2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Congress Center Hamburg
Sept 28 - Oct 2, 2015. Hamburg, Germany

Optimized Color Models for High-Quality 3D Scanning

Karthik S. Narayan and Pieter Abbeel

***Abstract*— We consider the problem of estimating high-quality color models of 3D meshes, given a collection of RGB images of the original object. Applications of a database of high-quality colored meshes include object recognition in robot vision, virtual reality, graphics, and online shopping. Most modern approaches that color a 3D object model from a collection of RGB images face problems in (1) producing realistic colors for non-Lambertian surfaces and (2) seamlessly integrating colors from multiple views. Our approach efficiently solves a non-linear least squares optimization problem to jointly estimate the RGB camera poses and color model. We discover that incorporating 2D texture cues, vertex color smoothing, and texture-adaptive camera viewpoint selection into the optimization problem produces qualitatively more coherent color models than those produced by competing methods. We further introduce practical strategies to accelerate optimization. We provide extensive empirical results on the BigBIRD dataset [15], [21]: results from a user study with 133 participants indicate that on all 16 objects considered, our method outperforms competing approaches. Our code is available for download online at `http://rll.berkeley.edu/iros2015colormodels`.**

I. Introduction and Related Work

The advent of consumer-grade depth sensors such as the Kinect has led to widespread interest in 3D scanning. Shape reconstruction in particular has witnessed rich exploration. In reconstructing shape, variants on the KinectFusion algorithm have gained popularity [17], [24] due to the algorithm's realtime reconstruction ability. Researchers have also explored alternative methods, notably multiview stereo-based approaches, which particularly emphasize the importance of color and local features in reconstruction [12], [6], [5], [10]. Recent work has also proposed techniques to produce higher quality models by amalgamating information provided by both depth sensor and color information [15].

Recovering accurate color models given a shape model and a collection of color images has also been keenly explored. Accurate color models of 3D objects have been shown to play major roles in object and instance recognition, particularly for cluttered scenes [25], [22], [4], [14]. These approaches automatically annotate reconstructed shape models with color and texture features computed from calibrated RGB images. Other than robot vision, applications of high quality color models include virtual reality, computer graphics, and online shopping. In this paper, we propose a color model reconstruction method that outperforms competing methods.

Representing the reconstructed shape model as a mesh, a collection of triangles on a given vertex set, several widely used methods typically involve variants on volumetric blending, i.e., assigning an averaged color to each mesh vertex [5], [7], [17], [24], [16], [26], [23]. These approaches assume provided calibration information; in tandem with the provided mesh, these approaches establish corresponding points across the images and compute weighted averages for each mesh vertex. While some approaches advocate employing averaging with uniform weights [17], others recommend assigning greater weight to views that observe the vertex more frontally [24]. In an effort to combat specularity problems and the lack of frontal views for vertices, some

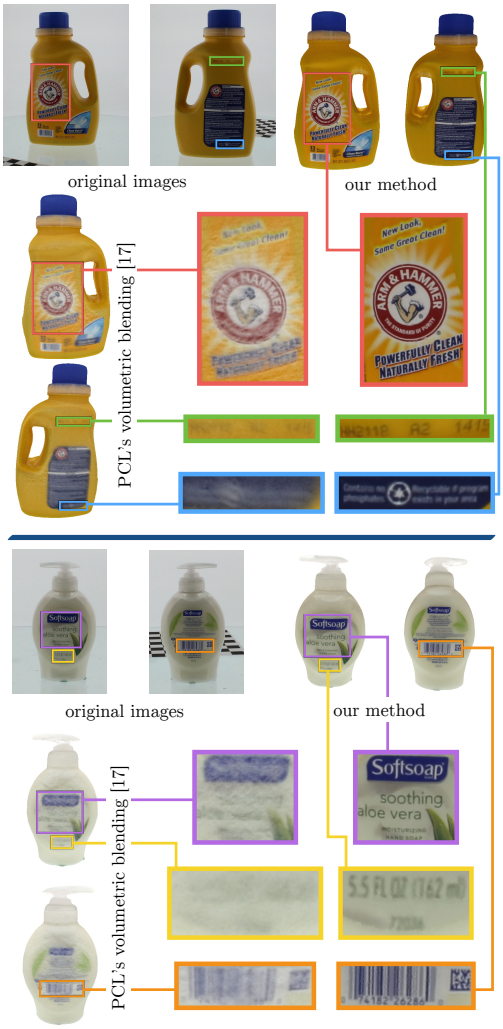


Fig. 1. Arm & Hammer Detergent (top) and Softsoap Aloe Vera (bottom), reconstructed using our method and PCL's volumetric blending [17]. The BigBIRD 3D scanning rig (Figure 8) captures 600 DSLR RGB images per object [21]; we show two per object here. Our color models can recover very fine textural details: in the Aloe Vera soap, the "5.5 FL OZ" text is 3 mm tall; the numbers under the barcode are just over 1 mm tall. PCL's volumetric blending smooths away these details.

Department of Electrical Engineering and Computer Science, University of California, Berkeley, Berkeley, CA, 94709

approaches disregard viewpoint extrinsics and instead give higher weights to views with greater color saturation [8], [5]. Given initial camera poses, a different set of methods explores how to locally optimize camera poses so as to maximize color agreement [19], [27], [18].

We jointly recover camera poses and a color model by efficiently solving a non-linear least squares optimization problem. While Zhou et. al. detail a similar approach [27], we observe that in practice (Section IV), their recovered color models suffer from specularities in RGB images and often contain ghosting and smoothed away textures. Although Hernandez et. al. [5] provide a remedy to eliminating specular highlighting artifacts, their method does poorly in recovering sharp textures.

Contributions. We demonstrate that incorporating 2D texture cues, vertex color smoothing, and texture-adaptive camera viewpoint selection into the optimization problem qualitatively ameliorates these problems. Ultimately, we demonstrate that our method produces qualitatively more coherent color models than those produced by competing methods. Results from a user study with 133 participants indicate that our method outperforms competing approaches, advancing the state of the art.

II. Problem Formulation

We take in as input a 3D mesh $\mathbf{M}$, whose representation is a collection of triangles in 3D space. Mesh $\mathbf{M}$ consists of a vertex set $\mathbf{P}$, where each $\mathbf{p} \in \mathbf{P}$ neighbors vertices $N(\mathbf{p}) \subset \mathbf{P}$. We additionally take in a collection of RGB images, $\{I_i\}$, that observe the original object; each image has an associated intrinsics matrix $\mathbf{K}_i \in \mathbb{R}^{3\times 3}$ and initial extrinsics matrix $\mathbf{T}_i^0 \in SE(3)$. For each $\mathbf{p} \in \mathbf{P}$, we wish to estimate $\vec{C}(\mathbf{p})$, vertex $\mathbf{p}$'s color.[1] The collection of vertex colors, which we denote $\mathbf{C} = \{\vec{C}(\mathbf{p})\}$, constitutes the color model we aim to learn.

We estimate the color model $\mathbf{C}$ by minimizing a non-linear least squares objective that refines the original camera extrinsics $\mathbf{T}^0 = \{\mathbf{T}_i^0\}$ so as to maximize each vertex's color agreement. Concretely, denote $V(\mathbf{p}) \subset \{I_i\}$ as the subset of images that observe vertex $\mathbf{p}$ without occlusion, $\mathbf{T}_i$ as the updated extrinsics matrix for image I_i, and $\vec{\Gamma}_i(\mathbf{p}, \mathbf{T}_i)$ as the color obtained by projecting $\mathbf{p}$ onto image I_i using extrinsics $\mathbf{T}_i$ and intrinsics $\mathbf{K}_i$. For each $I_i \in V(\mathbf{p})$, we would like the error residual $\|\vec{C}(\mathbf{p}) - \vec{\Gamma}_i(\mathbf{p}, \mathbf{T}_i)\|^2$ to be small. This reasoning suggests that we minimize the following objective:

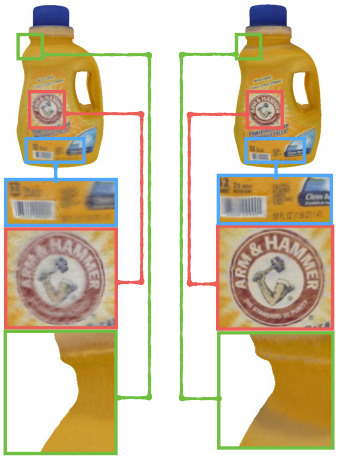
$$\mathscr{J}(\mathbf{C}, \mathbf{T}) = \frac{1}{2} \sum_{\mathbf{p}\in\mathbf{P}} \sum_{I_i \in V(\mathbf{p})} \|\vec{C}(\mathbf{p}) - \vec{\Gamma}_i(\mathbf{p}, \mathbf{T}_i)\|^2 \qquad (1)$$

where $\mathbf{T} = \{\mathbf{T}_i\}$. The only variables to minimize are $\mathbf{C}$ and $\mathbf{T}$. We compute $\vec{\Gamma}_i(\mathbf{p}, \mathbf{T}_i)$ by composing extrinsics matrix $\mathbf{T}_i$, (fixed) intrinsics matrix $\mathbf{K}_i$, and a color evaluation from I_i, which can be written as $\vec{\Gamma}_i(\mathbf{u}_i(\mathbf{g}(\mathbf{p}, \mathbf{T}_i)))$. The functions $\mathbf{g}$ and $\mathbf{u}$ are defined as:

$$\mathbf{g}(\mathbf{p}, \mathbf{T}_i) = \mathbf{T}_i \mathbf{p} \qquad (2)$$

$$\mathbf{u}([g_x, g_y, g_z, g_w]^T) = (c_x + g_x f_x / g_z, c_y + g_y f_y / g_z) \qquad (3)$$

where f_x, f_y are the focal lengths of I_i and (c_x, c_y) denotes the principal point of I_i; we obtain these values from $\mathbf{K}_i$. The function $\vec{\Gamma}_i([u_x, u_y]^T)$ computes the bilinearly interpolated intensity at coordinates (u_x, u_y) in I_i. In the following sections, we explore objective variants that successively improve reconstructed color models qualitatively. We delve into concrete optimization details in Section III.



(a) Iteration 0 (b) Iteration 200

Fig. 2. Mesh coloring (a), before, and (b), after, optimization using our implementation of the approach employed by Zhou et. al. [27]. Although this method improves texture coherence, the textures are still faded (particularly the Arm & Hammer logo). Further, solid yellow regions are still blotchy.

A. Accounting for Color Constancy and Specularities

In practice, specularities and the non-Lambertian nature of objects prevent us from achieving perfect color agreement, even after refining $\mathbf{T}$. Equation (1) asks for $\vec{C}(\mathbf{p})$ to agree with projected colors $\vec{\Gamma}_i(\mathbf{p}, \mathbf{T}_i)$ for *all* views $I_i \in V(\mathbf{p})$: as such, recovered colors in regions with white specularities may be heavily faded away, because residuals corresponding to images with high specularities will draw $\vec{C}(\mathbf{p})$ towards white. Indeed, we observe this behavior empirically. Zhou et. al. minimize a similar objective to Equation (1) [27].[2] As an example, applying this optimization problem to the Arm & Hammer detergent bottle drawn from the BigBIRD dataset [21] (692K vertices, 600 camera views), Figure 2 reveals that although the textures become clearer, they remain faded.

Rather than asking for color agreement from all views $V(\mathbf{p})$ for vertex $\mathbf{p}$, we consider selecting a subset. Images that present the most accurate colors for $\mathbf{p}$ typically have the most head-on, frontal views: i.e., the cosine of the angle subtending I_i's optical axis and $\mathbf{p}$'s normal is close to -1 (the optical axis and normal vectors point in opposite directions). We consider sorting $V(\mathbf{p})$ by this "foreshortening value"

[1] All color vectors in this paper are 4-vectors whose entries all lie between 0 and 1. The first entry is a grayscale intensity while the second through fourth are scaled RGB intensities.

[2] The primary difference is that during optimization, Zhou et. al. set color vectors to only contain grayscale values and set final RGB colors by computing a weighted average per color channel, where the weight for I_i is the cosine of the angle subtending I_i's optical axis and $\mathbf{p}$'s normal.

(a) N = 1 (b) N = 10 (c) N = 30 (c) N = 50 (d) N = 100 (e) N = 200 (f) N = 600

Fig. 3. The effect of $N = |V'(\mathbf{p};t_\mathbf{p})|$ on the sharpness of textured regions and the smoothness of non-textured regions: smaller (larger) N yield sharper (faded) textures, but blotchy (smooth) non-textured regions. When $N = 1$, we observe white regions on the detergent handle because these colors incorrectly bleed onto the white turntable background in the best viewing image; larger N rectify this problem.

and retaining at most the top N images per $\mathbf{p}$. Figure 3 summarizes the results for N ranging from 1 to 600 (total number of views) for the Arm & Hammer detergent bottle.

Interestingly, we discover that N offers a tradeoff between sharp textures and smooth non-textured regions. Shown in Figure 3, smaller N leads to rough colors on the handle while larger N smooths out this noise. For example, when $N = 1$, we observe white regions on the detergent handle because the most frontal views incorrectly bleed onto the white background in the best viewing image (see Figure 1 for sample viewing images); larger N rectify this problem. Across many objects, we found that setting N beyond 30 does not produce visibly smoother non-textured regions. As originally conjectured, smaller N leads to crisper textures while larger N leads to faded, washed out textures. Looking carefully at $N = 1$ reveals noisy boundaries for the Arm & Hammer logo while $N = 10$ provides cleaner results by averaging away this noise.

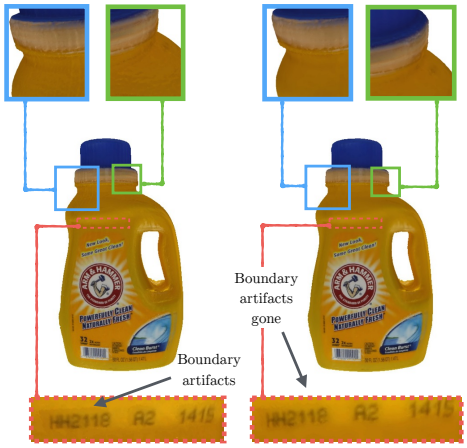
We wish to take the best of both worlds by setting N's value depending on whether $\mathbf{p}$ is considered "textured." To do this, we consider assigning a label $t_\mathbf{p}$ to each $\mathbf{p}$, where $t_\mathbf{p} = 1$ when $\mathbf{p}$ is textured and 0 otherwise. With this, we employ the objective:

$$\mathscr{J}(\mathbf{C},\mathbf{T}) = \frac{1}{2}\sum_{\mathbf{p}\in\mathbf{P}}\sum_{I_i\in V'(\mathbf{p};t_\mathbf{p})} \|\vec{C}(\mathbf{p}) - \vec{\Gamma}_i(\mathbf{p},\mathbf{T}_i)\|^2 \qquad (4)$$

where $V'(\mathbf{p};t_\mathbf{p})$ retains the top $N = 30$ views when $t_\mathbf{p} = 1$ and the top $N = 10$ views otherwise. Although interpolation between $N = 10, 30$ based on $t_\mathbf{p}$ is possible, we found that this produced less smooth color models. We discuss how to compute $t_\mathbf{p}$ in Section III-D; until then, for easier exposition, we assume that these labels have already been computed.

B. Smoothing Speckled Regions

Although adapting $N = |V'(\mathbf{p},t_\mathbf{p})|$ alleviates faded textures, we expect boundary artifacting due to the difference in the number of cameras employed in adjacent textured and untextured regions. Figure 4(a) (please use digital zoom in a PDF reader to view details) shows the colored detergent



(a) No smoothing, $\lambda = 0$ (b) Smoothing, $\lambda = 10$

Fig. 4. Detergent bottle's color model, optimized using Equation (4). Without smoothing, solid regions remain blotchy. Also, the difference in the number of cameras employed in textured and untextured regions leads to texture artifacts, as shown by the patchy "HH2118," "A2," and "1415" in (a). Optimization using Equation (5), which includes smoothing, eliminates blotchy regions while preserving textured regions; it further removes the patchy boundaries between textured and non-textured regions. Please use digital zoom to view details.

after optimizing Equation (4); the red dotted box reveals anticipated artifacts, while the blue and green boxes reveal slightly blotchy textures that still remain.

We ameliorate both problems by encouraging color agreement between pairs of vertices lying along edges of $\mathbf{M}$. We need to be careful to not blur away sharp textures, however: so, we only smooth edges where (1) both vertices are non-textured or (2) exactly one vertex is textured. The first case allows us to eliminate blotchiness while the second smoothly connects non-textured and textured vertices. Concretely, we consider optimizing the new objective

$$\begin{aligned}\mathscr{J}(\mathbf{C},\mathbf{T}) = \frac{1}{2}\sum_{\mathbf{p}\in\mathbf{P}}\sum_{I_i\in V'(\mathbf{p};t_\mathbf{p})} \|\vec{C}(\mathbf{p}) - \vec{\Gamma}_i(\mathbf{p},\mathbf{T}_i)\|^2 + \\ \frac{\lambda}{2}\sum_{\mathbf{p}\in\mathbf{P}}\sum_{\mathbf{p}'\in N(\mathbf{p})} (1 - t_\mathbf{p} t_{\mathbf{p}'}) \cdot \|\vec{C}(\mathbf{p}) - \vec{C}(\mathbf{p}')\|^2\end{aligned} \qquad (5)$$

where λ is a hyperparameter that allows us to trade off the contribution between the smoothing and original color-agreement terms. Figure 4(b) shows the detergent bottle after incorporating smoothing – shown in the blue and green boxes, non-textured regions are much smoother and the boundary artifacts have disappeared. Because we smooth only untextured regions and "texture to non-texture" boundaries, the detergent bottle's serial number is not smoothed away during optimization.

III. Optimization

We now discuss optimization algorithms to minimize the objective discussed in Section II-B. First, we discuss a naive optimization method based on the Gauss-Newton algorithm. Demonstrating that this is computationally intractable, we introduce an alternating optimization method which tractably minimizes the same objective. Section III-C concludes our discussion on optimization by discussing several practical improvements including (1) fixes to poor Hessian conditioning and (2) accelerating learning via multiscale optimization.

A. Gauss-Newton Optimization

Since $\mathscr{J}(\mathbf{C},\mathbf{T})$ is a non-linear least squares objective, we consider using the Gauss-Newton method for minimization. Equation (5) features two types of residuals:

$$\vec{r}^{(1)}_{i,\mathbf{p}} = \vec{C}(\mathbf{p}) - \vec{\Gamma}_i(\mathbf{p},\mathbf{T}_i) \tag{6}$$

$$\vec{r}^{(2)}_{\mathbf{p},\mathbf{p}'} = \vec{C}(\mathbf{p}) - \vec{C}(\mathbf{p}') \tag{7}$$

Let $\mathbf{C}^k$ and $\mathbf{T}^k$ denote the values of $\mathbf{C}$ and $\mathbf{T}$ at iteration k, and $\mathbf{x}^k = [\mathbf{C}^k,\mathbf{T}^k]$. We initialize the optimization with $\mathbf{x}^0 = [\mathbf{C}^0,\mathbf{T}^0]$ where (1) $\mathbf{T}^0$ is set to the provided initial calibrated extrinsics and (2) $\mathbf{C}^0(\mathbf{p})$ is set to the average of $\{\vec{\Gamma}_i(\mathbf{p},\mathbf{T}^0_i)\}_{I_i \in V'(\mathbf{p};t_\mathbf{p})}$. The Gauss-Newton procedure prescribes taking steps $\mathbf{x}^{k+1} = \mathbf{x}^k + \Delta\mathbf{x}^k$ where we solve for $\Delta\mathbf{x}^k$ in the following linear system:

$$\mathbf{J}^T\mathbf{J}\Delta\mathbf{x}^k = -\mathbf{J}^T\mathbf{r} \tag{8}$$

where $\mathbf{r} = [\mathbf{r}^{(1)},\mathbf{r}^{(2)}]$ is the residual vector and $\mathbf{J} = [J_{r^{(1)}},J_{r^{(2)}}]$ is the Jacobian of $\mathbf{r}$, both evaluated at $\mathbf{x}^k$:

$$\mathbf{r}^{(1)} = [\vec{r}^{(1)}_{i,\mathbf{p}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^k}]_{(i,\mathbf{p})} \tag{9}$$

$$\mathbf{r}^{(2)} = [\vec{r}^{(2)}_{i,\mathbf{p}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^k}]_{(i,\mathbf{p})} \tag{10}$$

$$\mathbf{J}_{r^{(1)}} = [\nabla\vec{r}^{(1)}_{i,\mathbf{p}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^k}]_{(i,\mathbf{p})} \tag{11}$$

$$\mathbf{J}_{r^{(2)}} = [\nabla\vec{r}^{(2)}_{i,\mathbf{p}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^k}]_{(i,\mathbf{p})} \tag{12}$$

We notice that $\mathbf{J}$ has a number of rows and columns that are both linear in $|\mathbf{P}|$; this renders solving Equation (8) intractable, since we typically operate on meshes with 100K+ vertices.

B. Alternating Optimization

We consider optimizing $\mathscr{J}(\mathbf{C},\mathbf{T})$ by alternating between minimizing $\mathbf{C}$ and $\mathbf{T}$. First, we discuss how to minimize the objective with respect to $\mathbf{C}$. We initialize the optimization method with $\mathbf{x}^0 = [\mathbf{C}^0,\mathbf{T}^0]$, computed as in Section III-A.

Optimizing C. In minimizing $\mathscr{J}(\mathbf{C},\mathbf{T})$ with respect to $\mathbf{C}$ after fixing $\mathbf{T}$, we are left with minimizing a quadratic objective. There exist many approaches to do this – we found that employing gradient descent with momentum and the adaptive learning rate method described in [9] offers a good tradeoff between speed and accuracy. Computing each gradient takes time linear in the number of edges in the mesh:

$$\nabla_{\vec{C}(\mathbf{p})}\mathscr{J}(\mathbf{C};\mathbf{T}) = \sum_{I_i \in V'(\mathbf{p};\,t_\mathbf{p})} [\vec{C}(\mathbf{p}) - \vec{\Gamma}_i(\mathbf{p},\mathbf{T}_i)] + \tag{13}$$

$$\lambda \sum_{\mathbf{p}' \in N(\mathbf{p})} (1 - t_\mathbf{p} t_{\mathbf{p}'}) \cdot [\vec{C}(\mathbf{p}) - \vec{C}(\mathbf{p}')] \tag{14}$$

Optimizing T. We minimize $\mathscr{J}(\mathbf{C},\mathbf{T})$ with respect to $\mathbf{T}$ after fixing $\mathbf{C}$ via Gauss-Newton. Ignoring terms that do not depend on $\mathbf{T}$, we rewrite Equation (5) as:

$$\mathscr{J}(\mathbf{C},\mathbf{T}) = \frac{1}{2}\sum_{I_i} \sum_{\{\mathbf{p}:I_i \in V'(\mathbf{p};\,t_\mathbf{p})\}} \|\vec{r}^{(1)}_{i,\mathbf{p}}\|^2 \tag{15}$$

Decomposing this sum of squares across all I_i, we can now compute separate Gauss-Newton updates for each image I_i, since $r_{j,\mathbf{p}}$ does not depend on $\mathbf{T}_j$ for $i \neq j$. We now discuss how to perform updates for a single image I_i. Defining $\mathbf{x}^k = [\mathbf{C}^k,\mathbf{T}^k]$ where $\mathbf{C}^k$ is fixed, we compute $\mathbf{J}$ and $\mathbf{r}$ as follows:

$$\mathbf{r} = [\vec{r}^{(1)}_{i,\mathbf{p}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^k}]_{(i,\mathbf{p})} \tag{16}$$

$$\mathbf{J} = [\nabla\vec{r}^{(1)}_{i,\mathbf{p}}(\mathbf{x})|_{\mathbf{x}=\mathbf{x}^k}]_{(i,\mathbf{p})} \tag{17}$$

We compute $\mathbf{r}$ using Equation (6). Computing $\mathbf{J}$ entails computing the partial derivatives of each entry in $\vec{r}_{i,\mathbf{p}}$ with respect to $\mathbf{T}$. For notational simplicity, let $r^{(1)}_{i,\mathbf{p}}$ denote the first entry of $\vec{r}^{(1)}_{i,\mathbf{p}}$ and $\Gamma_i(\mathbf{p},\mathbf{T}_i)$ denote the first entry of $\vec{\Gamma}_i(\mathbf{p},\mathbf{T}_i)$. We parameterize $\mathbf{T}_i$ by locally linearizing around $\mathbf{T}^k_i$; specifically, letting $\xi_i = (\alpha_i,\beta_i,\gamma_i,a_i,b_i,c_i)^T$ represent an incremental transform[3], we set:

$$\mathbf{T}_i \approx \begin{pmatrix} 1 & -\gamma_i & \beta_i & a_i \\ \gamma_i & 1 & -\alpha_i & b_i \\ -\beta_i & \alpha_i & 1 & c_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathbf{T}^k_i \tag{18}$$

We have that:

$$\nabla_{\mathbf{T}_i} r^{(1)}_{i,\mathbf{p}} = -\frac{\partial}{\partial\xi_i}(\Gamma_i(\mathbf{p},\mathbf{T}_i)) = -\frac{\partial}{\partial\xi_i}(\Gamma_i(\mathbf{u}_i(\mathbf{g}(\mathbf{p},\mathbf{T}_i)))) \tag{19}$$

$$= -\nabla\Gamma_i(\mathbf{u})\mathbf{J}_\mathbf{u}(\mathbf{g})\mathbf{J}_\mathbf{g}(\xi_i)|_{\mathbf{x}=\mathbf{x}^k} \tag{20}$$

We use Equation (18) to compute $\mathbf{J}_\mathbf{g}(\xi_i)$ and Equation (2) to compute $\mathbf{J}_\mathbf{u}(\mathbf{g})$. We evaluate $\nabla\Gamma_i(\mathbf{u})$ numerically: recall that we compute $\Gamma_i(\mathbf{u})$ via bilinear interpolation, so gradients are valid when $\mathbf{u}$ lies within I_i. After solving for $\Delta\mathbf{x}^k$, we map the resulting ξ_i back into $SE(3)$ and compute $\mathbf{T}^{k+1}_i$ via this update. By employing an alternating optimization strategy, optimizing all $\mathbf{T}$ reduces to solving a total of $|\{I_i\}|$ linear systems with 6 variables each, which we perform in parallel; Zhou et. al. employ a similar method in camera pose optimization [27].

[3]We use the bundle adjustment technique discussed in our previous work for initialization [21], so initializations of $\mathbf{T}_i$ are close to optimal – as such, incremental transforms are valid.

(a) Level 1, optimized (b) Level 2, optimized (b) Level 3, optimized

Fig. 5. Multiscale optimization: level 0 optimizes a Lindstrom-Turk-decimated mesh, level 1 optimizes the original mesh, and level 2 optimizes a $\sqrt{3}$-subdivided mesh.

(a) Before depth filtering (b) After depth filtering (c) Before SLIC smoothing (d) After SLIC smoothing

Fig. 6. (a) shows a sample response map for a single image after applying the technique in Section III-D, before depth discontinuity filtering; white regions denote higher responses. (b) shows the updated map after depth discontinuity filtering. (c) visualizes all $t_{\mathbf{p}}$, without SLIC smoothing. (d) visualizes the updated $t_{\mathbf{p}}$ after SLIC smoothing.

C. Coarse-to-Fine Levenberg-Marquardt Optimization

In making updates to each $\mathbf{T}_i$, the Hessian $\mathbf{J}^T\mathbf{J}$ may be poorly conditioned, leading to updates that cause some mesh vertices $\mathbf{p}$ to project outside the bounds of an image I_i. As a remedy, we employ damped Hessians $\mathbf{J}^T\mathbf{J}+\eta\mathbf{I}$ during optimization (a.k.a. Levenberg-Marquardt optimization). In updating a single $\mathbf{T}_i$, we first initialize η to 0. Upon making an update, we project all $\{\mathbf{p}|I_i \in V'(\mathbf{p};t_{\mathbf{p}})\}$ onto I_i; if any vertices fall outside I_i, we increase η to 0.001. We repeat this projection check and continue increasing η by a factor of 1.1 until all vertices fall within I_i; if we have not found a satisfactory update after 5 such trials, we do not update $\mathbf{T}_i$ during the current iteration.

In practice, we find that resolving small texture features such as text requires us to increase the density of vertices in $\mathbf{M}$ before optimization. As such, after reconstructing $\mathbf{M}$ using Narayan et. al.'s method [15], we employ $\sqrt{3}$-subdivision [11] without smoothing to increase the mesh's surface resolution while not altering $\mathbf{M}$'s geometry. We then apply the Levenberg-Marquardt procedure described above.

We empirically accelerate convergence without sacrificing solution quality by employing a coarse-to-fine optimization scheme. Rather than immediately use $\mathbf{M}$ in optimization, consider constructing the series of meshes: $\mathbf{M}_0, \mathbf{M}_1, \mathbf{M}_2$. Here, $\mathbf{M}_1$ is the original unaltered mesh obtained from [15] and $\mathbf{M}_2$ is obtained by a single application of $\sqrt{3}$-subdivision. In the other direction, we apply Lindstrom-Turk polygon simplification [13] to $\mathbf{M}_0$: specifically, $\mathbf{M}_{-1}$ has at most 50% the number of edges in $\mathbf{M}_0$. We use implementations for both $\sqrt{3}$-subdivision and Lindstrom-Turk polygon simplification provided in the open source Computational Geometry Algorithms Library (CGAL) [2].

We proceed by running Levenberg-Marquardt optimization on $\mathbf{M}_0$. Upon convergence, we initialize a new optimization problem on $\mathbf{M}_1$; because the vertices have changed, we re-initialize $\mathbf{C}$. However, we warm-start this new optimization problem using the converged $\mathbf{T}$ from $\mathbf{M}_0$. We repeat this up to $\mathbf{M}_2$. Figure 5 visualizes optimization progress. Multiscale optimization typically yields speedups of $2-3\times$ over directly optimizing over $\mathbf{M}_2$ (larger speedups for larger meshes); we did not find any noticeable differences between color models produced with and without multiscale optimization.

D. Texture Label Assignment

Before presenting empirical results, we discuss how to compute texture labels $t_{\mathbf{p}}$ for $\mathbf{p} \in \mathbf{P}$. Algorithm 1 provides the details. We first turn the camera images $\{I_i\}$ into grayscale images $\{I_i^{(gray)}\}$ via the luminosity method. We convolve each $I_i^{(gray)}$ with 10 kernels of size 10×10, whose entries are uniformly sampled from $[-1,1]$ and sum to 0. Taking the element-wise-maximum over these filtered images then gives us a response map, where higher responses correspond to textured regions in the image (see Figure 6(a)). We let this resulting response map be I_i'. Convolving with random filters is known to reveal high-frequency spatial patterns in images, e.g., edges [20]; as such, examining the maximum responses from an ensemble of such filters typically yields regions of high spatial frequency, i.e., textured regions.

As marked in Figure 6(a), depth-discontinuities can trigger high responses in the response map near object boundaries, which are not necessarily textured. To combat this, we (1) compute a z-buffered depth map using the intrinsics $\mathbf{K}_i$ and extrinsics $\mathbf{T}_i$ matrices associated with each image I_i and (2) compute a depth discontinuity map; a depth value is considered to be a discontinuity if, within a centered square window of 9 pixels, there is a difference in depth of more than 1 cm. We zero out entries in I_i' that are at a depth discontinuity. To average away sensor noise per image I_i, we (1) compute SLIC superpixels [3] of I_i and (2) set the value of each pixel in I_i to the average of all values in the superpixel that the pixel lies in. We apply a linear transform to ensure that entries of I_i' lie between 0 and 1. Figures 6c, d show the values of $t_{\mathbf{p}}$ with and without this noise-reduction step. The smoother consistency of Figure 6 yields color models with fewer blotchy regions.

Computing $t_{\mathbf{p}}$ entails projecting $\mathbf{p}$ onto all images where $\mathbf{p}$ is visible; let $\mathbf{P}_i \subset \mathbf{P}$ denote the subset of vertices which is visible in image I_i. We efficiently compute each $\mathbf{P}_i$ using a z-buffer technique; additionally, vertices within 9 pixels of a depth discontinuity are discarded from $\mathbf{P}_i$. We proceed by projecting $\mathbf{p}$ onto each response map I_i' for which $\mathbf{p} \in \mathbf{P}_i$, accumulating the lookup values into a list. We set $t_{\mathbf{p}}$ to a weighted mean of this list of values. The weight for image I_i is the absolute value of the cosine of the angle subtending image I_i's optical axis and vertex $\mathbf{p}$'s normal. We finally assign $t_{\mathbf{p}}$ to 0 or 1 by simply rounding its value.

Fig. 7. Juxtapositions of original BigBIRD images [21] with color model reconstructions from (1) our method, (2) Zhou et. al. [27] without deformation grid optimization, (3) Hernandez et. al. [8], and (4) PCL's volumetric blending [17]. Note that we do not employ Zhou et. al.'s deformation grid optimization, since this leads to divergence. Please use a PDF reader's digital zoom to view details.

Algorithm 1 Compute Texture Intensities

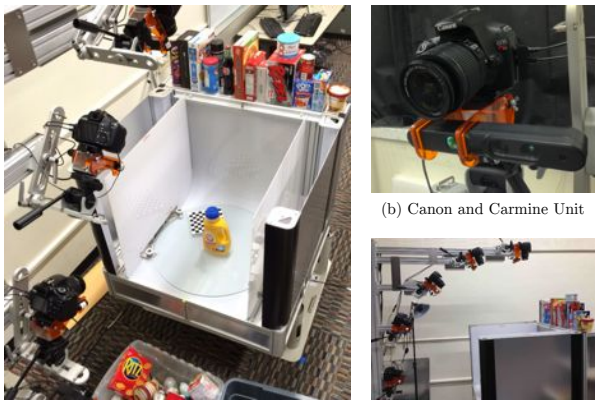
Require: Mesh $\mathbf{M}$ with vertex set $\mathbf{P}$, calibrated grayscale images $\{I_i^{(gray)}\}$, associated intrinsics $\{\mathbf{K}_i\}$ and extrinsics $\{\mathbf{T}_i\}$ matrices
  $F \leftarrow$ a list of 10 kernels of size 10×10, whose entries are uniformly sampled from $[-1,1]$ and sum to 0
  $I' \leftarrow \{\}$
  for $I_i^{(gray)} \in \{I_i^{(gray)}\}$, **in parallel, do**
    $I_i' \leftarrow$ matrix of zeros, with size of I_i
    for $f \in F$ **do**
      $C \leftarrow$ convolve f with $I_i^{(gray)}$
      $I_i' \leftarrow$ element-wise-max of I_i' and C
    end for
    $Z \leftarrow$ depth map for $I_i^{(gray)}$, computed with $\mathbf{M}, \mathbf{K}_i, \mathbf{T}_i$.
    $D \leftarrow$ depth discontinuity map computed from Z (see Section III-D)
    $I_i' \leftarrow$ zero I_i' where depth discontinuities exist in D.
    $I_i' \leftarrow (I_i' - \min(I_i'))/(\max(I_i') - \min(I_i'))$
    $S \leftarrow$ compute SLIC superpixels for I_i
    for $s \in S$ **do**
      $p \leftarrow$ average value of I_i' in superpixel s
      Replace all values in I_i' corresponding to pixels in s with value p
    end for
    $I'[i] \leftarrow I_i'$
  end for
  $t \leftarrow \{\}$
  for $\mathbf{p} \in \mathbf{P}$, **in parallel, do**
    $v \leftarrow$ list of values obtained by projecting $\mathbf{p}$ onto each I_i' where $\mathbf{p}$ is visible in I_i'
    $t_{\mathbf{p}} \leftarrow$ weighted mean of values in v (see Section III-D for weights)
    $t_{\mathbf{p}} \leftarrow 0$ if $t_{\mathbf{p}} \leq 0.5$, otherwise 1
  end for
  return t

IV. Experiments

We conduct experiments using objects drawn from the BigBIRD dataset, which we discussed in our previous work [21]. The dataset consists of high-quality scanned data for 125 household objects. Described in [21], 5 high resolution (12.2 MP) Canon Rebel T3 cameras and 5 PrimseSense Carmine 1.09 short-range depth sensors comprise our setup. We mounted each Carmine to a T3 using a platform from RGBDToolkit [1] and each T3 to an arm, the Ortery MultiArm 3D 3000 (Figures 8b, c).

All experiments are run on an Intel i7-4930K with 64 GB of memory. We provide timing information on the source code page (see abstract); in general, the optimization process typically takes 1-5 minutes per object, depending on the number of object vertices.

Data collection Scanning a single object consists of placing an object on the Ortery Photobench 260 turntable (Figure 8a) and running a UNIX command; otherwise, the process is automated and takes 5 minutes to scan a single object. As the turntable rotates in increments of 3 degrees, each of the Canon Rebel T3/Carmine units captures RGB-D images. We ultimately capture 600 point clouds, high-resolution and low-resolution RGB images from 5 polar $\times 120$ azimuthal views.



(a) Ortery Photobench, Perspective (b) Canon and Carmine Unit (c) Ortery Photobench, Side

Fig. 8. Our scanning setup consisting of (a), (c) the Ortery Photobench and (b) 5 Canon Rebel t3i + Carmine 1.09 units mounted together. We described a method to jointly calibrate all 10 sensors in [21].

Calibration Calibrating the sensors involves placing a chessboard in multiple orientations, detecting chessboard corners, and running a bundle adjustment optimizer [21].

We display 64 reconstructed color models in Figure 7: 16 objects $\times 4$ methods we consider: (1) ours, (2) Zhou et. al's [27], (3) Hernandez et. al.'s [8], and (4) PCL's volumetric blending [17]. We do not employ the deformation grid optimization that Zhou et. al. discuss, as the resulting LM updates are not small, leading to divergence.

A. Evaluation Methodology

To quantitatively compare our method with competing approaches, we conducted an online user survey (`https://goo.gl/forms/Feo2OqOdw4`), where each participant was given 16 multiple choice questions, one per object in Figure 7. Each question asked the participant: "Which of the following images matches 'Reference' most closely?" The reference image displayed the original BigBIRD image; we juxtaposed the reference image with color models estimated from our method and competing methods. We randomized the order in which the color models appeared within a question. Participants were allowed to answer with a tie, electing two methods as the best for a question. Participants were given as much time as needed to complete the 16 questions; we advertised the survey primarily via department emails. There were a total of 133 participants in the survey.

In creating Table I, for each question asked per participant, we assigned 1 point for a single elected method and 0.5 point to two elected methods when the participant chose a tie. To ensure that we did not receive overwhelming amounts of spurious data, we included two objects whose color models were difficult to distinguish in quality – O13 (cholula_chipotle_hot_sauce) and O15 (white_rain_sensations_apple_blossom); in the former case, our method was voted the best by 49.1% of participants

User Study Summary ($n = 133$): Which Method Matches the Reference Most Closely?																
	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	O15	O16
Our Method	**0.872**	**0.841**	**0.882**	**0.985**	**0.587**	**0.735**	**0.855**	**0.655**	**0.802**	**0.774**	**0.859**	**0.925**	**0.491**	**0.894**	**0.581**	**0.917**
Zhou et. al. [27]	0.005	0.019	0.000	0.008	0.049	0.027	0.021	0.031	0.054	0.117	0.080	0.016	0.019	0.033	0.018	0.004
Hernandez et. al. [8]	0.106	0.111	0.109	0.008	0.320	0.239	0.119	0.304	0.126	0.100	0.044	0.055	0.472	0.065	0.353	0.063
Volumetric blending [17]	0.018	0.029	0.009	0.000	0.044	0.000	0.004	0.010	0.018	0.009	0.016	0.004	0.019	0.008	0.048	0.016

TABLE I

O1 THROUGH O16 REPRESENT OBJECTS IN FIGURE 7: 3M_HIGH_TACK_SPRAY_ADHESIVE IS O1, WINDEX IS O2, ETC.

while Hernandez et. al.'s method received a close 47.2%. In the latter case, our method received 58.1% while Hernandez et. al.'s received 47.2%. Most of the other color models had fairly distinguishing features.

B. Analysis

Our method received the highest votes in all objects we considered, in most cases, by more than 30%, and often by more than 75%. Because Zhou et. al's method averages in all images that view a vertex [27], it provides ghosted, faded textures. Examples include front faces for pop_secret_butter and mom_to_mom_butternut_squash_pear as well as caps for pepto_bismol and listerine_green. Other objects with Zhou et. al. are often faded, as seen with the 3m_high_tack_spray_adhesive and v8_fusion_peach_mango. PCL's volumetric blending [17] suffers from similar problems, although in most cases, ghosting is more severe.

Per vertex, Hernandez et. al.'s approach averages the top 3 camera views that have the highest saturation on an HSV scale; they do not jointly optimize camera poses [8]. This model can produce vibrant, but highly distorted color models. Examples include front faces for pop_secret_butter, detergent, crystal_hot_sauce, and palmolive_green and caps for listerine_green, pepto_bismol, and mom_to_mom_butternut_squash_pear.

Despite the highly specular nature of our objects, our approach can reconstruct very fine textural details. In pepto_bismol, we clearly see the "digestive relief" text on the cap. In coca_cola_glass_bottle, we observe that Coca Cola created the bottle on 15 Sep 14 at 7:40 AM.[4] In listerine_green, the white plastic holding down the cap as well is crisp, as is the text "listerine" on the black cap.

Our method is not perfect. v8_fusion_peach_mango reveals blotchiness above the V8 symbol. palmolive_green shows ghosting around the green logo. detergent's barcode is not perfectly crisp and has few splotches of yellow.

V. CONCLUSION

We recover high-quality color models from the BigBIRD dataset by jointly optimizing a non-linear least squares objective over camera poses and a mesh color model. We incorporate 2D texture cues, vertex color smoothing, and texture-adaptive camera viewpoint selection into the objective, which allows us to outperform competing methods. We also discuss strategies to accelerate optimization speeds.

[4]Due to file size limits, to clearly see such tiny details, we recommend visiting the online survey: `https://goo.gl/forms/Feo2OqOdw4`

VI. ACKNOWLEDGEMENTS

KN was supported by an NDSEG Fellowship. This research was funded in part by ONR through a Young Investigator Program award. We thank NVIDIA for donating a Tesla K40 GPU. We thank Animesh Garg for useful discussions.

REFERENCES

[1] Rgbdtoolkit. http://www.rgbdtoolkit.com.
[2] CGAL. http://www.cgal.org.
[3] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. Slic superpixels. Technical report, 2010.
[4] A. Aldoma, F. Tombari, J. Prankl, A. Richtsfeld, L. Di Stefano, and M. Vincze. Multimodal cue integration through hypotheses verification for rgb-d object recognition and 6dof pose estimation. In *ICRA*, 2013.
[5] Hernández C.E. and F. Schmitt. Silhouette and stereo fusion for 3d object modeling. *CVII*, 2004.
[6] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *PAMI*, 2010.
[7] Ran Gal, Yonatan Wexler, Eyal Ofek, Hugues Hoppe, and Daniel Cohen-Or. Seamless montage for texturing models. In *CGF*, 2010.
[8] C. Hernández. Stereo and silhouette fusion for 3d object modeling from uncalibrated images under circular motion. *Doctoral thesis*, 2004.
[9] R.A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295–307, 1988.
[10] M. Jancosek and T. Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR*, pages 3121–3128, 2011.
[11] L. Kobbelt. Sqrt 3-subdivision. In *CGIT*, 2000.
[12] S. Lazebnik, Y. Furukawa, and J. Ponce. Projective visual hulls. *IJCV*, 2007.
[13] P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. In *Visualization'98.*, 1998.
[14] M. Martinez, A. Collet, and S. Srinivasa. Moped: A scalable and low latency object recognition and pose estimation system. In *ICRA*, 2010.
[15] K. S. Narayan, J. Sha, A. Singh, and P. Abbeel. Range sensor and silhouette fusion for high-quality 3d scanning. In *ICRA*, 2015.
[16] P. J. Neugebauer and K. Klein. Texturing 3d models of real world objects from multiple unregistered photographic views. In *CGF*, 1999.
[17] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, pages 127–136, 2011.
[18] K. Pulli, S. Piiroinen, T. Duchamp, and W. Stuetzle. Projective surface matching of colored 3d scans. In *3-DIM*, 2005.
[19] K. Pulli and L. G. Shapiro. Surface reconstruction and display from range and color data. *Graphical Models*, 2000.
[20] A. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng. On random weights and unsupervised feature learning. In *ICML*, 2011.
[21] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel. Bigbird: A large-scale 3d database of object instances. In *ICRA*, 2015.
[22] J. Tang, S. Miller, A. Singh, and P. Abbeel. A textured object recognition pipeline for color and depth image data. In *ICRA*, 2012.
[23] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald. Robust real-time visual odometry for dense rgb-d mapping. In *ICRA*, 2013.
[24] T. Whelan, M. Kaess, M.F. Fallon, H. Johannsson, J.J. Leonard, and J.B. McDonald. Kintinuous: Spatially extended KinectFusion. In *RSS-W on RGB-D: Advanced Reasoning with Depth Cameras*, 2012.
[25] Z. Xie, A. Singh, J. Uang, K. S. Narayan, and P. Abbeel. Multimodal blending for high-accuracy instance recognition. In *IROS*, 2013.
[26] H. Yamauchi, H. Lensch, J. Haber, and H. Seidel. Textures revisited. *The Visual Computer*, 2005.
[27] Q. Zhou and V. Koltun. Color map optimization for 3d reconstruction with consumer depth cameras. *ACM TOG*, 2014.