. 6) with its high number of unknowns in real-time is challenging. In the next section, we describe how to solve this optimization using a novel GPU-based Gauss-Newton solver that works on a patch subdivision in image space.

5 Parallel Energy Minimization

Our refinement energy $E(\mathbf{d}) : \mathbb{R}^N \rightarrow \mathbb{R}$ (Eq. 6) is non-linear given the image formation model and its dependence on the orientation of the surface normal. We use a row-major ordering of the pixels in the depth image D at the target resolution to obtain the parameter vector of the N unknown per-pixel depth values as follows:

$$\mathbf{d} = \left[\ldots, \quad D(i,j), \quad \ldots\right]^T. \tag{15}$$

Even at moderate resolutions, the objective has a considerable amount of parameters (i.e., $\approx 307k$ at a resolution of 640×480). To optimize a non-linear objective with such a high number of unknowns at real-time rates, we exploit the massively parallel architecture of modern GPUs. Minimizing E with respect to the unknown parameters is a non-linear least squares problem that can be rewritten as:

$$E(\mathbf{d}) = \sum_{k=1}^{M} r_k(\mathbf{d})^2. \tag{16}$$

The total number ($M = 9N$) of residual terms r_k depends on the shading gradient ($2N$ terms), depth (N terms), temporal ($3N$ terms) and smoothness constraints ($3N$ terms). The next sections describe our efficient parallel patch-based Gauss-Newton solver, that allows us to minimize this energy for more than 500,000 parameters at real-time rates.

5.1 Parallel Gauss-Newton Solver

We reformulate our objective E in terms of its residual vector $F : \mathbb{R}^N \rightarrow \mathbb{R}^M$ to obtain the classical Gauss-Newton form:

$$E(d) = ||F(\mathbf{d})||^2, \; F(\mathbf{d}) = \left[r_1(\mathbf{d}), \quad \ldots, \quad r_M(\mathbf{d})\right]^T. \tag{17}$$

Refined depth values $\mathbf{d}^*$ are then computed by minimizing:

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

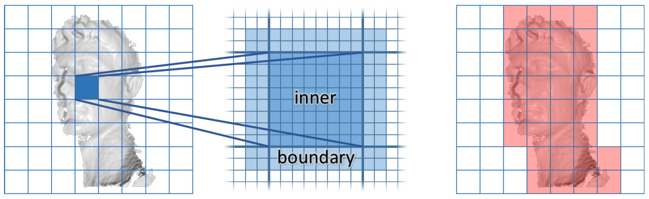
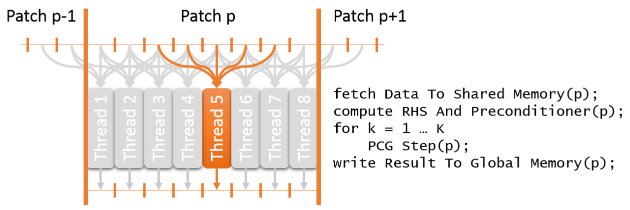


Figure 5: *To solve our problem efficiently, we subdivide our domain into patches (left). Each patch is optimized locally, which requires a two-pixel wide boundary (center). We can further optimize this procedure by only processing patches with foreground information (right).*



```
fetch Data To Shared Memory(p);
compute RHS And Preconditioner(p);
for k = 1 … K
    PCG Step(p);
write Result To Global Memory(p);
```

Figure 6: *For each patch p, a thread block is started with as many threads as the patch has pixels. First, all threads read patch data including a two-pixel size boundary to shared memory. Then, we perform multiple PCG steps within shared memory, and write the result back to global memory.*

Explicit linearization of the vector field $F(\mathbf{d})$ using Taylor expansion yields:

$$F(\mathbf{d}_{k+1}) \approx F(\mathbf{d}_k) + J(\mathbf{d}_k)\delta, \quad \delta = \mathbf{d}_{k+1} - \mathbf{d}_k. \tag{18}$$

$J(\mathbf{d}_k)$ is the Jacobian of F evaluated at the solution after k iterations. The resulting optimization problem is a linear least squares problem

$$\delta^* = \underset{\delta}{\operatorname{argmin}} \, ||F(\mathbf{d}_k) + J(\mathbf{d}_k)\delta||^2,$$

in the unknown optimal updates δ^*. We compute the δ^* as the solution of the corresponding normal equations:

$$J(\mathbf{d}_k)^T J(\mathbf{d}_k)\delta = -J(\mathbf{d}_k)^T F(\mathbf{d}_k).$$

These can be solved jointly on the complete domain using iterative solution techniques like preconditioned conjugate gradient (PCG). Previous work [Weber et al. 2013; Zollhöfer et al. 2014a] demonstrated the feasibility of this strategy in a GPU optimization framework for dynamics simulation and non-rigid registration, respectively. One important observation is that switching kernels has a significant impact on performance. The aforementioned methods are optimized such that they require 2 kernel calls for initialization and 3-4 kernel calls in the inner PCG loop, depending on whether the system matrix $J^T J$ is explicitly evaluated or by sequentially applying J^T and J. As a result, even for several thousands of variables, the optimization problem can be solved at interactive rates.

However, for our problem we should be able to optimize more than half a million values in real time, which is not possible with these approaches. To solve this, we develop an approach that can cope with the larger number of variables by exploiting the implicit topology of the depth mesh.

5.2 Patch-wise Optimization

With our error term, the computation for a single pixel only depends on a 5×5 image neighborhood. Thus, we can subdivide the domain into square patches (cp. Fig. 5), and perform the optimization patch-wise using a variant of the *Schwarz Alternating Procedure*. The optimization of a single patch happens per thread block, where all data can be kept in shared GPU memory. This procedure exploits the locality of the optimization constraints and the uniform tessellation of the optimization domain. It scales well to a higher number of unknowns and reduces kernel call overhead and global memory accesses by exploiting fast shared memory.

The optimization domain Ω (see Fig. 5) is partitioned into small rectangular sub-regions (patches)

$$\Omega = \bigcup_i \Omega_i, \quad \Omega_i \cap \Omega_j \neq \emptyset.$$

The linear systems corresponding to the sub-regions (without boundary) are solved independently by imposing Neumann constraints on the boundaries $\delta\Omega_i$. To be able to perform all computations in shared memory, including the shading gradient energy and the smoothness energy, we additionally have to read a two-pixel wide depth data for each patch boundary, so that the computation can be efficiently performed from local data. Optimization only happens on the inner variables; boundary values remain unchanged.

In each Schwarz iteration, inner, and boundary variables of a patch are first read and stored to shared memory. Then the inner variables are optimized, keeping the boundary values fixed. Finally, the inner variables are written back to global memory.

This decouples the patches and splits the set of parameters into unconstrained inner (d_i) and constrained boundary (d_b) variables:

$$\begin{bmatrix} A_{i,i} & A_{i,b} \\ A_{b,i} & A_{b,b} \end{bmatrix} \begin{bmatrix} d_i \\ d_b \end{bmatrix} = \begin{bmatrix} b_i \\ b_b \end{bmatrix}.$$

Since the boundary variables are considered to be fixed, the corresponding block entries can be moved to the right-hand side:

$$A_{i,i} d_i = b_i - A_{i,b} d_b.$$

Each local sub-problem on a sub-region (or patch) Ω_i is assigned to one thread block and solved in parallel using one thread per variable. The patch size is set based on the GPU L-1 cache; thus for our hardware setup, we use 16×16 patches. Including the boundary values, this results in a 20×20 grid that has to be kept in shared memory. The per-patch problem is solved using an iterative PCG solver, which is explained in the next section. The process is repeated N_e times or until convergence.

The entire algorithm is shown as pseudocode in Algorithm 1 and illustrated in Fig. 6. Note that Gauss-Newton and Schwarz iterations happen concurrently. After each Schwarz iteration, we also apply the delta updates, so each Schwarz iteration step implicitly performs a Gauss-Newton step. This does not incur any additional computation, because the PCG solver has to re-evaluate the Jacobian and the residuals anyway, but results in faster convergence.

Fig. 7 shows the convergence behavior depending on the number K of PCG steps. For the figure, we used $N_e = 20$ outer Gauss-Newton/Schwarz iterations, and $K = 1, 3, 5, 10$ inner PCG iterations. For $K > 10$ we observed no further improvement.

Note that we use no synchronization when writing depth values. As a result, some patches might already read updated boundary values, which leads to a mixture of *Multiplicative* and *Additive Schwarz*. In terms of convergence, this is not a problem; however, the approach becomes non-deterministic. On the other hand, avoiding synchronization improves performance. We shift the initial patch grid in each iteration by sub-patch steps based on a Halton sequence to improve convergence and to avoid patch structures from becoming visible in the solution.

Algorithm 1 Shared Memory PCG Kernel

```
for $i = 1 \dots N_e$ do
    for all patches $p$ in parallel do
        Fetch Data To Shared Memory(p);
        Compute RHS And Preconditioner(p);
        for $k = 1 \dots K$ do
            PCG Step(p);
        end for
        Write Result To Global Memory(p);
    end for
end for
```

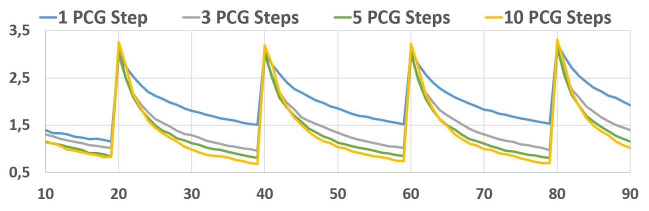


Figure 7: *Convergence of our optimization procedure for $N_e = 20$ and $K = 1, 3, 5, 10$ for four successive RGB-D images at 640 by 480 pixels. The abscissa shows the outer iterations.*

5.3 Patch-based Preconditioned Conjugate Gradient

Per patch, we solve the resulting linear optimization problem using a fast shared memory PCG solver. All per-patch PCGs corresponding to one Schwarz iteration are launched with a single kernel call. As can be seen in Algorithm 1, this includes shared memory initialization as well as running K PCG iterations and writing back the local patch results to global memory. In the PCG solver, we use a simple Jacobi preconditioner that can be readily parallelized. We exploit the memory hierarchy by caching all per-pixel data to registers and loading all data that has to be accessed by neighboring threads to shared memory. In each PCG iteration, a per-patch scalar product is required, for which we use a fast block reduction in shared memory. Excluding the block reductions, the inner PCG loop requires 6 synchronization points. The system matrix $J^T J$ is applied efficiently on-the-fly in each PCG step in an optimized kernel exploiting the sparsity of J.

5.4 Hierarchical Optimization Strategy

We run the proposed RGB-D shading-based refinement strategy in a hierarchical coarse-to-fine manner to allow for a faster convergence of our method. To this end, we build an image pyramid by successively restricting the input RGB-D data to the coarser levels. After, we sweep from coarse-to-fine (nested iteration) through the hierarchy and alternate between our patch-based Gauss-Newton solver and applying the prolongation operator. For prolongation and restriction, we use a bi-linear interpolation of the samples. Currently, we use a hierarchy with three levels.

5.5 Foreground Segmentation

Aside from the refinement of complete depth maps, we perform our optimization only on blocks containing foreground pixels (cp. Fig. 5). Based on the input depth, we mark all patches containing foreground pixels, compute a linear ordering of these using a fast prefix sum, and execute the refinement only on these foreground blocks.

Sequence	Camera	Resolution	Foreground	#Variables	Preprocess	Light Est.	Refinement	Σ
Augustus	PrimeSense	1280×1024	Yes	525k	7.0ms	2.5ms	26.4ms	35.9ms
Face	PrimeSense	1280×1024	Yes	245k	7.1ms	2.3ms	13.7ms	23.1ms
Body	PrimeSense	1280×1024	Yes	500k	6.9ms	2.4ms	25.6ms	34.9ms
Talking	Asus	640×480	No	307k	2.4ms	1.1ms	14.4ms	17.9ms
Vase	Asus	640×480	Yes	140k	2.6ms	1.0ms	7.2ms	10.8ms
Lucy	PrimeSense	640×480	No	307k	2.3ms	1.2ms	14.5ms	18.0ms
Flower	KinectOne	1920×1080	Yes	880k	19.5ms	3.7ms	43.2ms	66.4ms
Socrates	Asus	640×480	Yes	107k	2.5ms	0.9ms	6.2ms	9.6ms
Upper Body	PrimeSense	1280×1024	Yes	510k	6.8ms	2.5ms	25.8ms	35.0ms

Table 1: *Overview of test sequences, parameters and achieved performance - see Sec. 6 for details. Effective frame rates of our algorithm thus range from* 15 *fps at full HD to* 93 *fps at SVGA.*

6 Results

We tested our real-time enhancement software on data from a PrimeSense Carmine 1.09 Short Range (RGB res. 1280×1024, depth res 640×480, framerate 12 fps), a Kinect One (RGB res 1920×1080, depth res. 512×424, frame rate 30 fps), as well as an Asus Xtion Pro (RGB res. 640×480, depth res 640×480, framerate 30 fps) camera. Since video and RGB data are not frame synchronized in the Kinect One, the camera needs to be moved slowly in order to prevent artifacts. Our approach runs at real-time rates in excess of 30 fps. On both static and dynamic scenes and for all RGB-D sensors, a significant enhancement of detail compared to the raw depth data was achieved, see Fig. 12, Fig. 1, and the supplemental video and document, which show screen captured visualizations of the reconstructions before and after refinement.

In total, qualitative tests were done on 9 scenes, see Tab. 1. We always enable the prior term in lighting estimation by setting $\lambda_l = 10$. We set the empirically found weights for depth refinement as follows: $w_g = 1$, $w_s = 400$, $w_p = 10$, $w_r = 100$ for all static scenes; $w_g = 1$, $w_s = 100$, $w_p = 10$, $w_r = 0$ for all dynamic scenes. Please refer to Tab. 1 for details of the sequences and timings of the individual steps measured on an Intel Core i7-3770 CPU with 3.4GHz (16GB Ram) and an Nvidia Geforce GTX 780. The listed preprocessing steps include: depth-to-color alignment, filtering of depth and color, resampling of images, and foreground segmentation. For those results, the Gauss-Newton optimizer ran with the following parameters: 3 hierarchy levels, $N_e = 10, 8, 6$ outer iterations, and $K = 5, 5, 5$ PCG iterations (coarse-to-fine). We enable the temporal smoothness prior term for capturing static scenes, i.e., the Vase sequence, Lucy sequence and Socrates sequence. The required ICP alignment adds 3.5 ms for these three sequences to the total computation time. This yields effective frame rates between 15 fps at full HD and 93 fps at SVGA.

6.1 Evaluation

Quantitative Evaluation We quantitatively evaluate the accuracy of our method on two synthetic sequences that are 400 frames long. We use ground truth, detailed performance-captured face geometry [Valgaerts et al. 2012], and the ground truth lighting from St. Peter's Basilica [Debevec 1998] and render two RGB-D sequences. In the first one (CoA) the albedo is uniform, and in the second one (DA) we use a dense albedo map obtained from one of the captured face images. To synthesize the depth map sequences, we first obtain a quantized depth map from the stereo results of [Valgaerts et al. 2012], and then add Gaussian noise to mimic the noise from a depth sensor.

We compare our method with the space-time multi-lateral RGB-D filtering method of [Richardt et al. 2012] (STFilt), and with reconstructions of CoA and DA using the single-frame shading-based refinement algorithm in the offline method of [Valgaerts et al. 2012](SBRol). As an error metric, we employ the average pixel-wise Euclidean distance in mm per frame (d_e), as well as the average angular difference of normals in degrees (d_n). The distance and normal errors averaged over all frames are summarized in Tab. 2. Compared to STFilt, our method produces results with much lower distance and normal errors, as it obtains metrically faithful reconstructions as opposed to only plausible results (see Fig. 8). In comparison to the more involved offline method by [Valgaerts et al. 2012], our results exhibit comparable distance error, but our real-time capability comes at the price of a slightly higher error in reconstructed normal orientation. The respective error curves over time on both DA (Fig. 9) and CoA (additional material) further confirm the above conclusions.

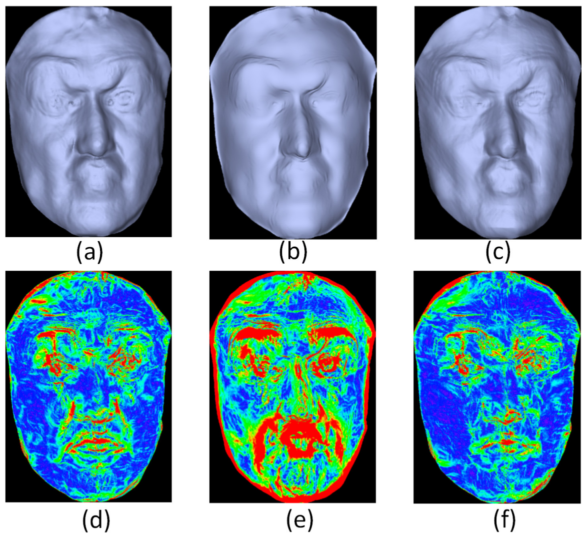


Figure 8: *Evaluation on one frame of synthetic CoA sequence : (a) our result, (b) results with STFilt [Richardt et al. 2012], (c) result with SBRol [Valgaerts et al. 2012]; (d) distance error heat map of our result (red=high), (e) of the much higher error of STFilt, and (f) the offline SBRol method with similar error to ours.*

Qualitative Comparison We also compared our method with STFilt on real-world data (talking sequence, see Tab. 1). Using the same hardware as previously described, our approach not only has a runtime advantage (55.8 fps against 8.5 fps), but also produces much more detailed results (see Fig. 10 and video).

Seq.	CoA		DA	
	d_e (σ)	d_n (σ)	d_e (σ)	d_n (σ)
Ours	0.43(0.24)	5.75(5.31)	0.43(0.23)	7.08(6.31)
STFilt	1.41(1.02)	10.32(11.22)	1.36(0.99)	10.28(10.65)
SBRol	0.43(0.27)	5.26(4.85)	0.42(0.22)	6.94(5.76)

Table 2: *Quantitative comparison against related methods. Our online approach performs significantly better in terms of distance d_e and normal error d_n (stddv. σ in brackets) than a competing state-of-the-art online method (STFilt [Richardt et al. 2012]), and even comes close to an offline shading-based refinement method (SBRol [Valgaerts et al. 2012]).*

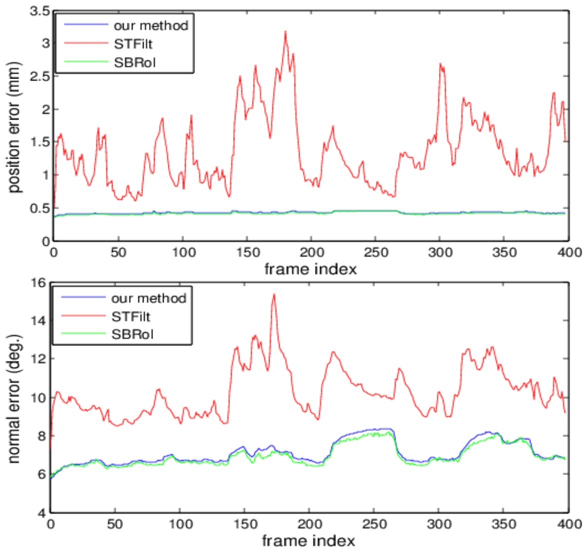


Figure 9: *Distance and normal error on synthetic sequence DA over all frames for our method, the RGB-D upsampling method STFilt [Richardt et al. 2012], and the offline approach SBRol [Valgaerts et al. 2012].*

6.2 Applications

Real-time 3D reconstruction We used our algorithm together with the real-time voxel-hashing-based hand-held scanning approach for depth cameras proposed by [Nießner et al. 2013]. With our drastically enhanced depth map quality, full 3D models with more detail can be reconstructed (voxel size of 0.5mm); see Fig. 1 and Fig. 11.

Deformable 3D reconstruction We also integrated our algorithm into the real-time deformable tracking approach by [Zollhöfer et al. 2014a], which leads to improved space-time coherent reconstructions of a non-rigidly deforming template, see Fig. 13.

6.3 Limitations

Our approach enables a leap forward in real-time scene reconstruction with depth cameras, but is still subject to several well-known shape from shading limitations. For instance, texture-copy artifacts are introduced by high-frequency albedo changes, causing problems for both online and offline methods [Richardt et al. 2012; Han et al. 2013; Yu et al. 2013]. Our adaptive refinement (Sect. 4.2) is efficient to mitigate the visual presence of these artifacts. However, we still cannot completely prevent them; for instance see Fig. 4 and Fig. 12 (top left). Generally, we believe that the underconstrained nature of this problem will inspire future research directions.

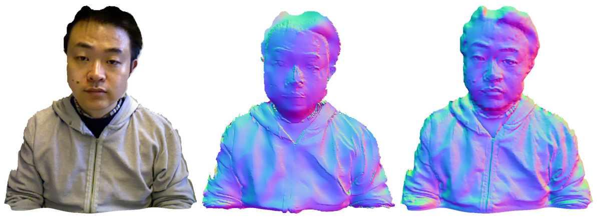


Figure 10: *On real-data, our approach (right) generates more detailed results at higher frame rates than STFilt [Richardt et al. 2012] (middle).*

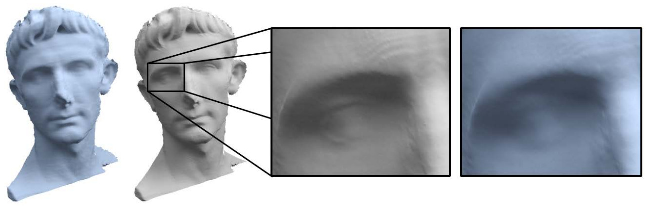


Figure 11: *Our enhanced depth maps used in the voxel-hashing-based framework of [Nießner et al. 2013]. Reconstruction with (white) and without (blue) refinement.*

In contrast to some offline methods, our real-time constraint allows only for a simplified light transport model. That is, our initial constant albedo assumption may exacerbate texture copying on general scenes; however, our results show that in practice, very faithful surface reconstructions with spatially-varying albedo are feasible. Due to the second order spherical harmonics representation, non-diffuse surfaces are still challenging for our method. In addition, we are not able to improve depth maps around silhouettes since the normal is undefined. We further assume a one-bounce local illumination and ignore lighting visibility, which may lead to errors in some cases. For example, hard shadows may result in artificial detail around their boundaries. An interesting future direction would be the incorporation of a screen-space ambient occlusion term to account for local visibility.

7 Conclusion

We presented the first method for real-time shading-based refinement of RGB-D data captured with commodity depth cameras in general uncontrolled scenes. This is enabled by a new real-time inverse rendering framework that approximates time-varying incident lighting as well as albedo in the scene. The algorithm then refines the raw depth of the camera by optimizing a complex non-linear energy using a new highly parallel Gauss-Newton solver on the GPU. The results are superior to previous online depth map enhancement algorithms, and on par with offline shape-from-shading approaches. Our experiments further show that the approach enables a new level of accuracy in handheld 3D scanning as well as deformable surface tracking.

Acknowledgements

This research was co-funded by the German Research Foundation (DFG), grant GRK-1773 Heterogeneous Image Systems, the ERC Starting Grant 335545 CapReal, and the Max Planck Center for Visual Computing and Communication. We would also like to thank Angela Dai for the video voice over.

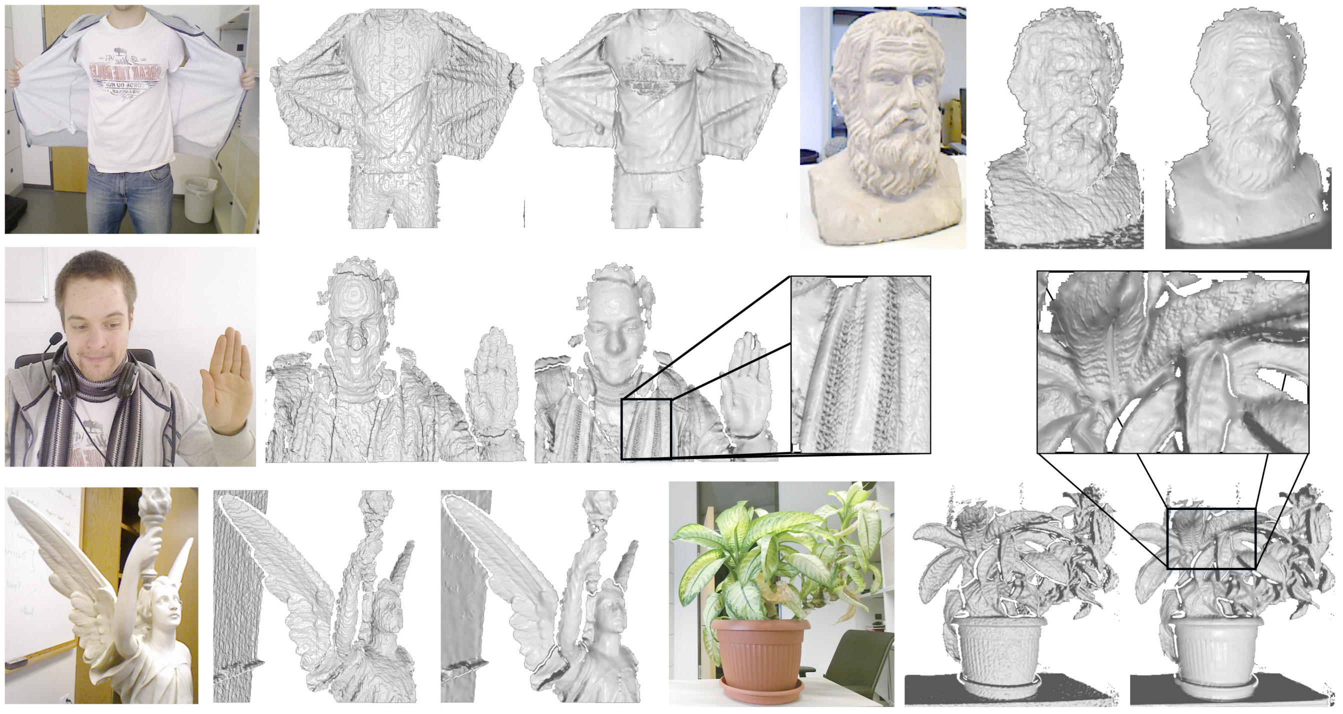


Figure 12: *From a raw depth map and an aligned RGB image, our approach generates rich details on real-world data. Our method captures far more details than the raw depth map on static scenes, i.e., Socrates sequence, Lucy sequence and Flower sequence. Besides, our per-frame refinement method can be readily applied to dynamic scenes, e.g. human performances, as shown on Body sequence and Upper Body sequence. The closeup of the scarf region demonstrates the amount of small-scale detail captured by our method.*

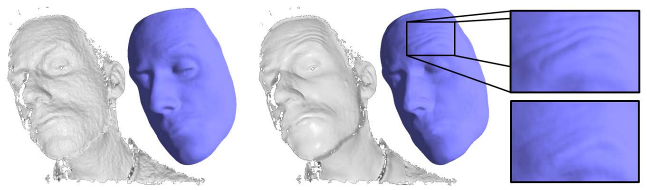


Figure 13: *Our enhanced depth maps (middle, gray) used in the non-rigid tracking framework of [Zollhöfer et al. 2014a]. Note the increased amount of small-scale detail in the reconstruction (middle, blue).*

References

AHMED, A. H., AND FARAG, A. A. 2007. Shape from shading under various imaging conditions. In *Proc. CVPR*, 1–8.

AODHA, O. M., CAMPBELL, N. D. F., NAIR, A., AND BROSTOW, G. J. 2012. Patch based synthesis for single depth image super-resolution. In *Proc. ECCV*, 71–84.

BARRON, J. T., AND MALIK, J. 2013. Intrinsic scene properties from a single rgb-d image. In *Proc. CVPR*, IEEE, 17–24.

BARRON, J. T., AND MALIK, J. 2013. Shape, illumination, and reflectance from shading. Tech. rep., EECS, UC Berkeley, May.

BEDER, C., BARTCZAK, B., AND KOCH, R. 2007. A combined approach for estimating patchlets from PMD depth images and stereo intensity images. In *Proc. DAGM*, 11–20.

BEELER, T., BICKEL, B., BEARDSLEY, P., SUMNER, B., AND GROSS, M. 2010. High-quality single-shot capture of facial geometry. *Proc. SIGGRAPH 29*, 3.

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

BERMANO, A., BRADLEY, D., ZUND, T. B. F., NOWROUZEZAHRAI, D., BARAN, I., SORKINE-HORNUNG, O., PFISTER, H., SUMNER, R., BICKEL, B., AND GROSS, M. 2014. Facial performance enhancement using dynamic shape space analysis. *ACM Transactions on Graphics 33*. to appear.

BESL, P. J., AND MCKAY, N. D. 1992. Method for registration of 3-d shapes. In *Robotics-DL tentative*, International Society for Optics and Photonics, 586–606.

BÖHME, M., HAKER, M., MARTINETZ, T., AND BARTH, E. 2008. Shading constraint improves accuracy of time-of-flight measurements. In *Proc. CVPR Workshop*.

CHAN, D., BUISMAN, H., THEOBALT, C., AND THRUN, S. 2008. A noise-aware filter for real-time depth upsampling. In *ECCV Workshop on multi-camera & multi-modal sensor fusion*.

CUI, Y., SCHUON, S., THRUN, S., STRICKER, D., AND THEOBALT, C. 2013. Algorithms for 3d shape scanning with a depth camera. *IEEE Trans. PAMI 35*, 5, 1039–1050.

DEBEVEC, P. 1998. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proc. SIGGRAPH*, 189–198.

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. In *Proc. NIPS*, 291–298.

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

FANELLO, S., KESKIN, C., IZADI, S., KOHLI, P., ET AL. 2014. Learning to be a depth camera for close-range human capture and interaction. *ACM Trans. Graph. 33*, 4.

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

GUDMUNDSSON, S. A., AANAES, H., AND LARSEN, R. 2008. Fusion of stereo vision and time-of-flight imaging for improved 3d estimation. *Int. J. Intell. Syst. Technol. Appl. 5*, 425–433.

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*.

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

HORN, B. K. 1974. Determining lightness from an image. *Computer graphics and image processing 3*, 4, 277–299.

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., AND FITZGIBBON, A. 2011. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proc. UIST*, ACM, 559–568.

KHAN, N., TRAN, L., AND TAPPEN, M. 2009. Training many-parameter shape-from-shading models using a surface database. In *Proc. ICCV Workshop*.

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

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.

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

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., IZADI, S., ET AL. 2011. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), IEEE international symposium on*, 127–136.

NIESSNER, M., ZOLLHÖFER, M., IZADI, S., AND STAMMINGER, M. 2013. Real-time 3d reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (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 *ICCV*, IEEE, 1623–1630.

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*, 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. *Computer Graphics Forum (Proceedings of Eurographics) 31*, 2 (May).

TUNWATTANAPONG, B., FYFFE, G., GRAHAM, P., BUSCH, J., YU, X., GHOSH, A., AND DEBEVEC, P. 2013. Acquiring reflectance and shape from continuous spherical harmonic illumination. *ACM Transactions on Graphics (TOG) 32*, 4, 109.

VALGAERTS, L., WU, C., BRUHN, A., SEIDEL, H.-P., AND THEOBALT, C. 2012. Lightweight binocular facial performance capture under uncontrolled lighting. *ACM Trans. Graph. 31*, 6.

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. *Computer Graphics Forum 32*, 1, 16–26.

WEI, G.-Q., AND HIRZINGER, G. 1996. Learning shape from shading by a multilayer network. *IEEE Trans. Neural Networks*.

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*.

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

YANG, Q., YANG, R., DAVIS, J., AND NISTR, D. 2007. Spatial-depth super resolution for range images. In *Proc. CVPR*, IEEE.

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.

ZHU, J., WANG, L., YANG, R., AND DAVIS, J. 2008. Fusion of time-of-flight depth and stereo for high accuracy depth maps. In *Proc. CVPR*.

ZOLLHÖFER, M., NIESSNER, M., IZADI, S., REHMANN, C., ZACH, C., FISHER, M., WU, C., FITZGIBBON, A., LOOP, C., THEOBALT, C., AND STAMMINGER, M. 2014. Real-time non-rigid reconstruction using an rgb-d camera. *ACM TOG (Proc. SIGGRAPH) 33*, 4.

ZOLLHÖFER, M., THIES, J., COLAIANNI, M., STAMMINGER, M., AND GREINER, G. 2014. Interactive model-based reconstruction of the human head using an rgb-d sensor. *Computer Animation and Virtual Worlds 25*, 3-4, 213–222.