Shape, Illumination, and Reflectance from Shading Supplementary Material

Jonathan T. Barron, *Member, IEEE,* and Jitendra Malik, *Fellow, IEEE*

1 Linearization and Rendering

Here we will detail how to calculate $S(Z, L)$ (the log-shading of some depth map Z subject to some spherical harmonic illumination L) and its analytical derivative efficiently, for the purpose of calculating A and backpropagating losses on A back onto Z . First, we convert Z into a set of surface normals:

$$N^x = \frac{Z * h_3^x}{B}, \quad N^y = \frac{Z * h_3^y}{B}, \quad N^z = \frac{1}{B}$$
$$B = \sqrt{1 + (Z * h_3^x)^2 + (Z * h_3^y)^2} \tag{1}$$

where $*$ is convolution. We also compute the following:

$$\begin{aligned} F_{11} &= (1 - N^x \times N^x) \times N^z \\ F_{22} &= (1 - N^y \times N^y) \times N^z \\ F_{13} &= -(N^x \times N^z \times N^z) \\ F_{23} &= -(N^y \times N^z \times N^z) \\ F_{12} &= -(N^x \times N^y \times N^z) \end{aligned} \tag{2}$$

where $\times$ is component-wise multiplication of two images. Let us look at the surface normal at one pixel: $\mathbf{n}_i = [N_i^x, N_i^y, N_i^z]^{\mathrm{T}}$. Rendering that point with spherical harmonics is:

$$S(\mathbf{n}_i, L) = [\mathbf{n}_i; 1]^{\mathrm{T}} \mathrm{M} [\mathbf{n}_i; 1] \tag{3}$$
$$\mathrm{M} = \begin{bmatrix} c_1 L_9 & c_1 L_5 & c_1 L_8 & c_2 L_4 \\ c_1 L_5 & -c_1 L_9 & c_1 L_6 & c_2 L_2 \\ c_1 L_8 & c_1 L_6 & c_3 L_7 & c_2 L_3 \\ c_2 L_4 & c_2 L_2 & c_2 L_3 & c_4 L_1 - c_5 L_7 \end{bmatrix}$$

$$c_1 = 0.429043 \quad c_2 = 0.511664$$
$$c_3 = 0.743125 \quad c_4 = 0.886227 \quad c_5 = 0.247708$$

Note that $S(\mathbf{n}_i, L)$ is the *log*-shading at pixel i, not the shading. This is different from the traditional usage of spherical harmonic illumination. Directly modeling log-shading makes optimization easier by guaranteeing that shading is greater than 0 without needing to clamp shading at 0, as is normally done. The gradient of the log-shading at this point with respect to the surface normal is:

$$B_i = \nabla_{\mathbf{n}_i} S(\mathbf{n}_i, L) = 2\mathbf{n}_i^{\mathrm{T}} \mathrm{M}[:, 1:3]$$

Where B is a three-channel image, where B^x is the gradient of S with respect to N^x, etc. Given the log-shading, we can infer what the log-albedo at this point must be:

$$A_i = I_i - S(\mathbf{n}_i, L) \tag{4}$$

After calculating $g(A)$ and $\nabla_A g(A)$, we can backpropagate the gradient onto Z as follows:

$$\begin{aligned} D_S &= -\nabla_A g(A) \\ D_x &= B^x \times F_{11} + B^y \times F_{12} + B^z \times F_{13} \\ D_y &= B^x \times F_{12} + B^y \times F_{22} + B^z \times F_{23} \\ \nabla_Z g(A) &= (D_S \times D_x) \star h_3^x + (D_S \times D_y) \star h_3^y \end{aligned} \tag{5}$$

where $\times$ is component-wise multiplication of two images and $\star$ is cross-correlation.

Let us construct the matrix J, the Jacobian matrix of all partial derivatives of S with respect to L, which is a n by 9 matrix (where n is the number of pixels in S), where row i is:

$$J_i = [c4, 2c_2 N_i^y, 2c_2 N_i^z, 2c_2 N_i^x, 2c_1 N_i^x N_i^y, 2c_1 N_i^y N_i^z, c_3 N_i^z N_i^z - c_5, 2c_1 N_i^x N_i^z, c_1 (N_i^x N_i^x - N_i^y N_i^y)]$$

We can use this matrix to backpropagate the gradient of the loss with respect to S onto L, as follows:

$$\nabla_L g(A) = J^{\mathrm{T}} D_S \tag{6}$$

We have described how to linearize a depth map, compute a log-shading image of that linearization with respect to a grayscale spherical-harmonic model of illumination, and backpropagate a gradient with respect to that shading image onto the depth map and the illumination model. To do the same for a color

• *J.T. Barron and J. Malik are with the Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA 94720.*
E-mail: barron,malik@eecs.berkeley.edu

image, we simply do the same procedure three times — though for efficiency's sake we need only linearize the depth map once.

2 Efficient Quadratic Entropy

Here we will detail a novel method for calculating the quadratic entropy measure introduced in [1], which we use in our parsimony prior on log-reflectance. Let $\mathbf{x}$ be a vector, N is the length of $\mathbf{x}$, and σ is the bandwidth parameter (the width of the Gaussian bump around each element of $\mathbf{x}$). Then the quadratic entropy of $\mathbf{x}$ under the Parzen window defined by $\mathbf{x}$ and σ is defined as:

$$\begin{aligned} H(\mathbf{x}) &= -\log\left(\frac{1}{Z}\sum_{i=1}^{N}\sum_{j=1}^{N}\exp\left(-\frac{(\mathbf{x}_i-\mathbf{x}_j)^2}{4\sigma^2}\right)\right) \\ Z &= N^2\sqrt{4\pi\sigma^2} \end{aligned} \tag{7}$$

Note that we will use $H(\cdot)$ here to describe entropy, rather than mean curvature. Our first insight is that this can be re-expressed as a function on a histogram of $\mathbf{x}$. Let W be the bin-width of the histogram of $\mathbf{x}$, let M be the number of bins, and let $\mathbf{n}_a$ be the count of $\mathbf{x}$ in bin a. Then:

$$H(\mathbf{n}) = -\log\left(\sum_{a=1}^{M}\mathbf{n}_a\sum_{b=1}^{M}\frac{\mathbf{n}_b}{Z}\exp\left(-\frac{W^2(a-b)^2}{4\sigma^2}\right)\right) \tag{8}$$

Though the computation complexity of this formulation is still quadratic with respect to M, if the histogram is constructed such that many datapoints fall in the same bin this formulation can be much more efficient in practice. Our second insight is that this can be expressed as a convolution of $\mathbf{n}$ with a small Gaussian filter. Let $\mathbf{g}$ be a Gaussian filter:

$$\mathbf{g}_d = \frac{1}{Z}\exp\left(-\frac{W^2d^2}{4\sigma^2}\right) \tag{9}$$

Where d is distance from the center. With this, we can rewrite $H(\mathbf{n})$ as follows:

$$H(\mathbf{n}) = -\log\left(\mathbf{n}^{\mathrm{T}}(\mathbf{n}*\mathbf{g})\right) \tag{10}$$

Where $*$ is convolution. This quantity is extremely efficient to compute, provided that the lengths of $\mathbf{n}$ and $\mathbf{g}$ are small, which is true provided that the range of $\mathbf{x}$ is not much larger than σ, which is generally true in practice.

This formulation also allows us to easily compute the gradient of $V(\mathbf{n})$ with respect to $\mathbf{n}$:

$$\nabla H(\mathbf{n}) = \left(\frac{-2}{\mathbf{n}^{\mathrm{T}}(\mathbf{n}*\mathbf{g})}\right)(\mathbf{n}*\mathbf{g}) \tag{11}$$

Histogramming is a non-smooth operation, making this approximation to entropy not differentiable with respect to $\mathbf{x}$. However, if instead of standard histogramming we use linear interpolation to construct $\mathbf{n}$, then the gradient with respect to $\mathbf{x}$ is non-zero and can be calculated easily.

Let R_L and R_U define the bounds on the range of the bins, with $R_U = \min(\mathbf{x})$ and $R_L = \max(\mathbf{x})$. The fenceposts assigned to datapoint x_i are b_L and b_U, where b_L is the largest fencepost below it, and b_U is the smallest fencepost above it:

$$b_L = \lfloor (x_i - R_L)/W \rfloor, \quad b_U = b_L + 1 \tag{12}$$

x_i will be assigned to those bins according to these weights:

$$w_L = (x_i - b_L)/W, \quad w_U = 1 - w_L \tag{13}$$

When adding x_i to the histogram, we just add these two weights to the appropriate bins:

$$\mathbf{n}_L = \mathbf{n}_L + w_L, \quad \mathbf{n}_U = \mathbf{n}_U + w_U \tag{14}$$

The partial derivatives of the histogram with respect to x_i are simple:

$$\frac{\partial \mathbf{n}_L}{\partial x_i} = -\frac{1}{W}, \quad \frac{\partial \mathbf{n}_U}{\partial x_i} = \frac{1}{W} \tag{15}$$

With this, we can construct the Jacobian J of $\mathbf{n}$ with respect to $\mathbf{x}$, which is a M by N sparse matrix. With this, we can calculate the gradient of H with respect to $\mathbf{x}$:

$$\nabla H(\mathbf{x}) \approx J^{\mathrm{T}}\nabla H(\mathbf{n}) \tag{16}$$

This approximation to quadratic entropy is, in practice, extremely efficient and extremely accurate. Other techniques exist for computing approximations to this quantity, most notably the fast Gauss transform and the improved fast Gauss transform. Also, $H(\mathbf{x})$ could be computed exactly using the naive formulation in Equation 7. The naive formulation is completely intractable, as the computation complexity is $O(N^2)$. The FGT-based algorithms are $O(N \log N)$, and provide no efficient way to compute $\nabla H(\mathbf{x})$, which makes those algorithms impossible to use in our gradient-based optimization scheme. Our approximation has a complexity of $O(N)$ (provided the kernel in the convolution is small) and allows for $\nabla H(\mathbf{x})$ to be approximated extremely efficiently. In practice, our model produces approximations of entropy that are usually within 0.01% of the true entropy, which is similar to the accuracy obtained using the fast Gauss transform or the improved fast Gauss transform, and is 10 or 100 times faster than the FGT-based algorithms.

This techniques for computing quadratic entropy for a univariate signal can easily be generalized to higher dimensions. We use a three-dimensional generalization to compute the quadratic entropy of a color (whitened) log-reflectance image. Instead of constructing a 1D histogram with linear interpolation, we construct a 3D histogram using trilinear interpolation, and instead of convolving our 1D kernel with a

Gaussian filter, we convolve the 3D histogram with three separable Gaussian filters.

Note that this formulation is extremely similar to the bilateral grid [2], which is a tool for high-dimensional Gaussian filtering (used mostly for bilateral filtering, hence the name). The calculation of our entropy measure is extremely similar to the "splat, blur, slice" pipeline in other high-dimensional Gaussian filtering works [3], except that after the "slice" operation we take the inner product of the input "signal" and the blurred output signal. This means that we need not actually compute the slice operation, but can instead just compute the inner product directly in the histogram space. This connection means that the body of work for efficiently computing this quantity in the context of image filtering can be directly adapted to the problem of computing high-dimensional entropy measures. Recent work [3] suggests that for dimensionalities of 3, our bilateral grid formulation is the most efficient among existing techniques, but that this entropy measure could be computed reasonably efficiently in significantly higher-dimensionality spaces (up to 8 or 16) using more sophisticated techniques.

3 Mean Curvature

Here we will detail how to calculate $H(Z)$ (the mean curvature of a depth map Z, not entropy) and its analytical derivative efficiently. Mean curvature on a surface is a function of the first and second partial derivatives of that surface.

$$H(Z) = \frac{\left(1+Z_x^2\right) Z_{yy} - 2Z_x Z_y Z_{xy} + \left(1+Z_y^2\right) Z_{xx}}{2\left(1+Z_x^2+Z_y^2\right)^{3/2}} \quad (17)$$

To calculate this for a discrete depth map, we will first approximate the partial derivatives using filter convolutions.

$$Z_x = Z * h_3^x, \quad Z_y = Z * h_3^y \quad (18)$$
$$Z_{xx} = Z * h_3^{xx},\ Z_{yy} = Z * h_3^{yy},\ Z_{xy} = Z * h_3^{xy}$$
$$h_3^x = \tfrac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad h_3^y = \tfrac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$
$$h_3^{xy} = \tfrac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \quad h_3^{yy} = \tfrac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ -2 & -4 & -2 \\ 1 & 2 & 1 \end{bmatrix}, \quad h_3^{xx} = \tfrac{1}{4}\begin{bmatrix} 1 & -2 & 1 \\ 2 & -4 & 2 \\ 1 & -2 & 1 \end{bmatrix}$$

We then compute the following intermediate "images", and use them to compute $H(Z)$.

$$\begin{aligned} M &= \sqrt{1+Z_x^2+Z_y^2} \\ N &= (1+Z_x^2)Z_{yy} - 2Z_xZ_yZ_{xy} + (1+Zy^2)Z_{xx} \\ D &= 2M^3 \\ H(Z) &= N/D \end{aligned} \quad (19)$$

When computing $H(Z)$, we also compute the following, which are stored until after the loss function with respect to $H(Z)$ has been calculated, at which point they will be used to backpropagate the gradient of the loss function using the chain rule.

$$\begin{aligned} F_x &= 2(Z_xZ_{yy} - Z_{xy}Z_y) - \frac{3Z_xN}{M^2} \\ F_y &= 2(Z_{xx}Z_y - Z_xZ_{xy}) - \frac{3Z_yN}{M^2} \\ F_{xx} &= 1+Z_y^2 \\ F_{yy} &= 1+Z_x^2 \\ F_{xy} &= -2Z_xZ_y \end{aligned} \quad (20)$$

Given $f(H(Z))$ and $\nabla_{H(Z)} f$, a loss function and the gradient of that loss function with respect to $H(Z)$, we can calculate $\nabla_Z f$, the gradient of the loss with respect to Z, as follows:

$$\begin{aligned} B &= \frac{\nabla_{H(Z)} f}{D} \\ \nabla_Z f &= (BF_x) \star h_3^x + (BF_y) \star h_3^y \\ &+ (BF_{xx}) \star h_3^{xx} + (BF_{yy}) \star h_3^{yy} + (BF_{xy}) \star h_3^{xy} \end{aligned} \quad (21)$$

Adjacent variables are component-wise multiplication of two images, / is component-wise division, $*$ is convolution and $\star$ is cross-correlation.

4 Noisy Shape Observation

One of the reasons that using shading cues to recover shape (as we are attempting here) is challenging, is that shading is a fundamentally poor cue for low-frequency (coarse) shape variation. Shading is directly indicative of only the shape of a point relative to its neighbors: fine-scale variations in shape produce sharp, localized changes in an image, while coarse-scale shape variations produce very small, subtle changes across an entire image. Both algorithms [4] and humans [5] therefore make errors in estimating coarse depth when using only shading. Bas relief sculptures take advantage of this by conveying the impression of a rich, deep 3D scene, using only the shading produced by a physically shallow object.

To deal with this issue, we will construct our prior on shape to allow for an external observation of shape to be incorporated into inference. This observation may be produced by a stereo algorithm, or by some depth sensor such as a laser rangefinder or the Kinect. These depth sensors or stereo algorithms often produce depth maps which are noisy or incomplete, or most often blurry — lacking fine-scale shape detail. Because of the complementary strengths of stereo and shading, combining the two can often yield very accurate results [6], [7].

We will construct a loss function to encourage our recovered depth Z to resemble the raw sensor depth $\hat{Z}$:

$$f_o(Z, \hat{Z}) = \sum_i \left(((Z * b(\sigma_Z))_i - \hat{Z}_i)^2 + \epsilon^2 \right)^{\frac{\gamma_o}{2}} \quad (22)$$

This is simply a hyperlaplacian distribution with an exponent of γ_o on the difference between $(Z * b(\sigma_Z))$

and $\hat{Z}$ at every pixel, with ϵ added in to make the loss differentiable everywhere. $b(\sigma_Z)$ is a 2D Gaussian filter with a standard deviation of σ_Z, and $*$ is convolution, so $(Z * b(\sigma_Z))_i$ is the value of a blurry version of our shape estimate Z at pixel location i. We tune γ_o on the training set, which sets it to ~ 1, and we set $\epsilon = 1/100$. The robust nature of this cost encourages Z to resemble $\hat{Z}$, while allowing it to occasionally differ drastically. In our experiments we use $Z^* * b(30)$ as our $\hat{Z}$, which is a reasonably proxy for a stereo algorithm or low-resolution depth-sensor, and we set $\sigma_Z = 30$ as that value (unsurprisingly) performs best during cross-validation.

For the one variant of our model in which we incorporate a noisy external shape observation, our prior on shapes gains an additional term, and becomes:

$$f(Z) = \lambda_k f_k(Z) + \lambda_i f_i(Z) + \lambda_c f_c(Z) + \lambda_o f_o(Z, \hat{Z}) \quad (23)$$

Where λ_o is cross-validated on the training set.

5 Efficient Computation

Our model is fairly computationally expensive. Evaluating our loss function and its gradient takes close to a second, and optimization requires that the loss be evaluated hundreds of times. To make this model more tractable, we use some additional tricks to speed up the computation of the loss function.

First, our smoothness priors for reflectance and shape require repeatedly computing the negative log-likelihood of a Gaussian scale mixture. Computing this naively is very expensive, but it can be made extremely efficient by pre-computing a lookup table of the negative log-likelihood, and indexing into that to compute the gradient and its loss. For the multivariate GSM used in our smoothness prior for color reflectance, we can construct a lookup table of negative log-likelihood with respect to Mahalanobis distance under the covariance matrix Σ in our GSM.

When computing our smoothness priors, it's often fastest to pre-compute the pairs of pixels within all 5×5 windows, and construct a sparse matrix where for each pair, we have a row in which the column corresponding to one pixel in the pair is set to 1 and the column corresponding to the other pixel is set to -1. With this, a vector of pairwise distances between pixels can be computed efficiently with one sparse matrix-vector product. Also, expressing this pairwise distance computation as a matrix multiplication allows gradients to be easily backpropagated from the vector of differences onto the raw pixels by simply multiplying the gradient vector by the transpose of this matrix.

The prior for absolute reflectance can be computed efficiently using the same bilateral-grid trick used for entropy: splat the signal into a histogram, compute the loss of the histogram, and then backpropagate onto the data. For even more efficiency, we can use the same histogram for both the entropy prior and the absolute prior, which means that for each evaluation of the loss function, we only need to compute one histogram from the reflectance and one backpropagation from the histogram to the reflectance.

6 Error Metrics

Choosing good error metrics for this task is challenging. We will use the geometric mean of six error metrics: two for shape, one for illumination, one for shading, one for reflectance, and the MIT intrinsic images error metric introduced in [8], which we will refer to as RS-MSE (though which the original authors call "LMSE"). We use the geometric mean of these metrics as it is insensitive to the different dynamic ranges of the constituent error metrics, and is difficult to trivially minimize in practice.

Our first shape error metric is:

$$Z\text{-MAE}(\hat{Z}, Z^*) = \frac{1}{n} \min_{\beta} \sum_{x,y} \left| \hat{Z}_{x,y} - Z^*_{x,y} + b \right| \quad (24)$$

This is the shift-invariant absolute error between the estimated shape $\hat{Z}$ and the ground-truth shape Z^*. This error metric is sensitive to all errors in shape estimation, except for the absolute distance of the shape from the viewer (which is unknowable under orthographic projection). It can be computed efficiently by setting b to the median of $\hat{Z} - Z^*$.

Our second shape error metric is:

$$N\text{-MAE}(\hat{N}, N^*) = \frac{1}{n} \sum_{x,y} \arccos\left(\hat{N}_{x,y} \cdot N^*_{x,y}\right) \quad (25)$$

This is the mean error between the normal field $\hat{N}$ of our estimated shape $\hat{Z}$ and the normal field N^* of the ground-truth shape Z^*, in radians. This metric is most sensitive to very fine-scale errors in $\hat{Z}$, which is what determines surface orientation.

For illumination, our error metric is:

$$L\text{-MSE}(\hat{L}, L^*) = \frac{1}{n} \min_{\alpha} \sum_{x,y} ||\alpha V(\hat{L})_{x,y} - V(L^*)_{x,y}||_2^2 \quad (26)$$

Which is the scale-invariant MSE of a rendering of our recovered illumination $\hat{L}$ and the ground-truth illumination L^*. $V(L)$ is a function that renders the spherical harmonic illumination L on a sphere and returns the log-shading. $V(L)_{x,y}$ is a 3-vector of log-RGB at position (x, y) in the renderings. The α multiplier makes this error metric invariant to absolute scaling, meaning that estimating illumination to be twice as bright or half as bright doesn't change the error. But because there is only one multiplier rather than individual scalings for each RGB channel, this error metric is sensitive to the overall color of the illuminant. This choice seems consistent with what we would like: estimating absolute intensity of an illuminant from a single image is both incredibly

difficult and not very useful, but estimating the color of the illuminant is a reasonable thing to expect from an algorithm, and would be useful for many applications (color constancy, relighting, reflectance estimation, etc). We impose our error metric in the space of visualizations of the illumination rather than in the space of the actual spherical harmonic coefficients that generated that visualization, both because it makes our error metric invariant to the choice of illumination model, and because we found that often the recovered illumination could look quite similar to the ground-truth, while having a very different spherical harmonic representation.

For shading and reflectance, we use:

$$S\text{-MSE}(\hat{s}, s^*) = \frac{1}{n}\min_{\alpha} \sum_{x,y} \left\|\alpha\hat{s}_{x,y} - s^*_{x,y}\right\|_2^2 \quad (27)$$

$$R\text{-MSE}(\hat{r}, r^*) = \frac{1}{n}\min_{\alpha} \sum_{x,y} \left\|\alpha\hat{r}_{x,y} - r^*_{x,y}\right\|_2^2 \quad (28)$$

These are the scale-invariant MSEs of our recovered shading $\hat{s} = \exp(S(\hat{Z}, \hat{L}))$ and reflectance $\hat{r} = \exp(\hat{R})$. Just like in L-MSE, we are invariant to absolute scaling of all RGB channels at once, but not invariant to scaling each channel individually. This makes these error metrics sensitive to errors in estimating the overall color of the shading and reflectance images, but invariant to illumination. Note that these error metrics are of shading and reflectance, not of log-shading and log-reflectance, even though the rest of this paper is written almost entirely in terms of log-intensity. We could have used shift-invariant error metrics in log-intensity space, but we found these to be too sensitive to errors in dark regions of the image — places in which we'd expect any algorithm to do worse, simply because there is less signal.

Our final error metric is the metric introduced in conjunction with the MIT intrinsic images dataset [8], which the authors refer to as LMSE, but which we will call RS-MSE to minimize confusion with L-MSE. This metric measures error for both reflectance and shading, and is locally scale-invariant. The intent of the local scale-invariance is to make the metric insensitive to low-frequency errors in either shading or reflectance. In keeping with this spirit, we apply this error metric individually to each RGB channel and take the mean of those three errors as RS-MSE, making this error metric not just robust to low-frequency error, but robust to most errors in estimating the color of the illumination. This error metric therefore serves to be somewhat complementary to S-MSE and R-MSE, which are sensitive to everything except absolute intensity.

RS-MSE is the mean of the local scale-invariant MSE of shading and reflectance, normalized so that an estimate of all zeros has the maximum score of 1:

$$RS\text{-MSE}(\hat{s}, \hat{r}, s^*, r^*) = \frac{1}{2}\left(\frac{e(\hat{s}, s^*)}{e(\hat{s}, 0)} + \frac{e(\hat{r}, r^*)}{e(\hat{r}, 0)}\right) \quad (29)$$

Where $e(\cdot)$ is the sum of the scale-invariant MSE at all local windows w of size 20×20, spaced in steps of 10:

$$e(\hat{x}, x^*) = \sum_{w \in W} \min_{\alpha} \|\alpha\hat{x}_w - x^*_w\|_2^2 \quad (30)$$

As an aside, in our error metrics we repeatedly use scale-invariant MSE, of the form:

$$\min_{\alpha} \|\alpha\hat{x} - x^*\|_2^2 \quad (31)$$

The closed-form solution to this problem is:

$$\left\| \left(\frac{\hat{x}^{\mathrm{T}} x^*}{\hat{x}^{\mathrm{T}}\hat{x}}\right)\hat{x} - x^* \right\|_2^2 \quad (32)$$

7 Dataset

Here we will detail how we recover "ground-truth" shape and spherical harmonic illumination for each image of each object in our dataset. This is a simple photometric stereo algorithm, in which we optimize over shapes and illuminations to minimize the absolute error between renderings of our dataset and the actual images in our dataset. Absolute error is used to give us robustness to errors due to shadows and specularities, which our rendering engine (and therefore, our dataset) do not consider or address properly. Recovered shapes and illuminations were then cleaned up by hand to address bas-relief ambiguity issues[4]. We treat each RGB channel of each image as a separate image.

To account for varying reflectance, we compute a "shading" image for each image on our dataset.

$$s^*_{i,j} = \exp(I_{i,j} - R_i) \quad (33)$$

We will now detail each step in the inner loop of our iterative photometric stereo algorithm. We first take each current shape estimate Z, and linearize it to get a set of fixed surface normals. For each image j, we solve for the SH illumination that minimizes absolute error between the rendering and the shading image:

$$L_j \leftarrow \arg\min_{L} \sum_i |\exp(S(\mathbf{n}_i, L)) - s^*_{i,j}| \quad (34)$$

This optimization problem is solved using Iteratively Reweighted Least-Squares. We then fix each image's illumination L_j, and optimize over each object's normals $\mathbf{n}_i$.

$$\mathbf{n}_i \leftarrow \arg\min_{\mathbf{n}} \sum_j |\exp(S(\mathbf{n}, L_j)) - s^*_{i,j}| \quad (35)$$

This optimization is done with L-BFGS. In this step, the normals are decoupled, and so surface integrability is not enforced. Given this estimate of surface normals, we can compute a integrable surface Z which approximates this normal field using least-squares:

$$Z \leftarrow \arg\min_{Z} \sum_i \left(Z * h^x - \frac{\mathbf{n}_i^x}{\mathbf{n}_i^z}\right)^2 + \left(Z * h^y - \frac{\mathbf{n}_i^y}{\mathbf{n}_i^z}\right)^2$$

These three optimization steps are repeated until convergence (30 iterations). For the first 10 iterations, we constrain all of the illuminations belonging to the same object to be scaled and shifted versions of each other, but for the next 20 iterations we allow each illumination for every image to vary freely. The result of this algorithm is an estimate of Z for each object and an estimate of L for each RGB channel of every image.

This photometric stereo algorithm still suffers from Bas-Relief ambiguity[4] issues, despite the abundance of data. We therefore manually adjust each recovered Z over the three parameters of the Bas-Relief ambiguity by hand. Also, some regions of Z are clearly incorrect due to shadows. These regions are manually removed (and are not included in the evaluation of our error metrics which concern Z). After these manual tweaks to each shape, we update the set of illuminations to minimize absolute error once again. The two "cup" and "teabag" images did not have discriminative enough shape features for photometric stereo to recover reasonable second-order spherical harmonic illuminations, so for those objects we instead recover only first-order spherical harmonic illumination parameters (equivalent to point-light + ambient illumination), and set the other coefficients to 0.

The MIT Intrinsic Images dataset was not acquired with the goal of having the product of the "shading" and "reflectance" images be exactly equal to the diffuse image, which our model (and our baseline models) assume. That is, a lambertian rendering of our recovered shape and illumination resembles a scaled version of the original "shading" image. We correct for this by adjusting the brightness of the "shading" image such that it matches our rendering in a least-squares sense, and we use this "corrected" shading image in all of our experiments.

Note that the optimization tools we use for our photometric stereo algorithm are completely disjoint from the optimization techniques used by algorithm in our paper, despite the fact that those techniques could have been adapted to do photometric stereo. This was done intentionally to dispel any concerns that our results might be good simply because they were obtained using similar techniques as our photometric stereo algorithm.

Examples of our recovered shapes and illuminations, as well as the shading and reflectance images already contained in the MIT Intrinsic Images dataset, can be seen in Figures 1 and 2.

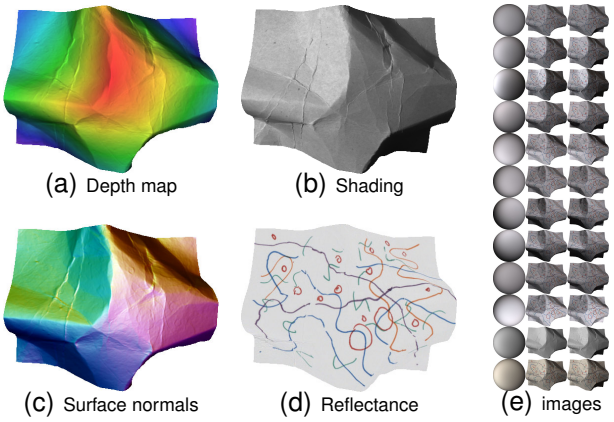

Fig. 1. An object from our dataset. In 1(a), 1(b), 1(c), and 1(d) we have our "ground-truth" shape, shading, surface normals, and reflectance, respectively. The shading and reflectance images come from the MIT Intrinsic Images dataset [8], and the shape and surface normals were produced by our photometric stereo algorithm. In 1(e) we have three columns, where the first contains the images from the MIT Intrinsic Images dataset [8], the third contains the illuminations recovered by our photometric stereo algorithm for each image, and the second column contains renderings of our ground-truth for each illumination, which demonstrate that our recovered models are reasonable. The second to last row of Figure 1(e) is the "shading" image from the MIT dataset, and the last row is the "diffuse" image, which is used as input to our model. The illumination on the last row is therefore what is referred to as the "ground-truth" illumination for this scene, in the rest of the paper.

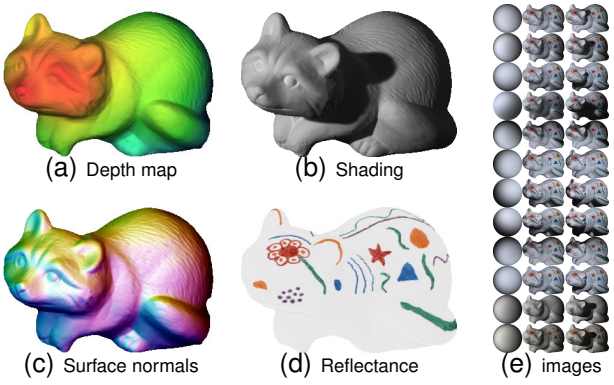

Fig. 2. Another object from our dataset, shown in the same format as Figure 1. Note that our illumination model cannot capture the cast shadows in the input images, which is why our renderings are shadowless.

8 Shape From Shading

Our model for recovering shape and albedo given a single image and illumination can easily be reduced to a model for doing classic shape-from-shading (recovering shape given a single image and illumination).

Our optimization problem becomes:

$$\underset{Z}{\text{minimize}} \qquad \lambda|I - S(Z, L)| + f(Z) \tag{36}$$

Where I is the input log-image, and λ is a multiplier that trades off the importance of the reconstruction terms against the regularizer on Z. $f(Z)$ and $S(Z, L)$ are the same as defined in the paper. Optimization is done using our multiscale optimization algorithm. This SFS algorithm is similar to past algorithms which optimize over a linearized representation of a depth map, with the primary difference being our choice of $f(Z)$.

This SFS algorithm is run on the shading images produced by the "intrinsic image" algorithms we benchmark against. This is a very generous comparison on our part, as we are effectively giving these other algorithms one-half of the model we present here, and we are assuming that illumination is known. We used our own shape-from-shading algorithm for fairness's sake, as it appears to outperform previous SFS algorithms. This means, however, that our improvement over these algorithms is not as much a reflection of the effectiveness of $f(Z)$ *in isolation*, but is instead a demonstration of the effectiveness of optimizing over $f(Z)$ and $g(A)$ to jointly recover shape and albedo, as opposed to recovering a shading image and then recovering shape from that shading image.

REFERENCES

[1] J. C. Principe and D. Xu, "Learning from examples with quadratic mutual information," *Workshop on Neural Networks for Signal Processing*, 1998.

[2] J. Chen, S. Paris, and F. Durand, "Real-time edge-aware image processing with the bilateral grid," *SIGGRAPH*, 2007.

[3] A. Adams, J. Baek, and M. A. Davis, "Fast high-dimensional filtering using the permutohedral lattice," *Eurographics*, 2012.

[4] P. Belhumeur, D. Kriegman, and A. Yuille, "The Bas-Relief Ambiguity," *IJCV*, 1999.

[5] J. Koenderink, A. Van Doorn, C. Christou, and J. Lappin, "Perturbation study of shading in pictures," *Perception*, 1996.

[6] A. Blake, A. Zisserman, and G. Knowles, "Surface descriptions from stereo and shading," *Image and Vision Computing*, 1986.

[7] J. T. Barron and J. Malik, "High-frequency shape and albedo from shading using natural image statistics," *CVPR*, 2011.

[8] R. Grosse, M. K. Johnson, E. H. Adelson, and W. T. Freeman, "Ground-truth dataset and baseline evaluations for intrinsic image algorithms," *ICCV*, 2009.

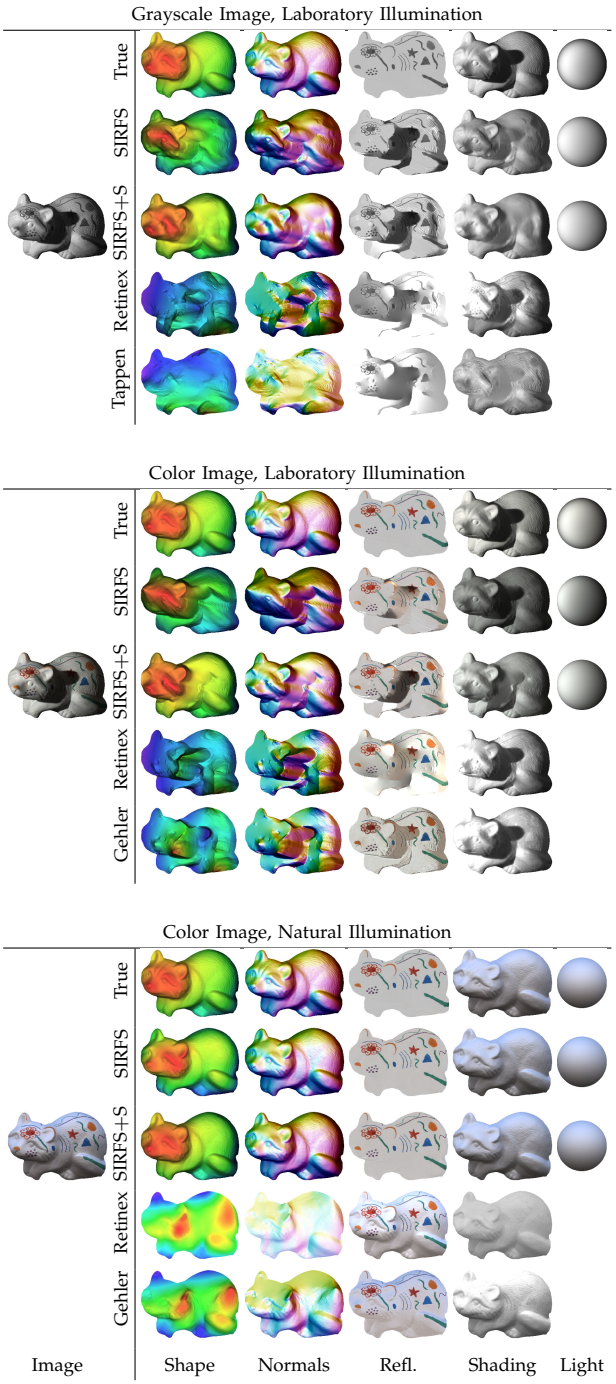

Fig. 3. Here we have a single image from the MIT-Berkeley Intrinsic Images dataset, under three color/illumination conditions. For each condition, we present the ground-truth, the output of SIRFS, the output of SIRFS+S (which uses external shape information), and the two best-performing intrinsic image techniques (for which we do SFS on the recovered shading to recover shape).

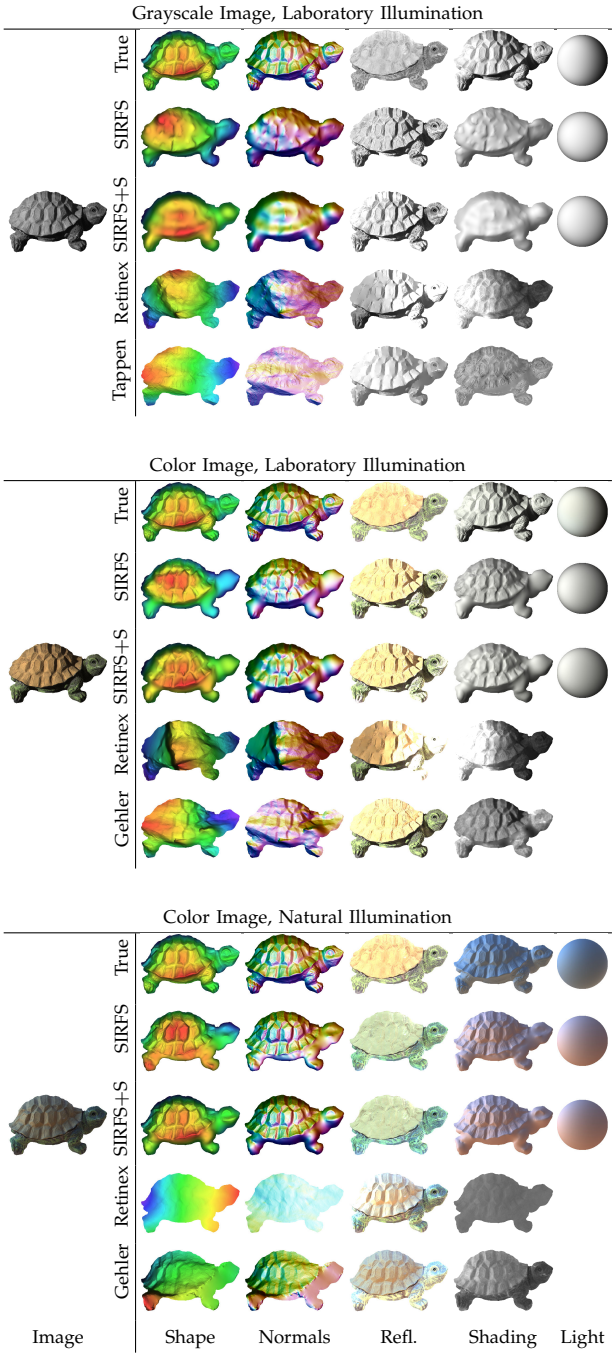

Fig. 4. Here we have another image from the MIT-Berkeley Intrinsic Images dataset.

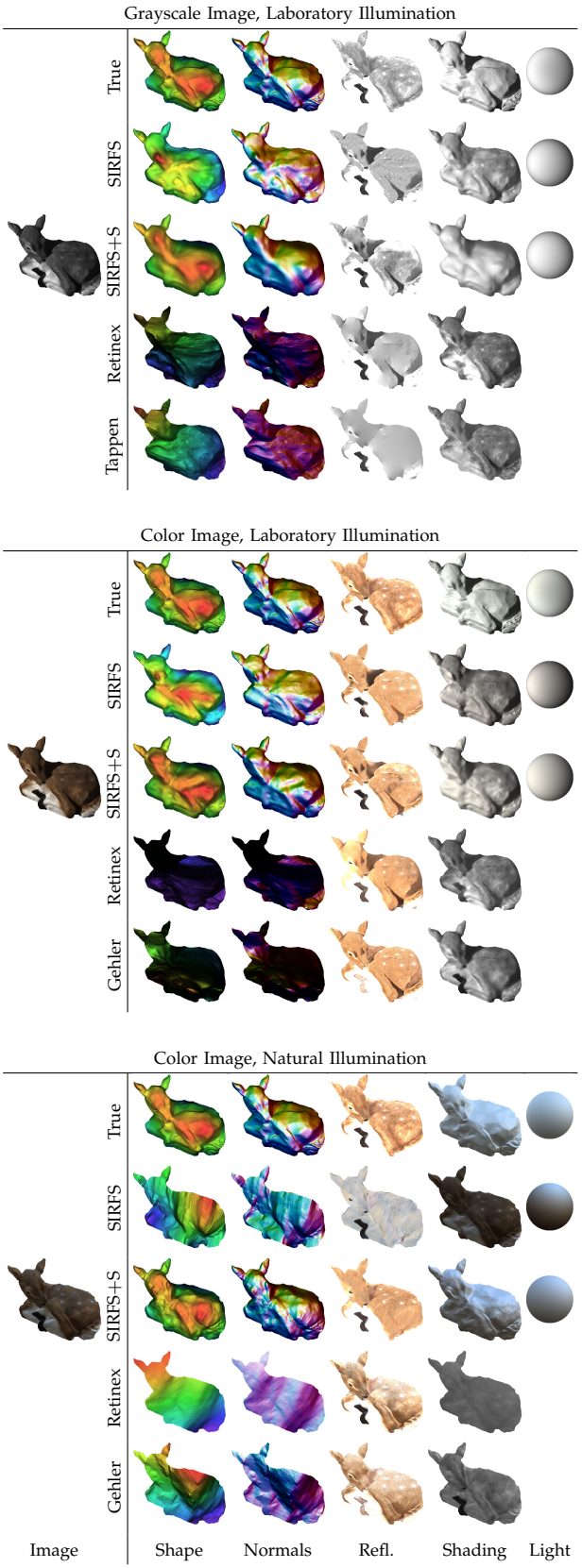

Fig. 5. Here we have another image from the MIT-Berkeley Intrinsic Images dataset.