3DLite: Towards Commodity 3D Scanning for Content Creation

JINGWEI HUANG, Stanford University
ANGELA DAI, Stanford University and Technical University of Munich
LEONIDAS GUIBAS, Stanford University
MATTHIAS NIESSNER, Technical University of Munich

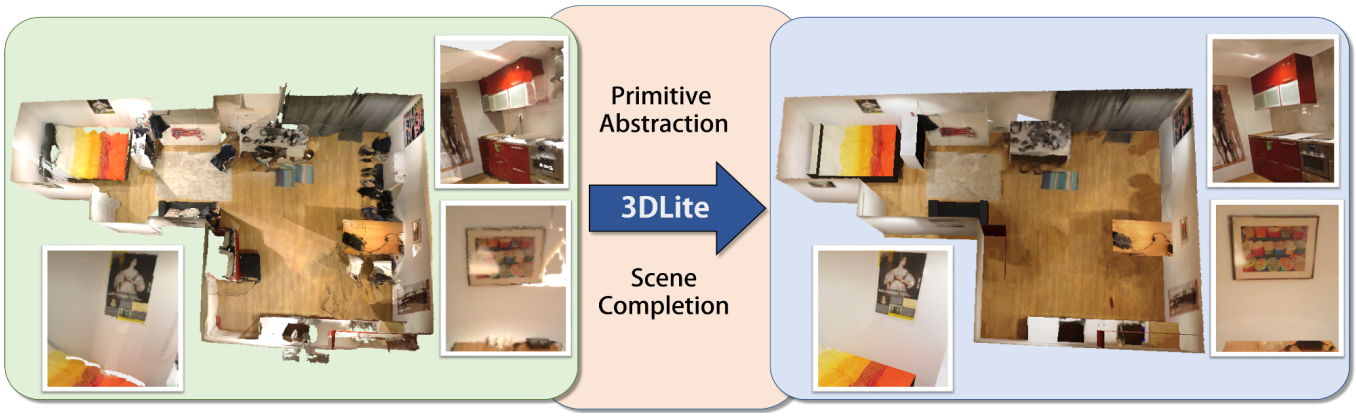


Fig. 1. 3DLite takes as input a 3D scene reconstruction captured with a commodity RGB-D sensor. The initial reconstruction on the left is obtained with a state-of-the-art volumetric fusion approach. Due to the nature of these methods, the 3D model contains many holes, noisy geometry, and blurry surface textures. 3DLite first computes a primitive abstraction of a scene, on top of which we formulate an optimization problem to generate sharp surface textures, combined from different input RGB frames. Based on this abstraction, we complete the high-level scene geometry, and use image inpainting to synthesize color unobserved regions. In the end, we obtain lightweight, low-polygon reconstructions that we believe are a step towards leveraging RGB-D scanning for content creation pipelines. Note the complete room geometry and the sharp surface textures on the right.

We present 3DLite[1], a novel approach to reconstruct 3D environments using consumer RGB-D sensors, making a step towards directly utilizing captured 3D content in graphics applications, such as video games, VR, or AR. Rather than reconstructing an accurate one-to-one representation of the real world, our method computes a lightweight, low-polygonal geometric abstraction of the scanned geometry. We argue that for many graphics applications it is much more important to obtain high-quality surface textures rather than highly-detailed geometry. To this end, we compensate for motion blur, auto-exposure artifacts, and micro-misalignments in camera poses by warping and stitching image fragments from low-quality RGB input data to achieve high-resolution, sharp surface textures. In addition to the observed regions of a scene, we extrapolate the scene geometry, as well as the mapped surface textures, to obtain a complete 3D model of the environment. We show that a simple planar abstraction of the scene geometry is ideally suited for this completion task, enabling 3DLite to produce complete, lightweight, and visually compelling 3D scene models. We believe that these CAD-like reconstructions are an important step towards leveraging RGB-D scanning in actual content creation pipelines.

[1]3DLite is available under http://graphics.stanford.edu/projects/3dlite/

0730-0301/2017/11-ART203 $15.00
https://doi.org/10.1145/3130800.3130824

CCS Concepts: • **Computing methodologies** → **Computer graphics**; *Shape modeling*; Mesh geometry models;

Additional Key Words and Phrases: RGB-D, scan, texture mapping

ACM Reference format:
Jingwei Huang, Angela Dai, Leonidas Guibas, and Matthias Nießner. 2017. 3DLite: Towards Commodity 3D Scanning for Content Creation. *ACM Trans. Graph.* 36, 6, Article 203 (November 2017), 14 pages.
https://doi.org/10.1145/3130800.3130824

1 INTRODUCTION

RGB-D scanning has made rapid advances in recent years with the introduction of commodity range sensors, such as the Microsoft Kinect, Intel RealSense, or Google Tango. State-of-the-art online and offline 3D reconstruction methods now allow remarkable capture and digitization of a variety of real-world environments, with faithful geometric fidelity [Chen et al. 2013; Choi et al. 2015; Dai et al. 2017b; Izadi et al. 2011; Newcombe et al. 2011; Nießner et al. 2013]. Although the intended applications of these methods cover a variety of gaming, virtual reality, and augmented reality scenarios, the quality of the resulting 3D models remains far from the caliber of artist-modeled content. In particular, current reconstructions still suffer from noise, oversmoothing, and holes, rendering them inadequate for use in production applications.

In many graphics applications, we observe that the surface textures are more important to visual perception than geometry; for

instance, many video games make use of techniques such as billboarding or bump mapping [Decoret et al. 1999] to achieve high-detail visuals at low cost, with almost imperceptible difference to using accurate geometry. Unfortunately, the color quality of existing 3D reconstruction methods often suffers from artifacts due to motion blur and rolling shutter from commodity color cameras (see Fig. 19(a)). This is compounded by oversmoothing and camera pose micro-misalignments due to popular reconstruction techniques. For instance, the seminal volumetric fusion work [Curless and Levoy 1996] is commonly used to generate a 3D model from input RGB-D frames by performing a weighted average over projected depth and color observations. While effectively regularizing out noise, this also results in oversmoothed geometry and color. Additionally, since camera poses are computed from imperfect color and noisy, relatively low-quality depth, they often suffer from micro-drift, which further exacerbates resulting visual artifacts such as ghosting and oversmoothing. Moreover, for effective use in gaming, VR, or AR applications, reconstructed environments must not have holes, which are always present in real-world scans, due to occlusions. Thus, our goal is to employ a relatively simple geometric representation of a scene to facilitate both high-quality texture mapping as well as surface- and texture-filling in occluded regions.

In this paper, we present 3DLite, an approach to generate lightweight, complete, CAD-like models with high-quality textures of large-scale indoor scenes. We take as input an RGB-D video sequence from a handheld commodity sensor, and first reconstruct the scene with existing 3D reconstruction methods. Since we aim to generate complete scenes and sharp, clean textures, we employ a primitive-based abstraction to represent the scanned environments. In particular, we use plane primitives, as planes facilitate texture mapping as well as scene completion through extrapolation, thus generating a denoised geometric representation of the scene. We first optimize for these primitives under a Manhattan assumption, since man-made environments are often designed in such highly structured fashion. In order to complete the scene geometry in occluded regions, we formulate a new hole-filling approach by extrapolating the planar primitives according to unknown space as seen by the camera trajectory. That is, we respect the known empty space in the scene and only fill holes in regions unseen by the camera. This generates a complete geometric representation of the scene. We then perform a novel texture optimization to map the scene geometry with sharp colors from the input RGB data. To generate complete textures for the entire scene, we follow this with a texture completion in unseen, hole-filled regions. Since camera poses estimated from noisy RGB-D data are prone to micro-drift, our texture optimization step solves for refined rigid and non-rigid image alignments, optimizing for photo-consistency using both sparse color and dense geometric information. To mitigate the effects of motion blur and auto-exposure, we perform an exposure correction, and select and stitch together only the sharpest regions of the input RGB images, obtaining globally consistent, sharp colors. Finally, for regions which lack color due to occlusions in the input scan, we inpaint the texture using color information from similar textures in the scene.

3DLite is a fully-automated, end-to-end framework designed to produce 3D models with high-quality textures mapped onto clean, complete, lightweight geometry, from consumer-grade RGB-D sensor data. We demonstrate its efficacy on a variety of real-world scenes, showing that we can achieve visually compelling results even with a simplified geometric representation.

In summary, our main contribution is a fully-automated 3D reconstruction system featuring

- a lightweight geometric abstraction,
- high-quality surface textures,
- and complete 3D representations w.r.t. both geometry and color.

2 RELATED WORK

3D reconstruction has been extensively studied over the last several decades. In this section, we discuss popular state-of-the-art approaches to generating 3D models of scanned environments, as well as the use of planar priors to guide reconstruction, and color map optimization on a geometric model.

3D reconstruction. Among both state-of-the-art online and offline 3D reconstruction methods, the most common approach to computing surface colors is through a moving average of the corresponding colors of the input images. Volumetric fusion [Curless and Levoy 1996], widely used by many state-of-the-art online and offline reconstruction methods [Chen et al. 2013; Choi et al. 2015; Dai et al. 2017b; Izadi et al. 2011; Newcombe et al. 2011; Nießner et al. 2013] is a very efficient approach to generating an implicit surface representation of a scene and effectively regularizing out noise input sensor data; unfortunately, it also leads to strong oversmoothing in both geometry and color, since depth and color measurements are fused projectively through a weighted average from varying views. Similarly, surfel-based reconstruction approaches, which generate a point representation of a scene, also tend to compute surface points and colors through a moving average [Keller et al. 2013; Whelan et al. 2015]. Oversmoothing from these averaging schemes is often further compounded by small errors in camera estimation (as poses are computed from noisy and blurry depth and color).

Planar impostors in image-based rendering. The idea of approximating distant geometry through images called *impostors* was widely used in the image-based rendering literature [Decoret et al. 1999; Sillion et al. 1997]. These images are in fact textured planes optimally positioned so as to generate approximately correct views from a range of viewpoints. Image-based rendering has also been recently used in custom renderers with a geometry proxy, and achieved high-quality results using sparse DSLR input images [Hedman et al. 2016].

Indoor and outdoor 3D reconstruction using planar priors. Since man-made environments are typically constructed in a highly structured fashion, with an abundance or orthogonal and parallel planes, a Manhattan plane assumption can be exploited to facilitate tracking in indoor scenes, particularly in the case of RGB-D scanning, as depth information is directly available. Many methods have been developed to incorporate planar information to improve camera tracking in 3D scanning scenarios. Dou et al. [2012] and Taguchi et al. [2013] both incorporate various plane correspondences with feature point correspondences to improve tracking robustness. Zhang et al. [2015] detect both planar structures and repeated objects to

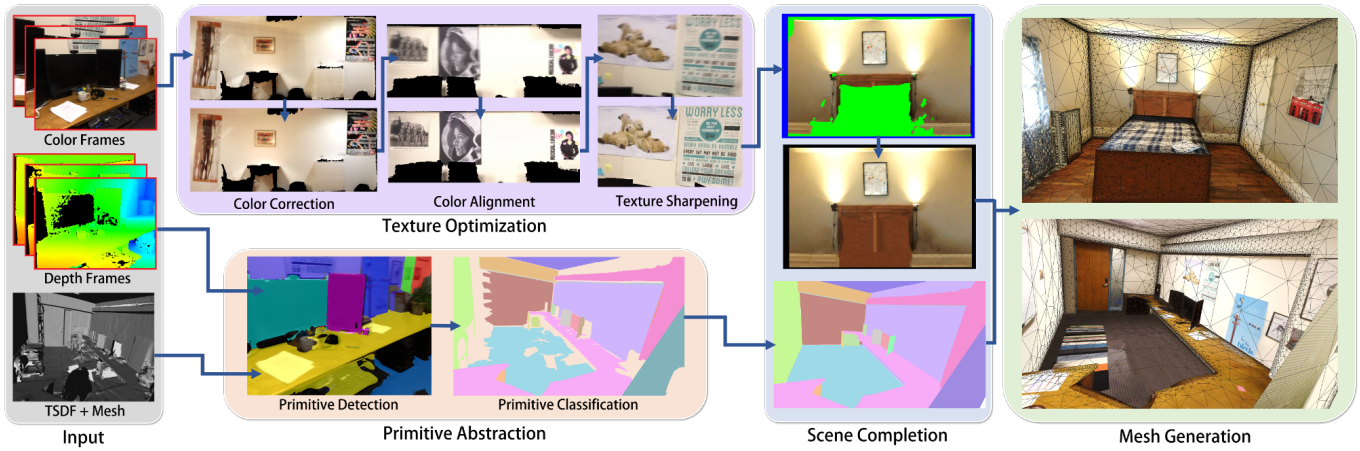


Fig. 2. Framework overview: our method takes a set of RGB-D frames as input, from which we compute a primitive abstraction that is used to optimize for sharp surface textures and infer missing scene parts. In the end, we obtain a low-polygonal, lightweight 3D reconstruction.

mitigate camera drift, achieving improved reconstruction quality in a KinectFusion-style framework. In order to reduce sensitivity towards potential errors in initially detected structural correspondences, Halber and Funkhouser [2017] employ a hierarchical optimization approach incorporating planar relationship constraints with sparse features, enabling registration of very long RGB-D scanning sequences.

In addition to improving tracking robustness, Dzitsiuk et al. [2017] use plane priors estimated directly on the implicit signed distance field representation of the scene to further de-noise and complete the reconstruction, producing very stable geometry even during real-time updates. Surface measurements near detected planes are replaced with the plane measurements to reduce noise, and plane geometry is extrapolated in unobserved space to fill holes. Our approach similarly exploits planes as a clean geometric representation of a 3D scene which can be leveraged to complete 3D scenes; however, Dzitsiuk et al. [2017] focus solely on geometry denoising and completion, whereas the goal of 3DLite is to create visually compelling models by generating high-quality texture for complete 3D models.

Color optimization for 3D reconstruction. Various approaches have been developed to create a color map on a geometric model from multiple input images. Several techniques use manually selected point correspondences to provide image-to-model registration [Neugebauer and Klein 1999; Ofek et al. 1997; Rocchini et al. 1999; Stamos and Allen 2000]. Given precisely registered pairs of color and depth, camera poses can be optimized for to maximize photo-consistency [Bernardini et al. 2001; Johnson and Kang 1999; Pulli et al. 2005; Pulli and Shapiro 2000]. For 3D scanning using commodity-sensor data with various misalignments arising from coarse geometry and optical irregularities, Zhou and Koltun [2014] account for these issues by optimizing for both the rigid camera poses as well as non-rigid warping for each image to maximize dense photo-consistency. Bi et al. [2017] build on this work: they synthesize a set of photometrically consistent aligned color images to produce high-quality texture mapping even under substantial geometric inaccuracies. We also build upon the approach of Zhou et al. [Zhou and Koltun 2014]; however, for our room-scale scenario, optimizing purely for a dense energy term remains sensitive to initial poses and easy to end up in local minima. Rather, we employ a sparse-to-dense optimization, using sparse color features and geometric primitive constraints to help reach the basin of convergence of the dense photo-consistency energy.

3 OVERVIEW

From an input RGB-D video, 3DLite first computes a primitive-based abstraction of the scene, and then leverages this representation to optimize for high-quality texture maps, as well as complete holes in the scene with both geometry and color; see Fig. 2. We capture the input RGB-D stream with a handheld, consumer-grade RGB-D sensor. Using a modern RGB-D reconstruction system (i.e., BundleFusion [Dai et al. 2017b]), we compute initial camera poses for each frame and a truncated signed distance field (TSDF) representation of the scene, from which we extract an initial mesh. To generate the primitive-based abstraction, we detect planes for each frame, and then merge them into scene primitives according to the estimated camera poses (see Sec. 4). We then optimize for the global geometric structure by favoring primitives to support a Manhattan world assumption, so as to encourage orthogonal and parallel structures.

From this lightweight representation of the scene, we then optimize for texture maps over the geometry, directly addressing the issues of motion blur and small camera pose misalignments. We apply an exposure correction to achieve consistent color across the input images, which may vary with auto-exposure and white balancing (see Sec. 5.2). We must then refine the camera poses to precisely align the color frames to the new model geometry. To this end, we build upon the approach of Zhou and Koltun [2014] to optimize for refined camera poses and non-rigid image corrections. For our large-scale scanning scenario, we introduce sparse color feature and geometric primitive constraints to help bring the optimization into the basin of convergence of a dense photometric consistency energy (see Sec. 5.3). In order to account for motion blur in input color

images, we introduce a method to sharpen color projected from input frames to the model. Rather than select sharpest keyframes throughout the input video, which may still select relatively blurry frames or lose color information by filtering out too many frames, we seek sharp image regions, from which we formulate a graph-cut based optimization for image sharpness and coherence (see Sec 5.4). This leads to a high-quality texture map over the known geometry.

We then complete the textured model to fill holes that were occluded or unseen in the original scan. While general, high-resolution scene completion is a very challenging task, our primitive-based abstraction enables effective hole-filling for both geometry and color on our scene representation. To complete the geometry of the model, we extrapolate primitives in unobserved space according to the camera trajectory, such that each primitive meets either another primitive or empty space (see Sec 6.1). We then complete the color in these regions through image inpainting, following Image Melding [Darabi et al. 2012] (see Sec. 6.2). This produces a clean, complete, lightweight model mapped with sharp textures.

4 PRIMITIVE-BASED ABSTRACTION

To compute our primitive-based abstraction of a scanned scene, we first detect planes for each input frame, then merge these detected planes into a globally consistent set of primitives, and finally perform a structural refinement, optimizing under parallel and orthogonal constraints.

4.1 Frame-based Plane Detection

From an input RGB-D video sequence comprised of a set of depth and color frames $\{f_i = (C_i, \mathcal{D}_i)\}$, we first use a state-of-the-art RGB-D reconstruction system to obtain initial camera poses T_i (frame-to-world) for each frame, and an initial surface $\mathcal{S}_0$. For each frame, we detect planes using the fast plane extraction of Feng et al. [2014]. Instead of detecting planes directly on the input sensor depth, which is noisy and often contains holes, we operate on depth rendered from $\mathcal{S}_0$. This rendered depth contains data accumulated over previous frames, which regularizes out noise and incorporates more information than a single input depth map, and is also well-aligned to the model geometry $\mathcal{S}_0$. For each detected plane $\mathcal{P}_k$ in frame f_i, we additionally store two terms containing information used for primitive merging and structural refinement, as described in Secs. 4.2 and Sec. 5.3.

- Plane parameter: $p = (n_x, n_y, n_z, w)$ where $\mathbf{n} = (n_x, n_y, n_z)$ is the unit normal. It represents the plane $\mathbf{n} \cdot \mathbf{x} + w = 0$ in the camera space of f_i.
- Distance Matrix: $\mathbf{D} = \frac{1}{N} \sum_q x_q x_q^T$, where x_q is the world space position of the q-th pixel in the depth map. We can then easily compute the average of square point-plane distances (ASD) of $\mathcal{P}_k$ with another plane $\mathcal{P}_l$ in frame f_j as $(p_l^j T_j^{-1}) \cdot \mathbf{D} \cdot (p_l^j T_j^{-1})^T$.

4.2 Primitive Classification

We need to aggregate the per-frame planar regions into a set of plane primitives representing the planes in the 3D scene. Two planar regions $\mathcal{P}_k$ from frame f_i and $\mathcal{P}_l$ from frame f_j are determined to be the same primitive if they are close and have enough overlap. We consider $\mathcal{P}_k$ and $\mathcal{P}_l$ to be close if $\max(\text{ASD}_{k\to l}, \text{ASD}_{l\to k}) < \tau_c$ (in practice, we use $\tau_c = 0.05$m). Overlap is computed by transforming $\mathcal{P}_k$ and into the camera space of f_j and computing the percentage of overlapping pixels relative to the minimum number of pixels belonging to $\mathcal{P}_k$ and $\mathcal{P}_l$. If this overlap percentage is greater than τ_o (we use $\tau_o = 0.3$), the regions are considered to overlap.

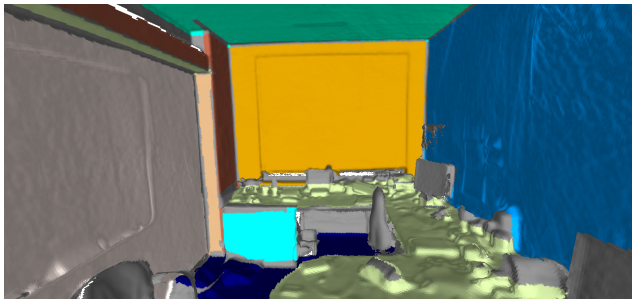


Fig. 3. Primitive decomposition: each plane primitive is denoted with a different color. Gray indicates no primitive association.

We perform this merging step hierarchically. For key frames at interval $n = 10$ frames, we merge planes within each key frames' set of n frames, and then merge planes among all key frames. Each frame then contains a set of plane primitives labeled according to the resulting merged set. By projecting the merged primitive labels from the frames onto $\mathcal{S}_0$, we obtain a primitive decomposition of the scene. Note that regions with multiple different primitive label assignments (typically occurring near intersections of different primitive) are removed from the primitive set. Fig. 3 shows an example primitive decomposition of an office scan, with different primitive denoted by different colors, and gray denoting no label assignment.

To filter out any potential outliers from the primitive classification projection onto $\mathcal{S}_0$, we filter out small plane primitives (area less than 0.2m^2) and use a RANSAC approach to filter out outlier points of larger primitives. For a plane primitive $\mathcal{P}_k$, we check for outliers by randomly selecting 3 of its associated vertices in $\mathcal{S}_0$ and fitting a plane $\mathcal{P}'_k$ to these vertices. A point is then considered an inlier for $\mathcal{P}'_k$ if its distance is smaller than τ_i. Since $\mathcal{S}_0$ often contains warping in large plane primitives due to sensor noise and distortion, as well as camera pose micro-drift, we conservatively set $\tau_i = 0.1$m. We repeat this process for 128 iterations, and update $\mathcal{P}_k$ with the best fitting plane to the largest set of inliers in the least squares sense.

4.3 Structural Refinement

Since we use a relatively conservative inlier threshold to compute the plane primitives, planes which should be orthogonal or parallel to each other are typically off by a few degrees. We thus perform a structure refinement optimization on the planar primitives to encourage them to conform to a Manhattan world. Similar to Halber and Funkhouser [2017], we formulate an energy minimization using parallel and orthogonal constraints:

$$E_s = E_d + \lambda E_a \tag{1}$$

$$E_d = \sum_{i}^{\#\text{planes}} \sum_{j}^{\#\text{plane verts}} D(\mathcal{P}_i, v_{ij})^2 , \tag{2}$$

$$E_a = \sum_{i,j \in \Omega} |A(\mathcal{P}_i, \mathcal{P}_j) - 90 \cdot n_{ij}|^2 . \tag{3}$$

For each plane primitive $\mathcal{P}$, E_d measures the plane fitting error, where $D(\mathcal{P}, v_i)$ is the distance from an inlier vertex v_i to $\mathcal{P}$. E_a measures the angle error between orthogonal planes, where $A(\mathcal{P}_i, \mathcal{P}_j)$ is the angle of two planes, Ω is a set of parallel and orthogonal plane pairs. The set Ω is collected by testing each pair of plane primitives, and adding them to the set if the angular difference between their normals lies in $[90n_{ij} - 10, 90n_{ij} + 10]$, where $n_{ij} \in \{0, 1, 2, 3\}$. The final energy E_s is then a linear combination of E_d and E_a, with $\lambda = \frac{E_a^0}{E_d^0}$ to balance the plane fitting error and structure error.

The optimization typically converges in about 10 iterations, with resulting angle errors less than 1° and plane fitting error about 1.1 times larger. As a result, we can rectify the orthogonal structure of the planar primitives with very small trade-off from fitting accuracy. From this optimized result, we can produce a lightweight, clean mesh $\mathcal{S}_p$ by projecting the vertices of $\mathcal{S}_0$ to their associated primitives, as shown in Fig 7(a).

5 TEXTURE OPTIMIZATION

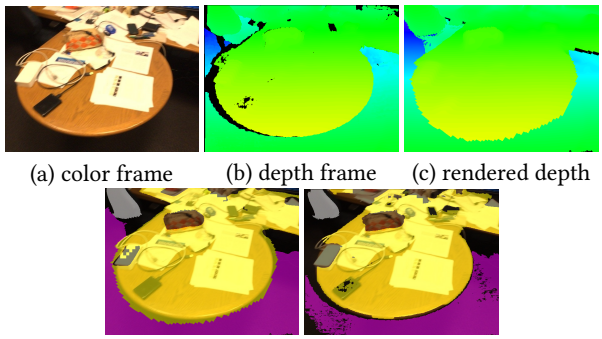
We aim to map our primitive-abstracted geometric model $\mathcal{S}_p$ with sharp, clear textures to produce a visually compelling 3D model. It is not suitable to directly project pixels from each color frame to the geometric model, due to the issues of motion blur and camera misalignments. In our texture optimization step, we directly address these problems with a texture optimization method to solve for refined camera poses and image warping, as well as a texture sharpening step which considers per-pixel image sharpness to solve for globally crisp colors.

5.1 Color-based Primitive Refinement

Since we originally detected plane primitives from rendered depth (Sec. 4.1), there may be some disagreement along primitive boundaries with the input RGB-D frame. Thus pixels that are close to primitive boundaries are easily mis-classified, and we must re-classify them to agree with the input color images. To ensure that primitive labels are coherent in the image domain and maintain boundaries consistent with the input color frames, we formulate a graph-cut [Boykov et al. 2001] based energy minimization problem:

$$E_l = \sum_p D(p, l_p) + \sum_{p,q} V_{pq} \delta(l_p, l_q) . \tag{4}$$

Here $D(p, l_p)$ indicates the penalty for pixel p to be classified as l_p. For confident regions, we won't change their labels l_p^0. So $D(p, l_p)$ is 1 for $l_p = l_p^0$ and inf for other labels. For regions that need to be reclassified, $D(p, l_p) = 1$ for all labels l_p. We additionally have a penalty at the boundary of the primitives, $\delta(l_p, l_q)$: p and q are neighbor pixels and δ is 1 if $l_p \neq l_q$ and 0 otherwise. V_{pq} measures how good it is to cut between p and q, depending on their color differences. We set $V_{pq} = e^{(||C(p)-C(q)||^2)/\sigma^2}$. For our experiments, we re-classify pixels within 10 pixels of the boundary of a 640×480 image, and use $\sigma = 30$ with colors $C(x) \in [0, 255]$. Figure 4 shows the result of our refinement.



(a) color frame (b) depth frame (c) rendered depth

(d) plane labels (e) refined labels

Fig. 4. Color-based primitive refinement: (a) Input color. (b) Input depth. (c) Rendered depth. (d) Primitive classification from rendered depth. (e) Refined primitive classification according to the color frame.

5.2 Color Transfer Optimization

In solving for a texture mapping, it is important to ensure consistent color across different views (e.g., due to auto-white balancing or auto-exposure). This can be achieved by transferring colors with a mapping function. For example, NRDC [HaCohen et al. 2011] search for correspondences between images, and optimizes for such a mapping function to transfer color style from one image to another. In the scenario of RGBD scanning, correspondences can be easily acquired from camera poses and geometry. Zhang et al. [2016] solves the problem by minimizing vertex radiance error viewed from different color frames, given the exposure. In order to model white balance and exposure changes, we refer to the NRDC model to transfer colors using three spline curves $(B_j(r, g, b) = (B_j^1(r), B_j^2(g), B_j^3(b)))$ for the rgb channels of the j-th frame. Then for a 3D point p_i and it's corresponding pixels $\{q_{ij}\}$ in frames $\{j\}$ under the current camera poses, we want the color $C(p_i)$ to be close to $B_j(q_{ij})$:

$$E_t = \sum_i \sum_j ||C(p_i) - B_j(q_{ij})||^2 + \lambda \sum_i \sum_j (B_j'(x_i) - 1)^2 . \tag{5}$$

While [Zhang et al. 2016] use the first frame as a reference exposure, we don't need this constraint. Instead, we regularize the derivatives of the transfer functions $B_j'(x_i)$ to be ≈ 1. This helps preserve the variance of the color. We use $\lambda = 0.1$, and x_i is a sequence of 10 integers from 0 to 250 with interval 25. Since this optimization depends on the quality of the camera transforms, we iterate this color correction step with the color image alignment optimization described below.

5.3 Refined Texture Map Alignment

Key to achieving high-quality texture is obtaining precisely aligned color frames. This is a challenging task, as there are artifacts from consumer-grade sensors and the geometry is often imprecise. The

color map optimization approach of Zhou and Koltun [2014] considers fixed geometry and jointly optimizes for color camera poses and image warping parameters. This reduces the dimensionality of the problem, using image warping to reduce potential color misalignment from geometric error, achieving compelling results for object scans. However, their optimization relies on dense photometric error, which has a relatively small basin of convergence and is thus rather sensitive to initial pose estimates. For our large-scale scanning scenario, this energy is not very robust to the variance in initial camera poses.

In our formulation, we solve for a set of camera parameters for each frame separately from non-rigid correction, which we found to be more effective in practice. We adopt the idea of EM optimization with the help of dense color errors from Zhou and Koltun, and solve to maximize photo-consistency. To aid convergence, we additionally introduce sparse feature and primitive-based constraints into the optimization. Thus our energy is defined as

$$E(\mathbf{T}) = E_c(\mathbf{T}) + \lambda_s E_s(\mathbf{T}) + \lambda_p E_p(\mathbf{T}),$$

where E_c represents the dense photometric energy, E_s the sparse feature term, and E_p primitive relationship constraints, and we solve for rigid camera poses (camera-to-world) $\mathbf{T} = \{T_i\}$ for each frame. Following Zhou and Koltun, we also optimize for proxy variables $C(p_i)$ denoting the color at 3D point p_i on the geometry surface. We perform this optimization hierarchically in coarse-to-fine fashion to aid convergence at fine-scale resolution, using 3 hierarchy levels in which p_i are sampled at 0.032m, 0.008m, 0.002m from the primitive-abstracted $\mathcal{S}_p$, respectively. λ_s and λ_p are set such that $E_c(\mathbf{T}_0) = \lambda_s E_s(\mathbf{T}_0) = \lambda_p E_p(\mathbf{T}_0)$.

Sparse Term. In our sparse matching term, we minimize the sum of world-space distances between pairs of sparse feature correspondences transformed from image space to the space of the 3D primitives. For a frame f_i and plane primitive $\mathcal{P}_k$, we transform a pixel p_i under the homography H_k^i, depending on $\mathcal{P}_k$ and the camera pose T_i, to bring it into the space of the primitive $\mathcal{P}_k$. Then for a pixel correspondence $\{p_{im}, p_{jm}\}$ from frames f_i and f_j and which correspond to 3D primitives $\mathcal{P}_k$ and $\mathcal{P}_l$ (when projected in 3D under the current camera poses), we optimize for the energy

$$E_s(\mathbf{T}) = \sum_m^{\#\text{corr}} ||H_k^i p_{im} - H_l^j p_{jm}||^2 . \quad (6)$$

Sparse features are detected by projecting each color frame into the plane primitives, detecting SIFT [Lowe 2004] features in the projected images. Correspondences are determined by SIFT matching followed by a verification step to ascertain that there is a valid 2D rigid transform bringing one set of correspondences from an image to another. Valid correspondence features are then back-projected to the original color frames to compose the sparse feature set. This produces a robust set of feature correspondences, since the projection into a 3D plane does not rely on depth maps which contain noise and distortion, and matching in the warped space contains reduced scale and affine variance than in the original color images.

Primitive Constraint Term. We restrict the per-frame planar regions computed in Sec 4.1 to align well with $\mathcal{S}_p$, geometrically.

$$E_g(\mathbf{T}) = \sum_i \sum_j (P_j T_j^{-1} p_i)^T (P_j T_j^{-1} p_i) . \quad (7)$$

Here, p_i is the homogeneous coordinate of the sampled 3D points from $\mathcal{S}_p$. P_j is the camera-space plane parameter (as in Sec. 4.1) of the i-th planar region of frame j.

Dense Term. Finally, the dense term measures photo-metric error from the color frames projected onto $\mathcal{S}_p$.

$$E_c(\mathbf{T}) = \sum_i ||C(p_i) - \sum_j I_j(\pi(T_j^{-1} p_i))||^2 , \quad (8)$$

where p_i is in the set of sampled 3D points of $\mathcal{S}_p$ and π denotes the perspective projection.

We solve for $E(\mathbf{T})$ by solving for both the per-frame transforms $\mathbf{T}$ and the proxy variable colors on the geometry $C(p_i)$. After an initial three iterations at the coarsest resolution, we are able to compute very reliable sets of correspondences for a color transfer correction (as described in Sec. 5.2); after the color correction, we further optimize for refined camera poses at the middle and high resolutions for five and two iterations respectively.

After solving for the rigid camera poses, we additionally optimize for non-rigid image warping to reduce color misalignment due to possible geometric error, following Zhou and Koltun [2014].

5.4 Texture Sharpening

Motion blur is a common problem when capturing handheld video using commodity RGB-D sensors. Even with perfect alignment of camera frames, a blurry image can still significantly decrease the quality of the model color. Most existing scanning methods average colors from all corresponding frames, or update the color online with a weighted combination. In order to mitigate this issue, the color map optimization of Zhou and Koltun [2014] selects sharpest keyframes every 1 to 5 seconds, with sharpness of a frame evaluated using the method by Crete et al. [Crete et al. 2007]. However, when we applied this to our scenario, not all blurry frames were filtered out, and if keyframes were selected with interval greater than 1 second, color information was lost in some regions. In fact, in the room-scale setting, this tends to preserve views where the camera is closer to the scene, as such images typically have more objects and thus higher color variance. However, we wish to preserve cameras with closer views to the scene, which capture textures with more detail. Thus we instead aim to only map the sharpest regions of the images onto the geometry; i.e., for each primitive in $\mathcal{S}_p$ we map the sharpest region of color from any single image. To this end, we consider both image pixel sharpness and image region sharpness in order to select key frames at an intervals of [10,20] frames. For image pixel sharpness, we use the metric of [Vu et al. 2012], and for image region sharpness we consider both pixel sharpness and visual density, where visual density measures the quality of the view of a pixel (accounting for far distances and glancing angles), and is defined as

$$r_j(p_i) = \left((T_j^{-1} \pi^{-1}(p_i))\big|_z \right)^{-2} \cdot \cos(\mathbf{r} \cdot \mathbf{n}) , \quad (9)$$

where p_i is a pixel of frame j, $\mathbf{r}$ is the ray from the camera to p_i, and $\mathbf{n}$ the camera-space normal at the 3D point corresponding to p_i. The region sharpness for a pixel is then defined as $S_j^{\text{reg}}(p_i) = S^{\text{pix}}(p_i) \cdot r_j(p_i)$.

Since we wish to avoid noise potentially introduced by sampling from many different frames for similar 3D locations, we formulate a graph-cut [Boykov et al. 2001] based energy optimization which further incorporates frame coherence:

$$E = \sum_p |S_{l(p)}(p) - S^{\max}(p)| + \lambda \sum_{(p,q)} ||C_{l(p)}(\pi(T_{l(p)}^{-1} p)) - C_{l(q)}(\pi(T_{l(q)}^{-1} q))|| \cdot \delta(l_p, l_q), \quad (10)$$

solving for frame index $l(p)$ for each point $p \in \mathcal{S}_p$. The first term encourages sharpness, with $S^{\max}(p)$ denoting the sharpest possible score for p. The second term encourages coherence by penalizing neighboring points p and q if they are associated with different frames, with penalty proportional to the respective color difference in order to achieve a seamless graph cut.

Figure 5 shows a challenging example, where a table texture is composed of projections from the 147 sharpest color frames of more than 3000 images. We see that directly sampling from frames with the highest region sharpness value ((c),(d)) improves upon the texture sharpness of Zhou and Koltun [2014] ((a),(b)), but contains some noise due to lack of coherence in the frames being sampled from. Our final texture sharpening result ((e), (f)) balances both sharpness and frame coherence, preserving sharpness while reducing noise.

Using direct color sampling can still result in noticeable artifacts when transitioning from sampling from one frame to another, as shown in Fig. 6(b). We perform a final step in to mitigate these effects in the sharpening process. We combine the divergence map $\triangle C(p)$ from the labeling resulting from solving Eq. 10, which represents texture information [Pérez et al. 2003], with the color map $\bar{C}(p)$ computed by averaging, and solve for the final texture $F(p)$ in the least squares optimization

$$E(F) = \sum_p ||F(p) - \bar{C}(p)||^2 + \lambda ||\triangle F(p) - \triangle C(p)||^2 . \quad (11)$$

As shown in Fig. 6, this maintains sharpness while removing boundary inconsistencies.

6 SCENE COMPLETION

While our texture-optimized, lightweight mesh contains sharp, compelling color, it nonetheless remains incomplete, due to occlusions or limited scanning coverage. 3D completion is a challenging task, as it typically requires example-based learning techniques [Dai et al. 2017c], and the problem becomes cubically more complex with higher resolutions and larger spatial extents. We thus exploit our planar primitive abstraction to simplify 3D scene completion in both geometry and color.

6.1 Geometry Completion

A plane-based primitive abstraction enables geometry completion by plane extrapolation in unseen regions [Dzitsiuk et al. 2017]. We use several rules to guide our completion approach, visualized in Fig. 8:

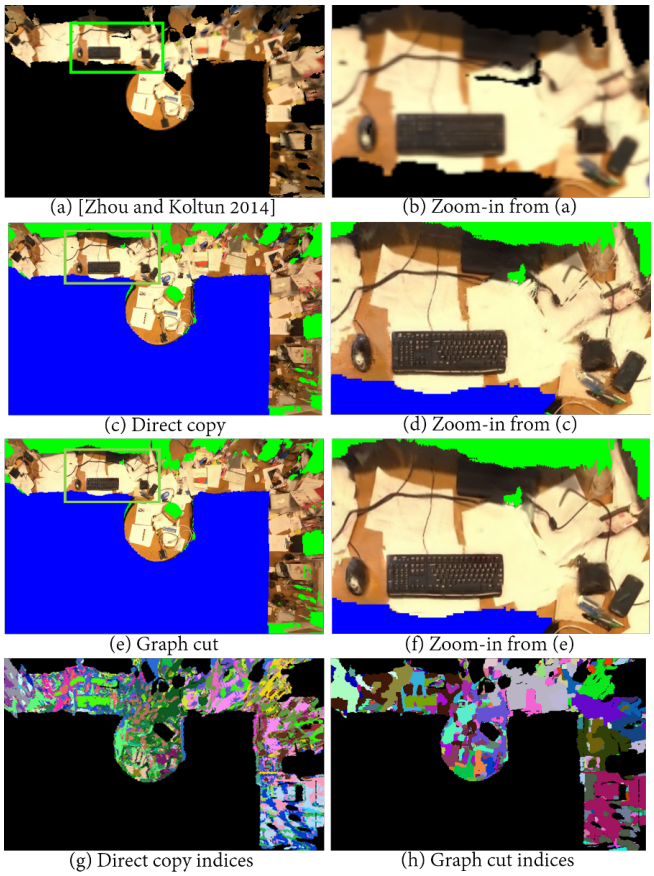


Fig. 5. Texture generation from multiple frames: (a) Color map optimization [Zhou and Koltun 2014] (averaging). (b) Zoom-in from (a). (c) Direct copy from sharpest region. (d) Zoom-in from (c). (e) Balance sharpness and region coherence. (f) Zoom-in from (e). (g) Frame indices with sharpest local region. (h) Optimized frame indices for color sampling.

- Two planes should be extrapolated to their intersection if the extrapolated area is unobserved.
- If three planes will intersect each other, extrapolate them to meet at a corner if the extrapolated area is unobserved.
- No planes should be extrapolated into open space (observed to be empty). Note that we use the initial TSDF to determine known occupied, known empty, and unobserved space, according to the camera trajectory.
- Holes self-contained in a plane should be filled if they are in unobserved space.

We search for all pairs of planes satisfying the first rule and extrapolate them. Then, we find all triplets of planes under the second rule and extrapolate them. Our algorithm is invariant to the extrapolation order since the final state forces all planes to meet in unobserved areas. For each pair of planar primitives, we attempt to detect potential extrapolation regions, and then repeat the same for all sets of three planar primitives which intersect. Finally, we fill in unobserved holes self-contained within planar primitives. Note that while this process achieves successful completion for scenes in which parts of all major planes have been observed, we cannot

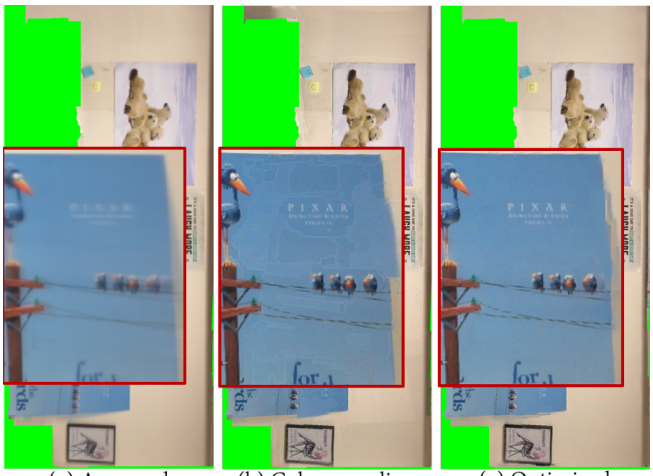


Fig. 6. Texture optimization combining averaged color and sampled divergence map. (a) Averaged color from all selected frames. (b) Direct color sampling from optimized frame indices. (c) Optimization combining averaged color and sampled divergence map.

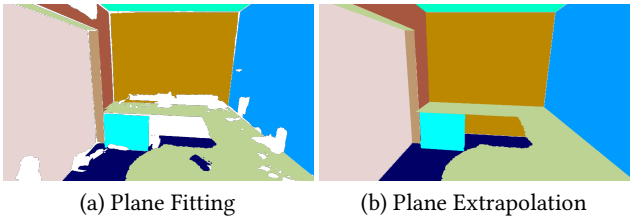


Fig. 7. Primitive fitting and extrapolation: (a) After primitive fitting and structural refinement, we project vertices to the primitives, producing a clean mesh. (b) We extrapolate primitives in occluded regions to close the surface.

generate geometry from scratch in the case of very large regions of unobserved scene geometry (see Sec 8.1 for further discussion).

Figs. 7 and 9 demonstrate our extrapolation and hole-filling to generate complete scenes.

6.2 Texture Completion

Our plane-based primitive abstraction reduces the 3D texture completion problem into a two-dimensional one over plane textures. As this texture synthesis problem is a classical one and has been well-studied [Barnes et al. 2009; Criminisi et al. 2004; Darabi et al. 2012; Pathak et al. 2016; Simakov et al. 2008; Wang et al. 2016; Yang et al. 2016], we employ state-of-the-art image inpainting techniques to complete texture in geometry-extrapolated regions.

Although recent deep learning based methods have shown impressive results, they require a large training set and are mostly constrained to fixed image resolutions. Thus we use Image Melding [Darabi et al. 2012], a patch-based approach incorporating image transforms and gradients into a mixed ℓ_2/ℓ_0 optimization to achieve high-quality image inpainting. We found the method to work well

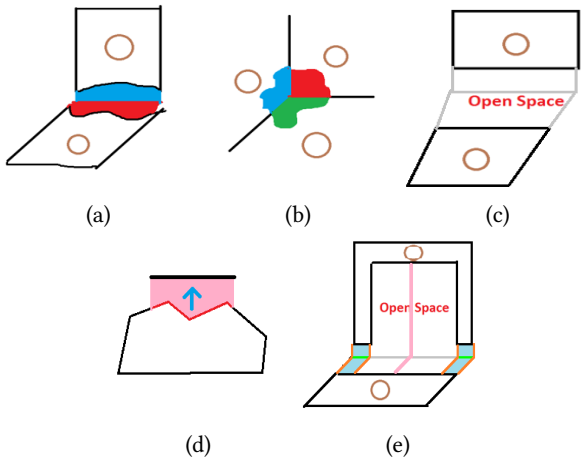


Fig. 8. Rules for plane extrapolation. (a) Two planes should extrapolate to their intersection. (b) Three planes should extrapolate to their intersection. (c) No planes should be extrapolated through known empty space. (d) Extrapolation (light red) can be viewed as a certain boundary (red) extending in the direction (blue) orthogonal to the intersection line. (e) A more complex example, showing invalidation of potential extrapolation due to known empty space. Only the blue regions should be extended towards the intersection.

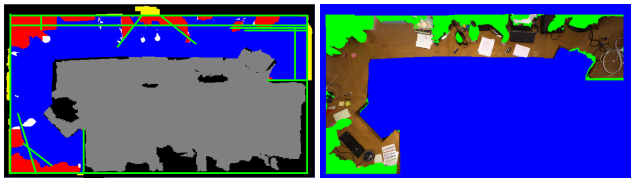
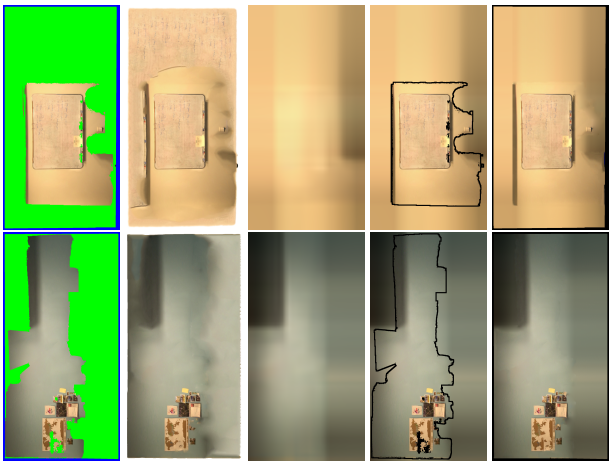


Fig. 9. An example of extrapolation and hole filling. Left: Green lines represent possible intersections between sets of planes. Blue represents the original scene geometry, red the extrapolated regions, yellow removed original geometry extending past intersections, gray known empty space, and white self-contained holes which were filled. Right: Completed plane primitive with texture.

for most cases, except when there are incomplete regions of a texture which cover a distinct foreground and smooth background, or when there are shadows on the background, as shown in Fig. 10(b). In these cases, we detect the background using image segmentation [Felzenszwalb and Huttenlocher 2004], and extend it to fill the entire image using laplacian smoothing (Fig. 10(c)). We then combine the synthesized background with the foreground (Fig. 10(d)), and synthesize pixels less than 10 pixels from the foreground using Image Melding (Fig. 10(e)).

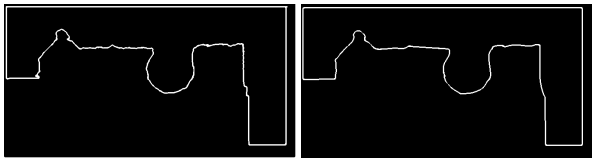
7 MESH GENERATION

In section, we discuss the procedure to generate the final lightweight mesh. We first denoise plane boundaries and then convert the planar primitive abstraction to a mesh, producing a final model $\approx$ 25 times smaller than the original mesh.



(a) (b) (c) (d) (e)

Fig. 10. Texture completion. (a) Original texture with green denoting regions to be synthesized. (b) Direct application of Image Melding [Darabi et al. 2012]. (c) Background color estimation. (d) Combined background and original texture. (e) Remaining pixels inpainted with Image Melding.

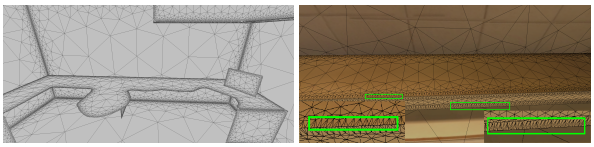


(a) Noisy boundary (b) Smoothed boundary

Fig. 11. Smoothing the boundary. Initial plane primitive boundaries are noisy (left), which we smooth with B-Splines and lines (right).

Boundary Refinement. Plane boundaries are often noisy, due to noise and distortion in the input sensor data and the initial reconstruction. To denoise the boundaries, we smooth them with a combination of line and B-spline fitting. Lines are fit to each vertex on the boundary (sampled at every 8mm) by using its 51 neighboring vertices. A vertex belongs to a line if the condition number of the covariance of these vertices is larger than 50. We then iteratively merge line-associated neighbor vertices together if their lines are no more than 5° apart. We fit B-splines to each consecutive sequence of vertices not belonging to any lines.

Primitive Remeshing. To generate our final mesh, we triangulate the planar primitives using constrained Delaunay triangulation [Chew 1987]. Figure 12(a) shows the resulting simplified triangle mesh, which accurately preserves primitive boundaries. Because we use discretized pixels to generate the original primitive boundaries, there can be small gaps between primitives which should be connected. To close these gaps, we project vertices near primitive intersections to the intersecting primitive, as shown in Figure 12(b).



(a) Mesh simplification. (b) Closing surfaces.

Fig. 12. Mesh simplification and vertex projection. (a) We simplify the planar primitives with constrained Delaunay triangulation. (b) We apply vertex projection to close the surfaces which should be connected.

	BundleFusion Scenes			ScanNet	
Scenes	office0	office1	office3	0567_01	0451_05
Region(s)	478	503	531	472	523
Plane(s)	21.5	17.1	23.2	16.5	25.4
Sharpness(s)	1138	1186	1266	811	8436
Color(s)	11458	9553	14432	9866	9315
Boundary(s)	1.156	1.101	1.157	0.781	1.384

	ScanNet			other	
Scenes	0294_02	0271_01	0220_02	apt	offices
Region(s)	535	511	517	585	897
Plane(s)	23.9	18.7	27.4	38.1	61.8
Sharpness(s)	1011	828	935.4	1151	1968
Color(s)	16407	9601	8575	18226	35976
Boundary(s)	1.219	0.921	1.406	2.016	3.725

Table 1. Timings (seconds) for different components.

8 RESULTS

We tested our method on a variety of RGB-D scans of indoor scenes. All scans were captured using a *Structure Sensor*[2] mounted to an iPad Air, including several sequences from the BundleFusion data [Dai et al. 2017b] as well as the ScanNet [Dai et al. 2017a] dataset. We use 640×480 color and depth, synchronized at 30Hz. Note that we are agnostic to the type of RGB-D sensor used. For each scan, we compute the initial camera poses using BundleFusion. All 3DLite experiments were performed on an Intel Core i7 2.6GHz CPU, with each scan taking on average 5 hours to process. Detailed timings are shown in Table 1.

Qualitative Comparison. We first compare our lightweight, textured meshes of 10 scenes with 3D reconstructions computed using BundleFusion and VoxelHashing [Nießner et al. 2013]. Note that since BundleFusion's online re-integration updates use color discretized to bytes, there may be some small color artifacts in the resulting reconstruction, so we first run BundleFusion to obtain camera poses and then compute a surface reconstruction with VoxelHashing. As shown in Figs. 13 and 14, even with simplified geometry, our high quality textures provide visually compelling results on completed scenes, drastically reducing various color oversmoothing artifacts.

Primitive Abstraction. We show several examples of our primitive-based geometry in Fig. 15, which allows us to produce denoised and complete geometry compared to the original reconstruction, in

[2]https://structure.io

Fig. 13. Reconstructed meshes with high-quality surface textures. 3DLite produces completed scenes while significantly reducing color artifacts compared to reconstructions generated with BundleFusion [Dai et al. 2017b] and VoxelHashing [Nießner et al. 2013].

	BundleFusion Scenes			ScanNet	
Scenes	office0	office1	office3	0567_01	0451_05
FaceNum	62559	68874	63479	43153	89621

	ScanNet			other	
Scenes	0294_02	0271_01	0220_02	apt	offices
FaceNum	58709	58790	82864	92898	174200

Table 2. Number of faces in 3DLite models.

a much more lightweight representation, which is more than 300 times smaller than the original mesh under the same resolution. The face number of our scenes is shown in Table 2.

Compared to traditional plane fitting using RANSAC, our method is more robust in detecting relatively small planes, as shown in Fig. 16.

Color Alignment. We analyze our color alignment approach, showing the effect of our sparse and primitive geometry terms in Fig. 17 (optimizing for rigid poses only). Without color optimization, the result contains oversmoothing and ghosting. Optimizing for rigid poses under a dense photometric error using Zhou and Koltun [2014] sharpens the image but still retains some artifacts where initial camera poses were too far from the basin of convergence. Our new terms are able to bring the optimization to a precise alignment of color frames.

Color Transfer Correction. We show the effect of the color transfer optimization in Fig. 18, significantly reducing not only visible artifacts from auto-exposure and auto-white balancing, but also various color inconsistencies which can occur even under fixed exposure and white balancing settings.

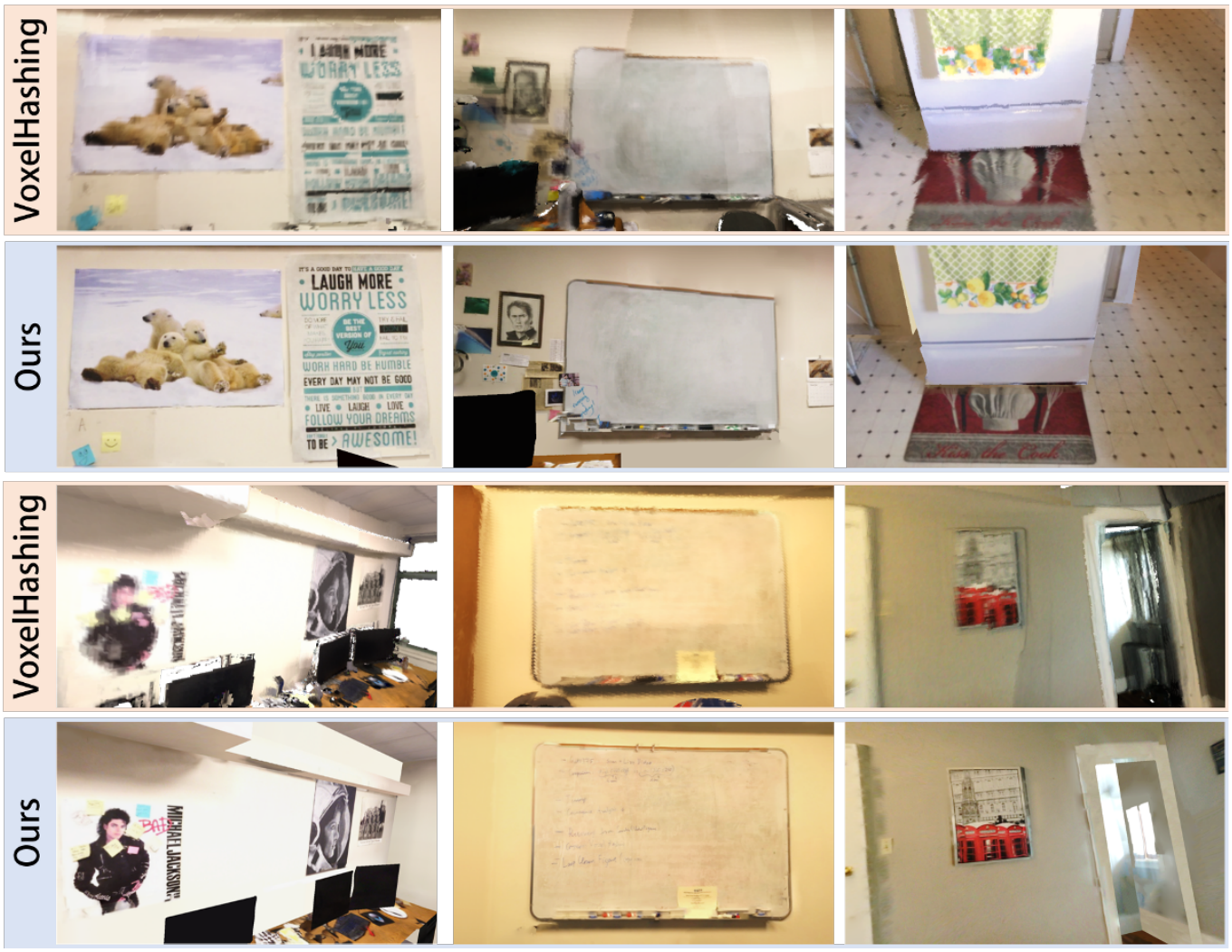


Fig. 14. Zoomed-in views showing our reconstructed 3d models compared to reconstructions produced by BundleFusion [Dai et al. 2017b] and VoxelHashing [Nießner et al. 2013].

Texture Sharpening. In Fig. 19, we show the effectiveness of our texture sharpening approach. Since we generate a consistent texturing from sharp regions of the input images, images rendered from our 3D model can be sharper than some of the original RGB images (which often contain motion blur), as well as deblurred RGB images using advanced video motion deblur techniques [Su et al. 2016]. Such deblurring, while noticeably reducing blur in several parts of the image, still has some difficulty near image boundaries, and with large motion.

Texture Completion. Fig. 20 shows several texture completion results using background filling along with Image Melding [Darabi et al. 2012] to synthesize colors in unobserved regions. We are thus able to generate complete scene models with textured geometry.

8.1 Limitations

While 3DLite can robustly generate lightweight, abstracted models with sharp textures, it still suffers from several limitations, as visualized in Fig. 21. If there is geometry entirely unseen in the original scan, we cannot generate it from scratch to complete these regions. Additionally, small objects (e.g., keyboards, mice) are often projected onto planar primitives; this is not too dissimilar from reconstructions generated from volumetric fusion, where the coarse resolution, noise, and distortion of a commodity depth sensor can also make small objects difficult to distinguish geometrically. Further, while state-of-the-art texture synthesis methods can achieve impressive results, they are not perfect, and may still have difficulty synthesizing color in large missing regions. Most notably, our current primitive abstraction is plane-based, so relatively non-planar objects (e.g., some chairs) are not captured in our geometric abstraction. We hope to extend our approach in combination with CAD

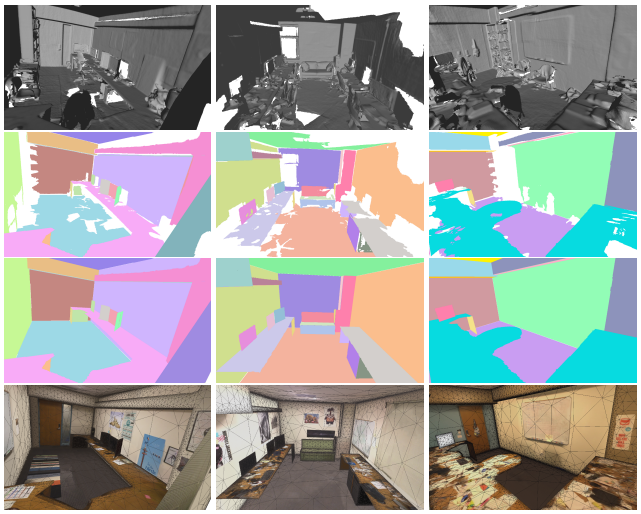


Fig. 15. Primitive-based Abstraction. First row: original mesh. Second row: after primitive fitting. Third row: after geometry completion by primitive extrapolation. Last row: final result after texture mapping and mesh generation.

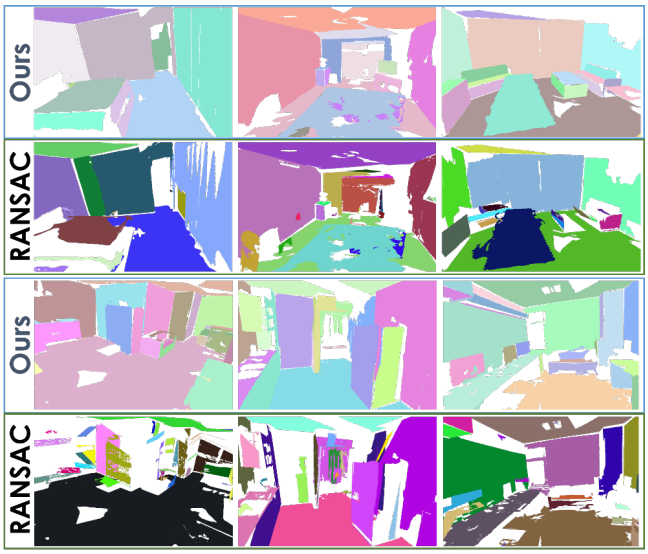
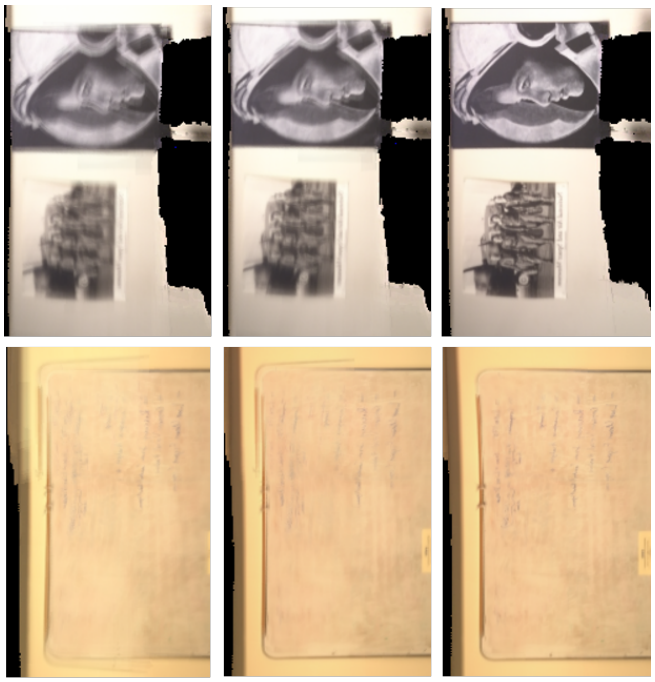


Fig. 16. Comparison between our method and plane fitting with RANSAC. Our method is more robust in detecting relatively small planes.

model retrieval, alignment, and texturing in order to retain all scene geometry and texture.

9 CONCLUSION

We have presented a new approach to generating visually compelling 3D reconstructions, making a step towards production-ready models for content creation pipelines by addressing the issues of oversmoothed, low quality color and incompleteness of existing



(a) No optimization (b) [Zhou and Koltun 2014] (c) Ours

Fig. 17. Color Alignment. (a) Color averaging. (b) Camera poses optimized with only dense color [Zhou and Koltun 2014]. (c) Camera poses optimized using our method, with dense color, sparse feature and geometry information.

large-scale 3D reconstructions. 3DLite focuses on generating high-quality textures on abstracted geometry through texture optimization and sharpening, generating a consistent 3D model with textures that can be sharper than the original RGB images. We further exploit our primitive-based representation to complete scene geometry and color in unseen regions in an input scan. We believe that this is a first step towards commodity 3D scanning for content creation, and we hope that this will pave the way for handheld 3D scanning to be used in production pipelines.

ACKNOWLEDGMENTS

This work is funded by a Google Tango research grant, a Google Focused Research award, NSF grants CCF-1514305 and IIS-1528025, a Samsung GRO award, and supported by a Stanford Graduate Fellowship.

REFERENCES

Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan Goldman. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics-TOG* 28, 3 (2009), 24.

Fausto Bernardini, Ioana M. Martin, and Holly Rushmeier. 2001. High-quality texture reconstruction from multiple scans. *IEEE Transactions on Visualization and Computer Graphics* 7, 4 (2001), 318–332.

Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. 2017. Patch-Based Optimization for Image-Based Texture Mapping. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)* 36, 4 (2017).

Yuri Boykov, Olga Veksler, and Ramin Zabih. 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence* 23, 11 (2001), 1222–1239.

Fig. 18. Color transfer correction. (a)(b), and (d) are taken with auto exposure and white balancing, and (c) with fixed exposure and white balancing. Our color transfer optimization compensates for color inconsistencies in both scenarios.

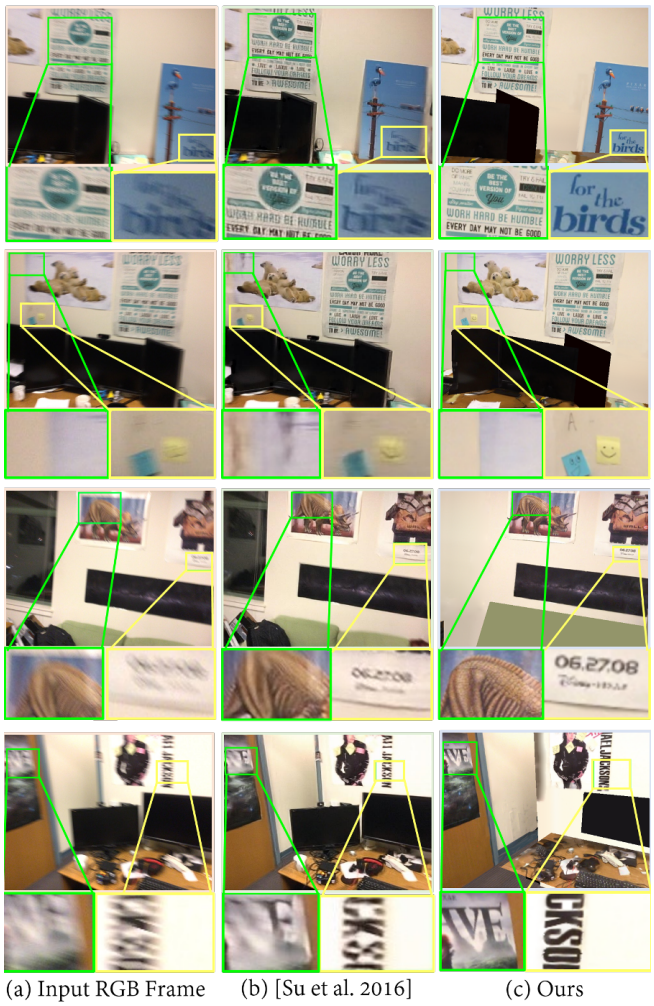


Fig. 19. Texture sharpening comparison. (a) Most input RGB frames are blurry. (b) Deep Video Deblurring [Su et al. 2016] reduces some of the blur. (c) Our approach produces consistently sharp results.

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

L Paul Chew. 1987. Constrained delaunay triangulations. In *Proceedings of the third annual symposium on Computational geometry*. ACM, 215–222.

Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. 2015. Robust reconstruction of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5556–5565.

Frederique Crete, Thierry Dolmiere, Patricia Ladret, and Marina Nicolas. 2007. The blur effect: perception and estimation with a new no-reference perceptual blur metric. In *Electronic Imaging 2007*. International Society for Optics and Photonics, 64920I–64920I.

Antonio Criminisi, Patrick Pérez, and Kentaro Toyama. 2004. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on image processing* 13, 9 (2004), 1200–1212.

Brian Curless and Marc Levoy. 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.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. 2017a. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. 2017b. BundleFusion: Real-time Globally Consistent 3D Reconstruction using On-the-fly Surface Re-integration. *ACM Transactions on Graphics 2017 (TOG)* (2017).

Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. 2017c. Shape Completion using 3D-Encoder-Predictor CNNs and Shape Synthesis. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B Goldman, and Pradeep Sen. 2012. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Trans. Graph.* 31, 4 (2012), 82–1.

Xavier Decoret, François Sillion, Gernot Schaufler, and Julie Dorsey. 1999. Multi-layered impostors for accelerated rendering. In *Computer Graphics Forum*, Vol. 18. Wiley Online Library, 61–73.

Mingsong Dou, Li Guan, Jan-Michael Frahm, and Henry Fuchs. 2012. Exploring high-level plane primitives for indoor 3D reconstruction with a hand-held RGB-D camera. In *Asian Conference on Computer Vision*. Springer, 94–108.

M. Dzitsiuk, J. Sturm, R. Maier, L. Ma, and D. Cremers. 2017. De-noising, Stabilizing and Completing 3D Reconstructions On-the-go using Plane Priors. In *International Conference on Robotics and Automation (ICRA)*.

Pedro F Felzenszwalb and Daniel P Huttenlocher. 2004. Efficient graph-based image segmentation. *International journal of computer vision* 59, 2 (2004), 167–181.

Chen Feng, Yuichi Taguchi, and Vineet R Kamat. 2014. Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 6218–6225.

Yoav HaCohen, Eli Shechtman, Dan B Goldman, and Dani Lischinski. 2011. Non-rigid dense correspondence with applications for image enhancement. *ACM transactions on graphics (TOG)* 30, 4 (2011), 70.

Maciej Halber and Thomas Funkhouser. 2017. Fine-To-Coarse Global Registration of RGB-D Scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

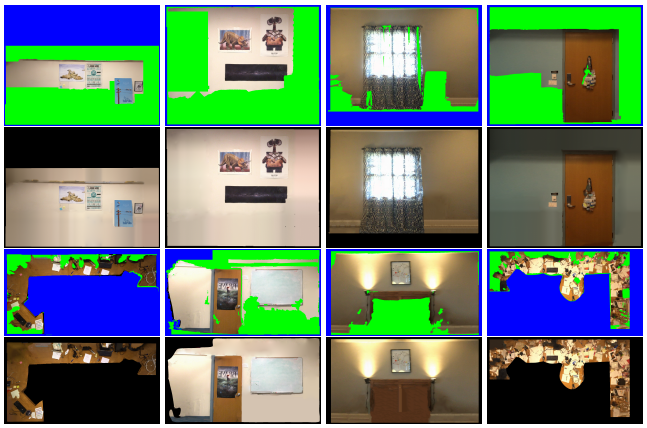


Fig. 20. Texture completion using background filling with Image Melding [Darabi et al. 2012] to synthesize missing colors. Green represents unobserved regions to be synthesized, and blue empty space.

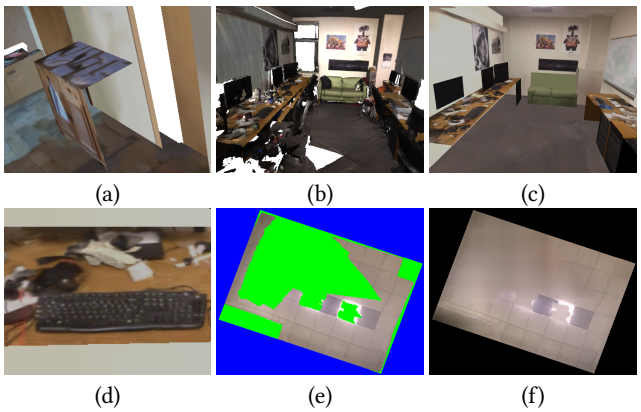


Fig. 21. Limitations of our method. (a) Geometry entirely unseen in the original scan cannot be synthesized. (b-c) Non-planar objects (e.g., some chairs) are not captured by our current primitive abstraction. (d) Small objects may be projected onto planar surfaces, as the depth camera resolution is low. (e-f) Texture inpainting can have difficulty synthesizing large missing regions.

Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. 2016. Scalable Inside-Out Image-Based Rendering. 35, 6 (2016), 231:1–231:11.

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

Andrew Edie Johnson and Sing Bing Kang. 1999. Registration and integration of textured 3D data. *Image and vision computing* 17, 2 (1999), 135–147.

Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. 2013. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *3DTV-Conference, 2013 International Conference on*. IEEE, 1–8.

David G Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60, 2 (2004), 91–110.

Peter J Neugebauer and Konrad Klein. 1999. Texturing 3d models of real world objects from multiple unregistered photographic views. In *Computer Graphics Forum*, Vol. 18. Wiley Online Library, 245–256.

Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. KinectFusion: Real-Time Dense Surface Mapping and Tracking. In *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*. IEEE, 127–136.

Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. 2013. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 169.

Eyal Ofek, Erez Shilat, Ari Rappoport, and Michael Werman. 1997. Multiresolution Textures from Image Sequences. *IEEE Comput. Graph. Appl.* 17, 2 (1997), 18–29.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. 2016. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2536–2544.

Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson image editing. In *ACM Transactions on Graphics (TOG)*, Vol. 22. ACM, 313–318.

Kari Pulli, Simo Piiroinen, Tom Duchamp, and Werner Stuetzle. 2005. Projective surface matching of colored 3D scans. In *3-D Digital Imaging and Modeling, 2005. 3DIM 2005. Fifth International Conference on*. IEEE, 531–538.

Kari Pulli and Linda G Shapiro. 2000. Surface reconstruction and display from range and color data. *Graphical Models* 62, 3 (2000), 165–201.

Claudio Rocchini, Paolo Cignoni, Claudio Montani, and Roberto Scopigno. 1999. Multiple textures stitching and blending on 3D objects. In *Rendering TechniquesâĂŹ 99*. Springer, 119–130.

François Sillion, George Drettakis, and Benoit Bodelet. 1997. Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery. In *Computer Graphics Forum*, Vol. 16. Wiley Online Library.

Denis Simakov, Yaron Caspi, Eli Shechtman, and Michal Irani. 2008. Summarizing visual data using bidirectional similarity. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 1–8.

Ioannis Stamos and PE Allen. 2000. 3-D model construction using range and image data. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, Vol. 1. IEEE, 531–536.

Shuochen Su, Mauricio Delbracio, Jue Wang, Guillermo Sapiro, Wolfgang Heidrich, and Oliver Wang. 2016. Deep Video Deblurring. *arXiv preprint arXiv:1611.08387* (2016).

Yuichi Taguchi, Yong-Dian Jian, Srikumar Ramalingam, and Chen Feng. 2013. Point-plane SLAM for hand-held 3D sensors. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 5182–5189.

Cuong T Vu, Thien D Phan, and Damon M Chandler. 2012. S3: A Spectral and Spatial Measure of Local Perceived Sharpness in Natural Images. *IEEE Transactions on Image Processing* 21, 3 (2012), 934–945.

Tuanfeng Y Wang, Hao Su, Qixing Huang, Jingwei Huang, Leonidas Guibas, and Niloy J Mitra. 2016. Unsupervised texture transfer from images to model collections. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 177.

Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. 2015. ElasticFusion: Dense SLAM without a pose graph. Robotics: Science and Systems.

Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. 2016. High-Resolution Image Inpainting using Multi-Scale Neural Patch Synthesis. *CoRR* abs/1611.09969 (2016). http://arxiv.org/abs/1611.09969

Edward Zhang, Michael F Cohen, and Brian Curless. 2016. Emptying, refurnishing, and relighting indoor spaces. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 174.

Yizhong Zhang, Weiwei Xu, Yiying Tong, and Kun Zhou. 2015. Online structure analysis for real-time indoor scene reconstruction. *ACM Transactions on Graphics (TOG)* 34, 5 (2015), 159.

Qian-Yi Zhou and Vladlen Koltun. 2014. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 155.