Shapes, Paint, and Light

Jonathan Barron

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2013-142
http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-142.html

August 12, 2013

Shapes, Paint, and Light

by

Jonathan Tilton Barron

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Jitendra Malik, Chair
Professor Trevor Darrell
Professor Bruno Olshausen

Fall 2013

Shapes, Paint, and Light

Abstract

Shapes, Paint, and Light

by

Jonathan Tilton Barron

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Jitendra Malik, Chair

A fundamental problem in computer vision is that of inferring the intrinsic, 3D structure of the world from flat, 2D images of that world. Traditional methods for recovering scene properties such as shape, reflectance, or illumination rely on multiple observations of the same scene to overconstrain the problem. Recovering these same properties from a single image seems almost impossible in comparison — there are an infinite number of shapes, paint, and lights that exactly reproduce a single image. However, certain explanations are more likely than others: surfaces tend to be smooth, paint tends to be uniform, and illumination tends to be natural. We therefore pose this problem as one of statistical inference, and define an optimization problem that searches for the *most likely* explanation of a single image. Our model, which we call "SIRFS", can be viewed as a superset of several classic computer vision problems (shape-from-shading, intrinsic images, color constancy, illumination estimation, etc) and outperforms all previous solutions to those constituent problems.

Though SIRFS performs well on images of segmented objects, it performs poorly on images of natural scenes, which contain occlusion and spatially-varying illumination. We therefore additionally present Scene-SIRFS, a generalization of SIRFS in which we have a mixture of shapes and a mixture of illuminations, and those mixture components are embedded in a "soft" segmentation of the input image. We additionally use the noisy depth maps provided by RGB-D sensors (in this case, the Kinect) to improve shape estimation.

To Estee, Shelley, Laura, and Will.

Contents

Acknowledgments

Thanks to Jitendra Malik, for being a truly amazing advisor. Thanks to Trevor Darrell and Bruno Olshausen, for your wisdom and enthusiasm. Thanks to Pieter Abbeel and Dan Klein, for the outside-opinions and the meta-education.

Thanks to Sam Roweis, Sven Dickinson, and Fahiem Bacchus, for starting me on the path to research. And thanks to Rob Fergus, Yann Lecun, and David W. Hogg, for helping me find my footing. And thanks to Bill Freeman and Ted Adelson, for the encouragement and the hospitality.

Thanks to all the members of Malik-Group: Michael Maire, Patrik Sundberg, Subhransu Maji, Chunhui Gu, Lubomir Bourdev, Pablo Arbelaez, Thomas Brox, Cees Snoek, Bharath Hariharan, Georgia Gkioxari, Saurabh Gupta, Ross Girshick, Abhishek Kar, Panna Felsen, and Karthik Narayan. Grad school would have been impossible without your advice, your company, and your whiskey cabinet.

Thanks to all my fellow SDH/VLC-mates: Sergey Karayev, Yangqing Jia, Oriol Vinyals, Hyun Oh Song, Judy Hoffman, Ning Zhang, Jeff Donahue, Jon Long, Evan Shelhamer, Woody Hoburg, Nick Hay, Dave Golland, David Hall, Taylor Berg-Kirkpatrick, David Burkett, Adam Pauls, Greg Durrett, Jono Kummerfeld, Mohit Bansal, Fabian Wauthier, Rob Carroll, and Adam Roberts. I never thought I could absorb so much good will and so many good ideas via cubicle-osmosis.

Thanks to my loving and supportive family. I hope this is some consolation for how much time I spent on my laptop during vacations. Thanks to my wonderful girlfriend Estee, for making this all worthwhile. And thanks to Tupper, for not chewing up my laptop.

Chapter 1

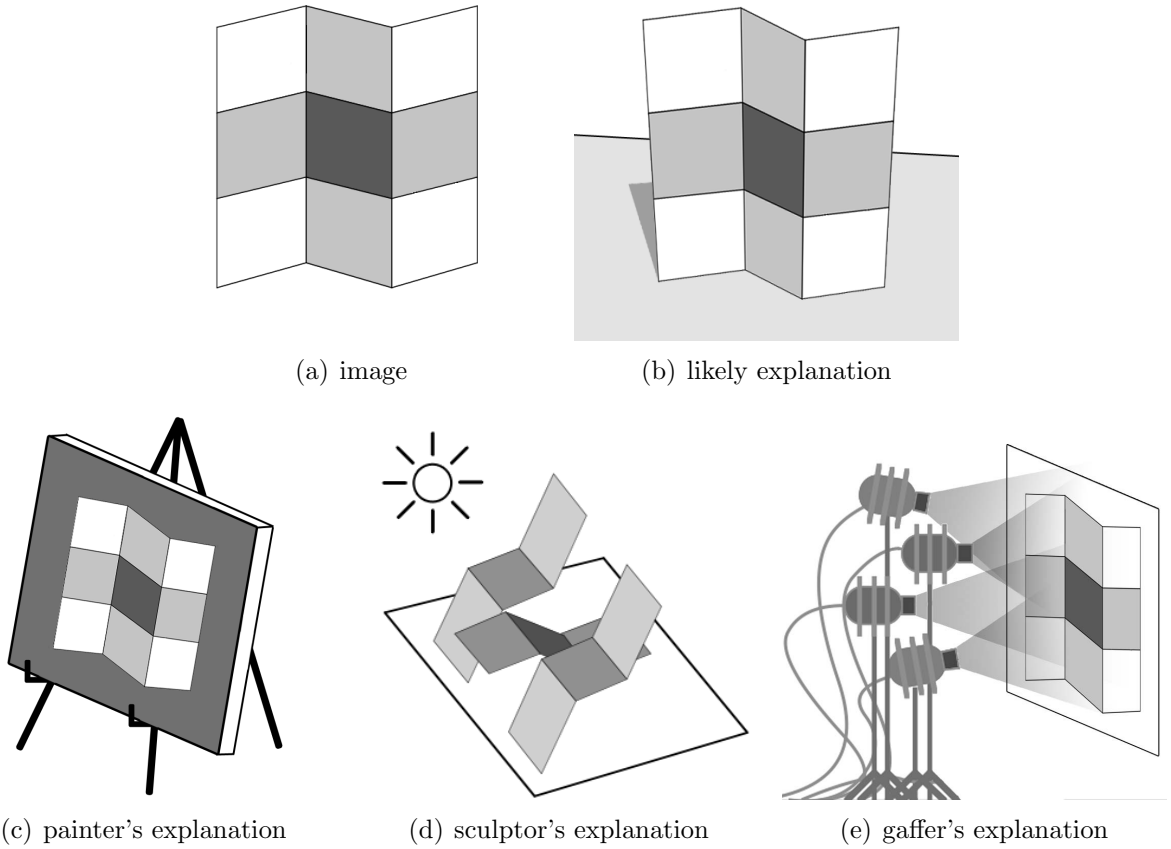
Introduction

At the core of computer vision is the problem of taking a single image, and estimating the physical world which produced that image. The physics of image formation makes this "inverse optics" problem terribly challenging and underconstrained: the space of shapes, paint, and light that exactly reproduce an image is vast.

This problem is perhaps best motivated using Adelson and Pentland's "workshop" metaphor [2]: consider the image in Figure 1.1(a), which has a clear percept as a twice-bent surface with a stroke of dark paint (Figure 1.1(b)). But this scene could have been created using any number of physical worlds — it could be realistic painting on a canvas (Figure 1.1(c)), a complicated arrangement of bent shapes (Figure 1.1(d)), a sophisticated projection produced by a collection of lights (Figure 1.1(e)), or anything in between. The job of a perceptual system is analogous to that of a prudent manager in this "workshop", where we would like to reproduce the scene using as little effort from our three artists as possible, giving us Figure 1.1(b).

This metaphor motivates the formulation of this problem as one of statistical inference. Though there are infinitely many possible explanations for a single image, some are more likely than others. Our goal is therefore to recover *the most likely explanation* that explains an input image. We will demonstrate that in natural depth maps, reflectance maps, and illumination models, very strong statistical regularities arise that are similar to those found in natural images [26, 69]. We will construct priors similar to those used in natural image statistics, but applied separately to shape, reflectance, and illumination. Our algorithm is simply an optimization problem in which we recover the most likely shape, reflectance, and illumination under these priors that exactly reproduces a single image. Our priors are powerful enough that these intrinsic scene properties can be recovered from a single image, but are general enough that they work across a variety of objects.

Earlier versions of this work have been presented in a piecemeal fashion, over the course of many papers [5, 7, 6, 8]. This dissertation is meant to simplify and unify those previous works.



(a) image (b) likely explanation

(c) painter's explanation (d) sculptor's explanation (e) gaffer's explanation

Figure 1.1: A visualization of Adelson and Pentland's "workshop" metaphor [2]. The image in 1.1(a) clearly corresponds to the interpretation in 1.1(b), but it could be a painting, a sculpture, or an arrangement of lights.

1.1 Prior work

The question of how humans solve the underconstrained problem of perceiving shape, reflectance, and illumination from a single image appears to be at least one thousand years old, dating back to the scientist Alhazen, who noted that "Nothing of what is visible, apart from light and color, can be perceived by pure sensation, but only by discernment, inference, and recognition, in addition to sensation." In the 19th century the problem was studied by such prominent vision scientists as von Helmholtz, Hering and Mach [34], who framed the problem as one of "lightness constancy" — how humans, when viewing a flat surface with patches of varying reflectances subject to spatially varying illumination, are able to form a reasonably veridical percept of the reflectance ("lightness") in spite of the fact that a darker patch under brighter illumination may well have more light traveling from it to the eye compared to a lighter patch which is less well illuminated.

Land's Retinex theory of lightness constancy [51] has been particularly influential in computer vision since its introduction in 1971. It provided a computational approach to the problem in the "Mondrian World", a 2D world of flat patches of piecewise constant reflectance. Retinex theory was later made practical by Horn [40], who was able to obtain a decomposition of an image into its shading and reflectance components using the prior belief that sharp edges tend to be reflectance, and smooth variation tends to be shading.

In 1978, Barrow and Tenebaum defined what they called the problem of "intrinsic images": recovering properties such as shape, reflectance, and illumination from a single image [9]. In doing so, they described a challenge in computer vision which is still largely unsolved, and which our work directly addresses. Because this problem is so fundamentally underconstrained and challenging, the computer vision community has largely focused its attention on more constrained and tractable sub-problems. Over time, "intrinsic images" has become synonymous with the problem that Retinex addressed, that of separating an image into shading and reflectance components [35, 40, 51]. This area has seen seen some recent progress [13, 71, 74, 33], though the performance of Retinex, despite its age, has proven hard to improve upon [35]. The limiting factor in many of these "intrinsic image" algorithms appears to be that they treat "shading" as a kind of image, ignoring the fact that shading is, by construction, the product of some shape and some model of illumination. By addressing a superset of this "intrinsic image" problem and recovering shape and illumination instead of shading, our model produces better results than any intrinsic image technique.

Related to the problem of lightness constancy or "intrinsic images" is the problem of color constancy, which can be thought of as a generalization of lightness constancy from grayscale to color, in which the problem is simplified by assuming that there is just one single model of illumination for an entire image, rather than a spatially-varying "shading" effect. Early techniques for color constancy used gamut mapping techniques [31], finite dimensional models of reflectance and illumination [58], and physically based techniques for exploiting specularities [47]. More recent work uses contemporary probabilistic tools, such as modeling the correlation between colors in a scene [27], or performing inference over priors on reflectance and illumination [20]. All of this work shares the assumptions of "intrinsic image" algorithms that shape (and to a lesser extent, shading) can be ignored or abstracted away.

The second subset of the Barrow and Tenenbaum's original "intrinsic image" formulation that the computer vision research community has focused on is the "shape-from-shading" (SFS) problem. SFS is traditionally defined as: recovering the shape of an object given a single image of it, assuming illumination and reflectance are known (or assuming reflectance is uniform across the entire image). This problem formulation is very complimentary to the shape-vs-reflectance version of the "intrinsic images" problem, as it focuses on the parts of the problem which "intrinsic images" ignores, and vice-versa.

The shape-from-shading problem was first formulated in the computer vision community by Horn in 1975 [41], though the problem existed in other fields as that of "photoclinometry" [66]. The history of SFS is well surveyed in [21, 80]. Despite being a severe simplification of the complete intrinsic images problem, SFS is still a very ill-posed and underconstrained,

Figure 1.2: A bas-relief sculpture. The bas-relief sculpture on the right appears to be a deep three-dimensional scene, despite being a physically shallow piece of metal.

and challenging problem. Onenotable difficulty in SFS is the bas-relief ambiguity [11], which states (roughly) that the absolute orientation and scaling of a surface is ambiguous given only shading information. This ambiguity holds true not only for SFS algorithms, but for human vision as well [48]. An example of this ambiguity can be seen in bas-relief sculptures, such as in Figure 1.2, which convey the percept of a rich, deep, three-dimensional scene despite being physically shallow. We address this ambiguity by imposing priors on shape, building on notions of "smoothness" priors in SFS [44], and by allowing for external observations of shape (such as those produced by a stereo system or depth sensor) to be introduced.

Our model can be viewed as a generalization of an "intrinsic image" algorithm or color constancy algorithm in which shading is explicitly parametrized as a function of shape and illumination. Similarly, our model can be viewed as a shape-from-shading algorithm in which reflectance and illumination are unknown, and are recovered. Our model therefore addresses the "complete" intrinsic images problem, as it was first formulated. By addressing the complete problem, rather than two sub-problems in isolation, we outperform all previous algorithms for either subproblem. This is consistent with our understanding of human perception, as humans use spatial cues when estimating reflectance and shading [34, 18].

Because the intrinsic images problem is so challenging given only a single image, a much more popular area of research in computer vision has been to introduce additional data to better constrain the problem. Instances of this approach are photometric stereo [78], which use additional images with different illumination conditions to estimate shape, and in later work reflectance and illumination [10]. Our algorithm produces the same kinds of output as the most advanced photometric stereo algorithm, while requiring only a single image.

Problem	Input	Output
"Intrinsic Images" [13, 40, 74]	1 Image	Shading, Refl.
Shape from Shading [41]	1 Shading, Illum.	Shape
Shape from Contour [19, 49]	1 Image	Coarse Shape
Photometric Stereo [10, 78]	k Images	Shape, Shading, Refl., Illum.
Inverse Global Illumination [79]	k Images, Shape	Shading, Refl., Illum.
Stereo / Structure from Motion [36]	k Images	Shape
Spatial Layout Estimation [38, 70]	1 Image	Coarse Shape
Morphable Models [17]	1 Image of a face	Shape, Shading, Refl., Illum.
SIRFS	1 Image of a masked object	Shape, Shading, Refl., Illum.

Table 1.1: A taxonomy of "intrinsic image"-style problems in computer vision. For each technique we list the minimum required input and the maximum output.

"Structure from motion" or binocular stereo [36, 76] uses multiple images to recover shape, but ignores shading, reflectance, and illumination. Inverse global illumination [79] recovers reflectance and illumination given shape and multiple images, while we recover shape and require only a single image.

Recent work has explored using learning to directly infer the spatial layout of a scene from a single image [38, 70]. These techniques ignore illumination and reflectance, and produce only a coarse estimate of shape.

A similar approach to our technique is that of category-specific morphable models [17] which, given a single image of a very specific kind of object (a face, usually), estimates shape, reflectance, and illumination. These techniques use extremely specific models (priors) of the objects being estimated, and therefore do not work for general objects, while our priors are general enough to be applicable on a wide variety of objects: a single model learned on teabags and squirrels can be applied to images of coffee cups and turtles.

To provide a unifying view of this complete intrinsic image problem, we present Table 1.1, which is a sort of taxonomy of these problems according to the input they require, and the output they have been shown to produce. We see that our model (SIRFS) is capable of producing all of the output of any other technique, while requiring only a single image, and while not requiring that the input belong to some specific category.

The driving force behind our model are our priors on shape, reflectance, and illumination. To construct these priors we build upon past work on natural image statistics, which has demonstrated that simple statistics govern local patches of natural images [26, 69, 43], and that these statistics can be used for denoising [63], inpainting [68], deblurring [25], etc. But these statistical regularities arise in natural images only because of *the statistical regularities in the underlying worlds that produced those images*. The primary contribution of this work is extended these ideas from natural images to the world that produced that natural image, which is assumed to be composed of natural depth maps and natural reflectance images. There has been some study of the statistics of natural depth maps [42], reflectance images

[67] and models of illumination [24], but ours is the first to use these statistical observations for recovering all such intrinsic scene properties simultaneously.

Chapter 2

SIRFS

This chapter will proceed as follows: In Section 2.1 we will formulate our problem as one of statistical inference and optimization, with respect to a set of priors over shape, reflectance, and illumination. In Sections 2.2, 2.3, and 2.4 we present and motivate our priors on reflectance, shape, and illumination, respectively. In Section 2.5 we describe the rendering engine used within SIRFS, and detail how to backpropagate gradients across that rendering engine. In Section 2.6 we explain how we solve our proposed optimization problem. In Section 2.7 we present a series of experiments with our model on real and pseudo-synthetic variants of a dataset for which we have ground-truth, and on additional real-world images. In Section 2.9 we detail some of the limitations of our model, some of which we will address in Chapter 3.

2.1 Problem Formulation

We call our problem formulation for recovering intrinsic scene properties from a single image of a (masked) object "shape, illumination, and reflectance from shading", or "SIRFS". SIRFS can be thought of as an extension of classic shape-from-shading models [39] in which not only shape, but reflectance and illumination are unknown. Conversely, SIRFS can be framed as an "intrinsic image" technique for recovering shading and reflectance, in which shading is parametrized by a model of shape and illumination. The SIRFS problem formulation is:

$$\begin{aligned} \underset{R,Z,L}{\text{maximize}} \quad & P(R)P(Z)P(L) \\ \text{subject to} \quad & I = R + S(Z, L) \end{aligned} \tag{2.1}$$

Where R is a log-reflectance image, Z is a depth-map, and L is a spherical-harmonic model of illumination [65]. Z and R are "images" with the same dimensions as I, and L is a vector parametrizing the illumination. $S(Z, L)$ is a "rendering engine" which linearizes Z into a set of surface normals, and produces a log-shading image from those surface normals and L (see the supplementary material). $P(R)$, $P(Z)$, and $P(L)$ are priors on reflectance, shape, and

illumination, respectively, whose likelihoods we wish to maximize subject to the constraint that the log-image I is equal to a rendering of our model $R + S(Z, L)$. We can simplify this problem formulation by reformulating the maximum-likelihood aspect as minimizing a sum of cost functions (by taking the negative log of $P(R)P(Z)P(L)$) and by absorbing the constraint and removing R as a free parameter. This gives us the following unconstrained optimization problem:

$$\underset{Z,L}{\text{minimize}} \quad g(I - S(Z, L)) + f(Z) + h(L) \tag{2.2}$$

where $g(R)$, $f(Z)$, and $h(L)$ (Sections 2.2, 2.3, and 2.4, respectively) are cost functions for reflectance, shape, and illumination respectively, which we will refer to as our "priors" on these scene properties [1]. Solving this problem (Section 2.6) corresponds to searching for the least costly (or most likely) explanation $\{Z, R, L\}$ for image I.

2.2 Priors on Reflectance

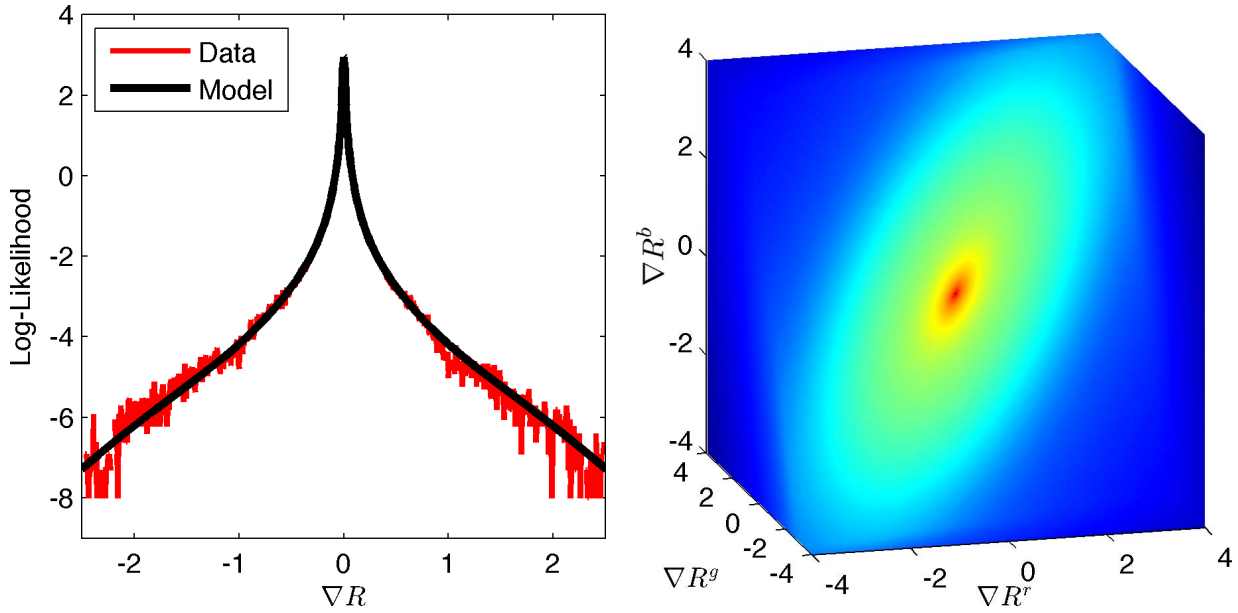
Our prior on reflectance consists of three components: 1) An assumption of piecewise constancy, which we will model by minimizing the local variation of log-reflectance in a heavy-tailed fashion. 2) An assumption of parsimony of reflectance — that the palette of colors with which an entire image was painted tends to be small — which we model by minimizing the global entropy of log-reflectance. 3) An "absolute" prior on reflectance which prefers to paint the scene with some colors (white, gray, green, brown, etc) over others (absolute black, neon pink, etc), thereby addressing color constancy. Formally, our reflectance prior $g(A)$ is a weighted combination of three costs:

$$g(R) = \lambda_s g_s(R) + \lambda_e g_e(R) + \lambda_a g_a(R) \tag{2.3}$$

where $g_s(R)$ is our smoothness prior, $g_e(R)$ is our parsimony prior, and $g_a(R)$ is our "absolute" prior. The λ multipliers are learned through cross-validation on the training set.

Our smoothness and parsimony priors are on the differences of log-reflectance, which makes them equivalent to priors on the ratios of reflectance. This makes intuitive sense, as reflectance is defined as a ratio of reflected light to incident light, but is also crucial to the success of our algorithm: Consider the reflectance-map ρ implied by log-image I and log-shading $S(Z, L)$, such that $\rho = \exp(I - S(Z, L))$. If we were to manipulate Z or L to increase $S(Z, L)$ by some constant α across the entire image, then ρ would be divided by $\exp(\alpha)$ across the entire image, which would accordingly decrease the differences between

[1] Throughout this dissertation we use the term "prior" loosely. We refer to loss functions or regularizers on Z, A, and L as "priors" because they often have an interpretation as the negative log-likelihood of some probability density function. We refer to minimizing entropy as a "prior", which is again an abuse of terminology. Our occluding contour "prior" and our external observation "prior" require first observing the silhouette of an object or some external observation of shape, respectively, and are therefore posteriors, not priors.



(a) univariate/grayscale GSM (b) multivariate/color GSM

Figure 2.1: Our smoothness prior on log-reflectance is a univariate Gaussian scale mixture on the differences between nearby reflectance pixels for grayscale images, or a multivariate GSM for color images. These distribution prefers nearby reflectance pixels to be similar, but its heavy tails allow for rare non-smooth discontinuities. Our multivariate color model captures the correlation between color channels, which means that chromatic variation in log-reflectance lies further out in the tails, making it more likely to be ignored during inference.

pixels of ρ. Therefore, if we placed priors on the differences of reflectance it would be possible to trivially satisfy our priors by manipulating shape or illumination to increase the intensity of the shading image. However, in the log-reflectance case $R = I - S(Z, L)$, increasing all of S by α (increasing the brightness of the shading image) simply decreases all of R by α, and does not change the differences between log-reflectance values (it would, however, affect our absolute prior on reflectance). Priors on the differences of log-albedo are therefore invariant to scaling of illumination or shading, which means they behave similarly in well-lit regions as in shadowed regions, and cannot be trivially satisfied.

2.2.1 Smoothness

The reflectance images of natural objects tend to be piecewise constant — or equivalently, variation in reflectance images tends to be small and sparse. This is the insight that underlies the Retinex algorithm [35, 51, 40], and informs more recent intrinsic images work [71, 74, 33].

Our prior on grayscale reflectance smoothness is a multivariate Gaussian scale mixture (GSM) placed on the differences between each reflectance pixel and its neighbors. We will

(a) some R (b) $g_s(R)$ (cost) (c) $\nabla g_s(R)$ (influence)

Figure 2.2: Here we have a color reflectance image R, and its cost and influence (derivative of cost) under our multivariate GSM smoothness prior. Strong, colorful edges, such as those caused by reflectance variation, are very costly, while small edges, such as those caused by shading, are less costly. But in terms of influence — the gradient of cost with respect to each pixel — we see an inversion: because sharp edges lie in the tails of the GSM, they have little influence, while shading variation has great influence. This means that during inference our model attempts to explain shading (small, achromatic variation) in the image by varying shape, while explaining sharp or chromatic variation by varying reflectance.

maximize the likelihood of R under this model, which corresponds to minimizing the following cost function:

$$g_s(R) = \sum_i \sum_{j \in N(i)} c\left(R_i - R_j \,;\, \boldsymbol{\alpha}_R,\, \boldsymbol{\sigma}_R\right) \tag{2.4}$$

Where $N(i)$ is the 5×5 neighborhood around pixel i, $R_i - R_j$ is a the difference in log-RGB from pixel i to pixel j, and $c\left(\cdot \,;\, \boldsymbol{\alpha},\, \boldsymbol{\sigma}\right)$ is the negative log-likelihood of a discrete univariate Gaussian scale mixture (GSM), parametrized by $\boldsymbol{\alpha}$ and $\boldsymbol{\sigma}$, the mixing coefficients and standard deviations, respectively, of the Gaussians in the mixture:

$$c(x; \boldsymbol{\alpha}, \boldsymbol{\sigma}) = -\log \sum_{j=1}^{M} \alpha_j \mathcal{N}\left(x \,;\, 0, \sigma_j^2\right) \tag{2.5}$$

We set the mean of the GSM is 0, as the most likely reflectance image under our model should be flat. We set $M = 40$ (the GSM has 40 discrete Gaussians), and $\boldsymbol{\alpha}_R$ and $\boldsymbol{\sigma}_R$ are trained on reflectance images in our training set using expectation-maximization. The log-likelihood of our learned model can be seen in Figure 2.1(a).

Gaussian scale mixtures have been used previously to model the heavy-tailed distributions found in natural images [63], for the purpose of denoising or inpainting. Effectively, using this family of distributions gives us a log-likelihood which looks like a smooth, heavy-tailed spline which decreases monotonically with distance from 0. Because it is monotonically decreasing, the cost of log-reflectance variation increases with the magnitude of variation, but because the distribution is heavy tailed, the influence of variation (the derivative of log-likelihood) is strongest when variation is small (that is, when variation resembles shading)

and weaker when variation is large. This means that our model prefers a reflectance image that is mostly flat but occasionally varies heavily, but abhors a reflectance image which is constantly varying slightly. This behavior is similar to that of the Retinex algorithm, which operates by shifting strong gradients to the reflectance image and weak gradients to the shading image.

To extend our model to color images, we simply extend our smoothness prior to a multivariate Gaussian scale mixture

$$g_s(R) = \sum_i \sum_{j \in N(i)} C\left(R_i - R_j \,;\, \boldsymbol{\alpha}_R,\, \boldsymbol{\sigma}_R, \Sigma_{\mathrm{R}}\right) \tag{2.6}$$

Where $R_i - R_j$ is now a 3-vector of the log-RGB differences, $\boldsymbol{\alpha}$ are mixing coefficients, $\boldsymbol{\sigma}$ are the scalings of the Gaussians in the mixture, and Σ is the covariance matrix of the entire GSM (shared among all Gaussians of the mixture).

$$C(\mathbf{x} \,;\, \boldsymbol{\alpha}, \boldsymbol{\sigma}, \Sigma) = -\log \sum_{j=1}^{M} \alpha_j \, \mathcal{N}\left(\mathbf{x} \,;\, \mathbf{0}, \sigma_j \, \Sigma\right) \tag{2.7}$$

We set $M = 40$ (the GSM has 40 discrete Gaussians), and we train $\boldsymbol{\alpha}_R$, $\boldsymbol{\sigma}_R$, and Σ_R on color reflectance images in our training set (we train a distinct model from the grayscale smoothness model). The log-likelihood of our learned model, and the training data used to learn that model, can be seen in Figure 2.1(b).

In color images, variation in reflectance tends to manifest itself in both the luminance and chrominance of an image (white transitioning to blue, for example) while shading, assuming the illumination is mostly white, primarily affects the luminance of an image (light blue transitioning to dark blue, for example). Past work has exploited this insight by building specialized models that condition on the chrominance variation of the input image [35, 40, 71, 74, 33]. By placing a multivariate prior over differences in reflectance, we are able to capture the correlation of the different color channels, which implicitly encourages our model to explain chromatic variation using reflectance and achromatic variation using shading without the need for any hand-crafted heuristics. See Figure 2.2 for a demonstration of this effect. Our model places more-colorful edges further into the tails of the distribution, thereby reducing their influence. Again, this is similar to color variants of the Retinex algorithm [35] which uses the increased chrominance of an edge as a heuristic for it being a reflectance edge. But this approach (which is common among intrinsic image algorithms) of using image chrominance as a substitute for reflectance chrominance means that these techniques fail when faced with non-white illumination, while our model is robust to non-white illumination.

2.2.2 Parsimony

In addition to piece-wise smoothness, the second property we expect from reflectance images is for there to be a small number of reflectances in an image — that the palette with which an

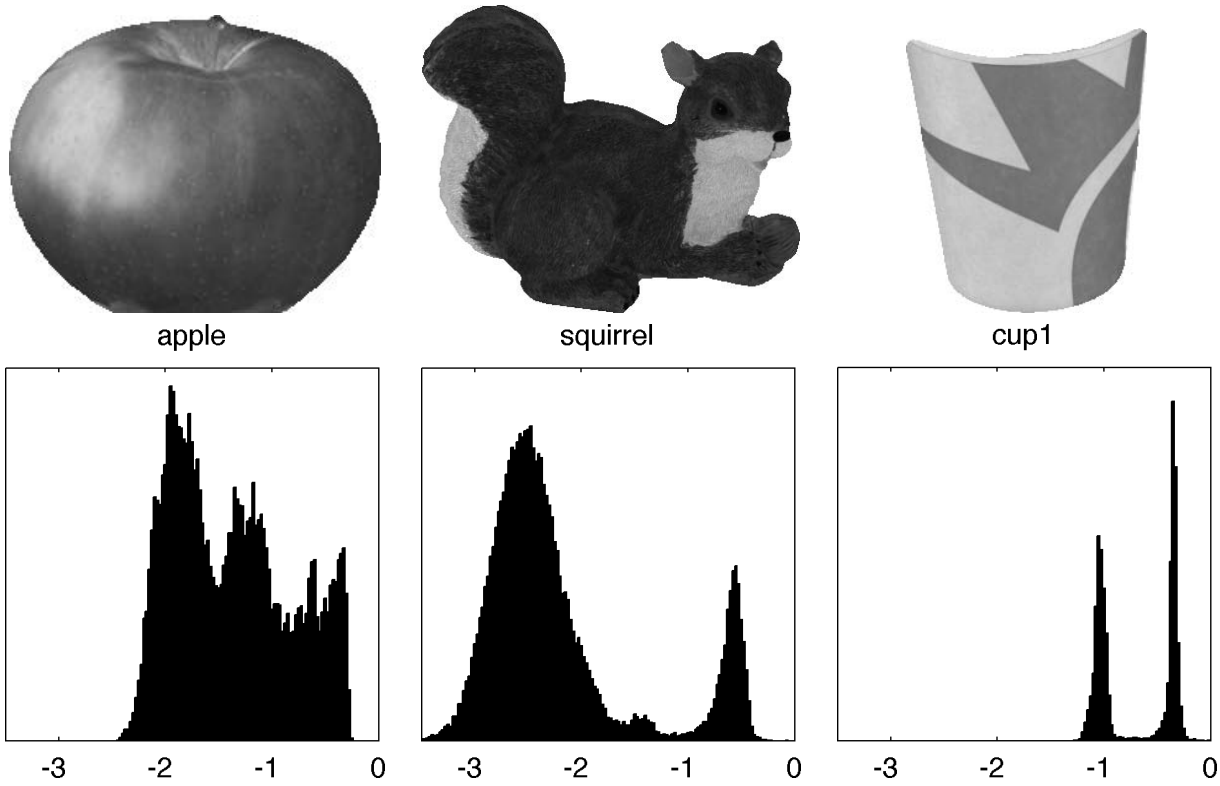


Figure 2.3: Three grayscale log-reflectance images from our dataset and their marginal distributions. Log-reflectance in an image tend to be grouped around certain values, or equivalently, these distributions tend to be low-entropy.

image was painted be small. As a hard constraint, this is not true: even in painted objects, there are small variations in reflectance. But as a soft constraint, this assumption holds. In Figure 2.3 we show the marginal distribution of grayscale log-reflectance for three objects in our dataset. Though the man-made "cup1" object shows the most clear peakedness in its distribution, natural objects like "apple" show significant clustering.

We will therefore construct a prior which encourages parsimony – that our representation of the reflectance of the scene be economical and efficient, or "sparse". This is effectively a instance of Occam's razor, that one should favor the simplest possible explanation. We are not the first to explore global parsimony priors on reflectance: different forms of this idea have been used in intrinsic images techniques [33], photometric stereo [3], shadow removal [28], and color representation [61]. We use the quadratic entropy formulation of [64] to minimize the entropy of log-reflectance, thereby encouraging parsimony. Formally, our parsimony prior

for reflectance is:

$$\begin{aligned} g_e(R) &= -\log\left(\frac{1}{Z}\sum_{i=1}^{N}\sum_{j=1}^{N}\exp\left(-\frac{(R_i-R_j)^2}{4\sigma_R^2}\right)\right) \\ Z &= N^2\sqrt{4\pi\sigma^2} \end{aligned} \tag{2.8}$$

This is quadratic entropy (a special case of Rényi entropy) for a set of points $\mathbf{x}$ assuming a Parzen window (a Gaussian kernel density estimator, with a bandwidth of σ_R) [64]. Effectively, this is a "soft" and differentiable generalization of Shannon entropy, computed on a set of real values rather than a discrete histogram. By minimizing this quantity, we encourage all pairs of reflectance pixels in the image to be similar to each other. However, minimizing this entropy does not force all pixels to collapse to one value, as the "force" exerted by each pair falls off exponentially with distance — it is robust to outliers. This prior effectively encourages Gaussian "clumps" of reflectance values, where the Gaussian clumps have standard deviations of roughly σ_R.

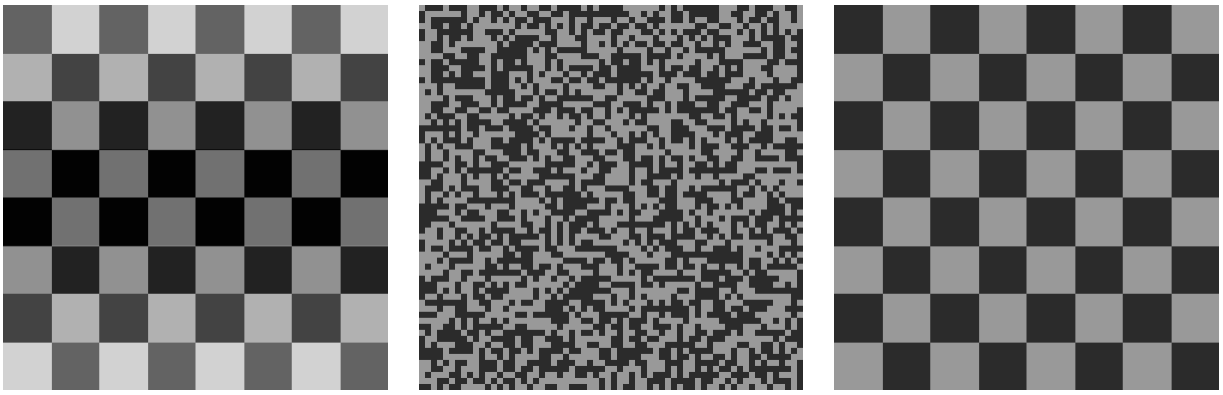
At first glance, it may seem that this global parsimony prior is redundant with our local smoothness prior: Encouraging piecewise smoothness seems like it should cause entropy to be minimized indirectly. This is often true, but there are common situations in which both of these priors are necessary. For example, if two regions are separated by a discontinuity in the image then optimizing for local smoothness will never cause the reflectance on both sides of the discontinuity to be similar. Conversely, simply minimizing global entropy may force reflectance to take on a small number of values, but need not produce large piecewise-smooth regions. The merit of using both priors in conjunction is demonstrated in Figure 2.4.

Generalizing our grayscale parsimony prior to color reflectance images requires generalizing our entropy model to higher dimensionalities. A naive extension of this one-dimensional entropy model to three dimensions is not sufficient for our purposes: The RGB channels of natural reflectance images are highly correlated, causing a naive "isotropic" high-dimensional entropy measure to work poorly. To address this, we pre-compute a whitening transformation from log-reflectance images in the training set, and compute an isotropic entropy measure in this whitened space during inference, which gives us an anisotropic entropy measure. Formally, our cost function is quadratic entropy in the space of whitened log-reflectance:

$$g_e(R) = -\log\left(\frac{1}{Z}\sum_{i=1}^{N}\sum_{j=1}^{N}\exp\left(-\frac{\|\mathrm{W_R}(R_i-R_j)\|_2^2}{4\sigma_R^2}\right)\right) \tag{2.9}$$

Where $\mathrm{W_R}$ is the whitening transformation learned from reflectance images in our training set, as follows: Let X be a $3\times n$ matrix of the pixels in the reflectance images in our training set. We compute the matrix $\Sigma = \mathrm{X}\mathrm{X}^{\mathrm{T}}$, take its eigenvalue decomposition $\Sigma = \Phi\Lambda\Phi^{\mathrm{T}}$, and from that construct the whitening transformation $\mathrm{W_R} = \Phi\Lambda^{1/2}\Phi^{\mathrm{T}}$ [2]. σ_R is the bandwidth of

[2] Our whitening transformation of reflectance is not strictly correct, as we do not first center the data by subtracting the mean. This was done both for mathematical and computational convenience, and because the origin of the space of log-reflectance (absolute white) is arguable the most reasonable choice for the "center" of our data.



(a) No parsimony (b) No smoothness (c) Both

Figure 2.4: A demonstration of the importance of both our smoothness and parsimony priors on reflectance. Using only a smoothness prior, as in 2.4(a), allows for reflectance variation across disconnected regions. Using only the parsimony prior, as in 2.4(b), encourages reflectance to take on a small number of values, but does not encourage it to form large piecewise-constant regions. Only by using the two priors in conjunction, as in 2.4(c), does our model correctly favor a normal, paint-like checkerboard configuration.

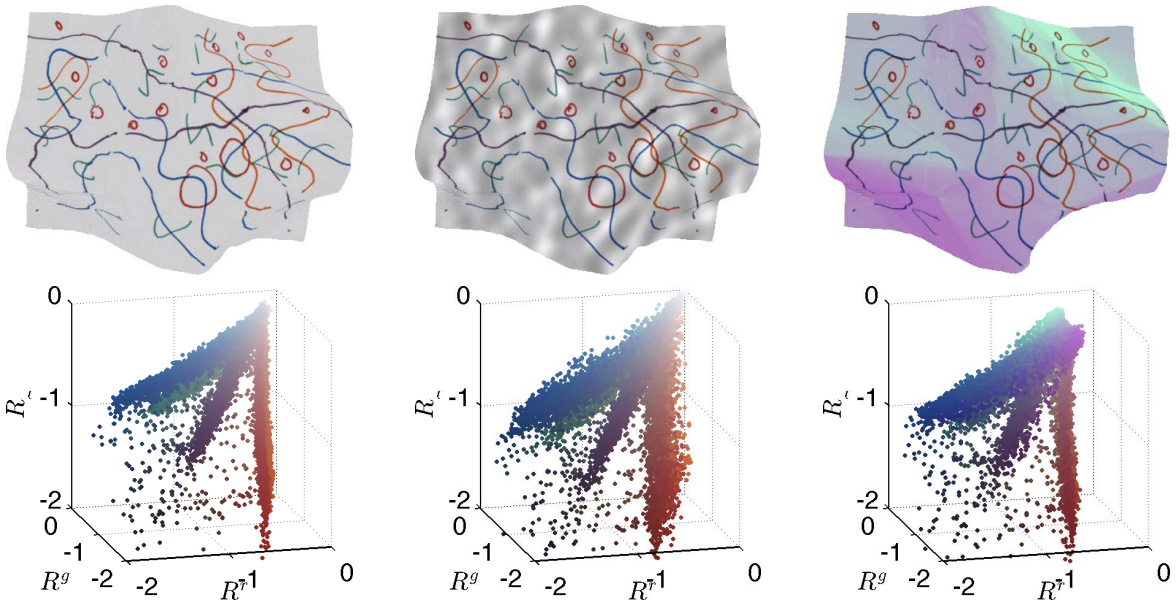
the Parzen window, which determines the scale of the clusters produced by minimizing this entropy measure, and is tuned through cross-validation (independently of the same variable for the grayscale case). See Figure 2.5 for a motivation of this model.

Naively computing this quadratic entropy measure requires calculating the difference between all N log-reflectance values in the image with all other N log-reflectance values, making it quadratically expensive in N to compute naively. In Section 2.2.3 we describe an accurate linear-time algorithm for approximating this quadratic entropy and its gradient, based on the bilateral grid [22].

2.2.3 Efficient Quadratic Entropy

Here we will detail a novel method for calculating the quadratic entropy measure introduced in [64], which we use in Equations 2.8 and 2.9 of 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:

$$\begin{aligned} H(\mathbf{x}) &= -\log\left(\frac{1}{Z}\sum_{i=1}^{N}\sum_{j=1}^{N}\exp\left(-\frac{(\mathbf{x}_i-\mathbf{x}_j)^2}{4\sigma^2}\right)\right) \\ Z &= N^2\sqrt{4\pi\sigma^2} \end{aligned} \tag{2.10}$$



(a) Correct (b) Wrong Shape (c) Wrong Light

Figure 2.5: Some reflectance images and their corresponding log-RGB scatterplots. Mistakes in estimating shape or illumination produce shading-like or illumination-like errors in the inferred reflectance, causing the the log-RGB distribution of the reflectance to be "smeared", and causing entropy (and therefore cost) to increase.

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:

$$H(\mathbf{n}) = -\log\left(\sum_{a=1}^{M}\mathbf{n}_a\sum_{b=1}^{M}\frac{\mathbf{n}_b}{Z}\exp\left(-\frac{W^2(a-b)^2}{4\sigma^2}\right)\right) \tag{2.11}$$

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:

$$\mathbf{g}_d = \frac{1}{Z}\exp\left(-\frac{W^2d^2}{4\sigma^2}\right) \tag{2.12}$$

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

$$H(\mathbf{n}) = -\log\left(\mathbf{n}^{\mathrm{T}}(\mathbf{n}*\mathbf{g})\right) \tag{2.13}$$

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 $\mathbf{n}$:

$$\nabla H(\mathbf{n}) = \left(\frac{-2}{\mathbf{n}^{\mathrm{T}}(\mathbf{n} * \mathbf{g})}\right)(\mathbf{n} * \mathbf{g}) \tag{2.14}$$

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:

$$b_L = \lfloor (x_i - R_L)/W \rfloor, \quad b_U = b_L + 1 \tag{2.15}$$

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

$$w_L = (x_i - b_L)/W, \quad w_U = 1 - w_L \tag{2.16}$$

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

$$\mathbf{n}_L = \mathbf{n}_L + w_L, \quad \mathbf{n}_U = \mathbf{n}_U + w_U \tag{2.17}$$

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

$$\frac{\partial \mathbf{n}_L}{\partial x_i} = -\frac{1}{W}, \quad \frac{\partial \mathbf{n}_U}{\partial x_i} = \frac{1}{W} \tag{2.18}$$

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 $\mathbf{x}$:

$$\nabla H(\mathbf{x}) \approx J^{\mathrm{T}} \nabla H(\mathbf{n}) \tag{2.19}$$

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 2.10. 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.

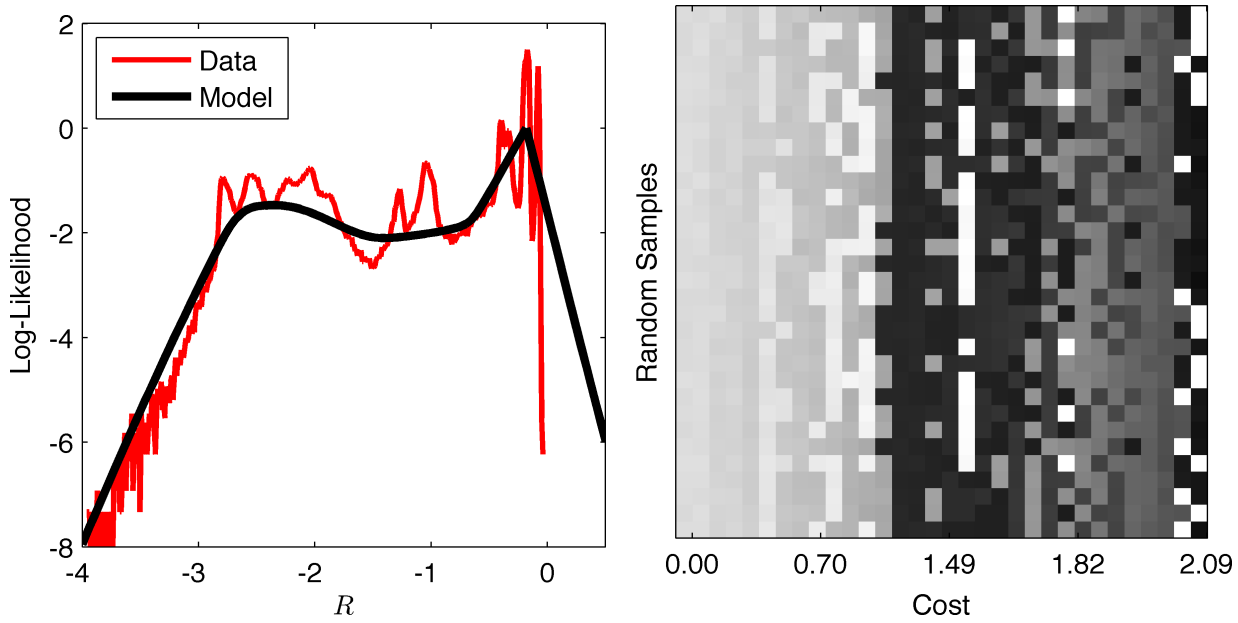
Note that this formulation is extremely similar to the bilateral grid [22], which is a tool for high-dimensional Gaussian filtering (used mostly for bilateral filtering, hence the name). The calculation of our entropy measure is extremely similar to the "splat, blur, slice" pipeline in other high-dimensional Gaussian filtering works [1], 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 [1] 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.

2.2.4 Absolute Reflectance

The previously described priors were imposed on *relative* properties of reflectance: the differences between nearby or not-nearby pixels. We must impose an additional prior on *absolute* reflectance: the raw value of each pixel in the reflectance image. Without such a prior (and the prior on illumination presented in Section 2.4) our model would be equally pleased to explain a gray pixel in the image as gray reflectance under gray illumination as it would nearly-black reflectance under extremely-bright illumination, or blue reflectance under yellow illumination, etc.

This sort of prior is fundamental to color-constancy, as most basic white-balance or auto-contrast/brightness algorithms can be viewed as minimizing a similar sort of cost: the gray-world assumption penalizes reflectance for being non-gray, the white-world assumption penalizes reflectance for being non-white, and gamut-based models penalize reflectance for lying outside of a gamut of previously-seen reflectances. We experimented with variations or combinations of these types of models, but found that what worked best was using a regularized smooth spline to model the log-likelihood of log-reflectance values.

For grayscale images, we use a 1D spline, which we have fit to log-reflectance images in



(a) Training data and PDF (b) Samples

Figure 2.6: A visualization of our "absolute" prior on grayscale reflectance, trained on the MIT Intrinsic Images dataset [35]. In 2.6(a) we have the log-likelihood of our density model, and the data on which it was trained. In 2.6(a) we have samples from our model, where the x axis is sorted by cost (y axis is random).

the training set as follows:

$$\underset{\mathbf{f}}{\text{minimize}} \quad \mathbf{f}^{\mathrm{T}}\mathbf{n} + \log\left(\sum_i \exp\left(-\mathbf{f}_i\right)\right) + \lambda\sqrt{(\mathbf{f}'')^2 + \epsilon^2} \tag{2.20}$$

Where $\mathbf{f}$ is our spline, which determines the non-normalized negative log-likelihood (cost) assigned to every reflectance, $\mathbf{n}$ is a 1D histogram of log-reflectance in our training data, and $\mathbf{f}''$ is the second derivative of the spline, which we robustly penalize (ϵ is a small value added in to make our regularization differentiable everywhere). Minimizing the sum of the first two terms is equivalent to maximizing the likelihood of the training data (the second term is the log of the partition function for our density estimation), and minimizing the third term causes the spline to be piece-wise smooth. The smoothness multiplier λ is tuned through cross-validation. A visualization of our prior can be found in Figure 2.6.

During inference, we maximize the likelihood of the grayscale reflectance image R by minimizing its cost under our learned model:

$$g_a(R) = \sum_i \mathbf{f}(R_i) \tag{2.21}$$

where $\mathbf{f}(R_i)$ is the value of $\mathbf{f}$ at R_i, the log-reflectance at pixel i, which we computed using linear interpolation (so that this cost is differentiable).

To generalize this model to color reflectance images, we simply use a 3D spline, trained on whitened log-reflectance pixels in our training set. Formally, to train our model we minimize the following:

$$\begin{aligned} \underset{\mathrm{F}}{\text{minimize}} \; & <\mathrm{F}, \mathrm{N}> + \log\left(\sum_i \exp\left(-\mathrm{F}_i\right)\right) + \lambda\sqrt{J(\mathrm{F}) + \epsilon^2} \\ & J(\mathrm{F}) = \mathrm{F}_{xx}^2 + \mathrm{F}_{yy}^2 + \mathrm{F}_{zz}^2 + 2\mathrm{F}_{xy}^2 + 2\mathrm{F}_{yz}^2 + 2\mathrm{F}_{xz}^2 \end{aligned} \tag{2.22}$$

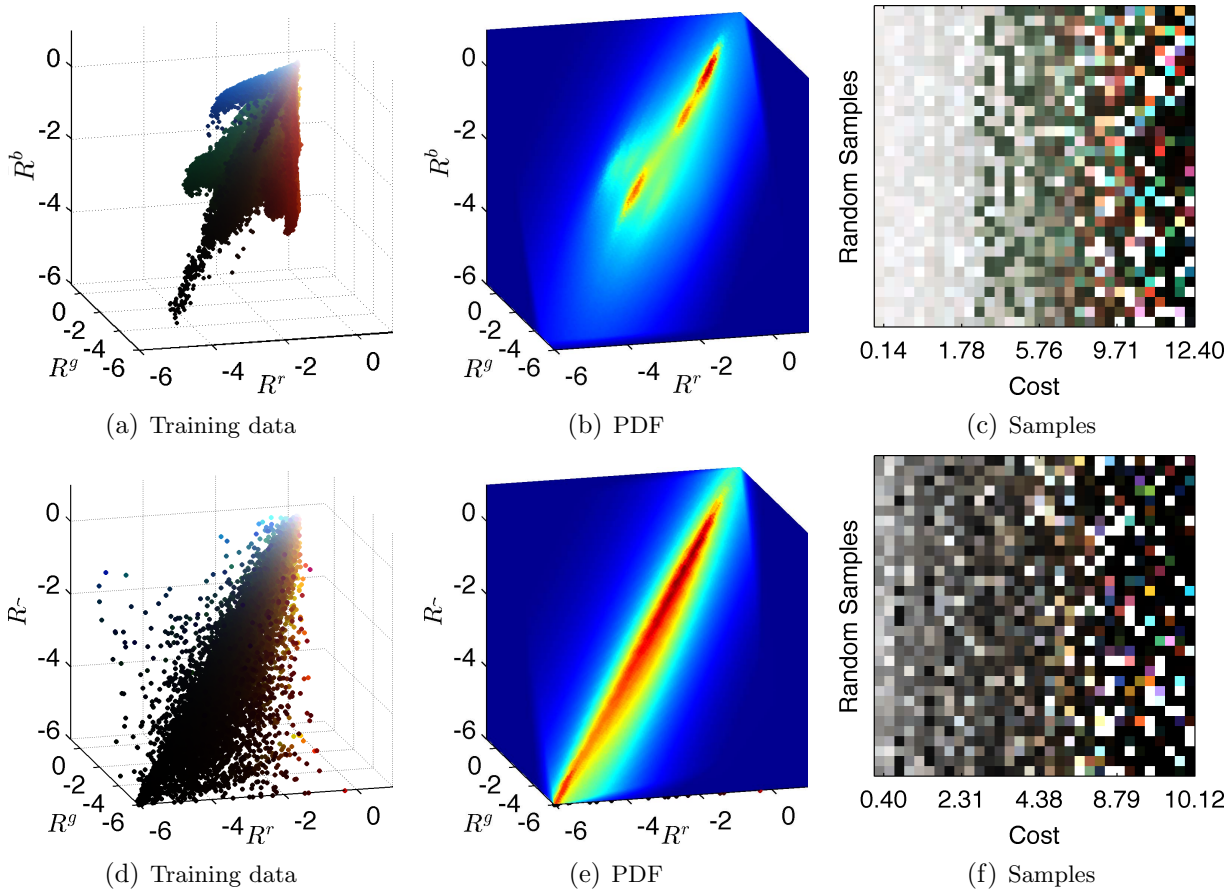
Where F is our 3D spline describing cost, N is a 3D histogram of the whitened log-RGB reflectance in our training data, and $J(\cdot)$ is a smoothness penalty (the thin-plate spline smoothness energy, made more robust by taking its square root). The smoothness multiplier λ is tuned through cross-validation. As in our parsimony prior, we use whitened log-reflectance to address the correlation between channels, which is necessary as our smoothness term is isotropic. A visualization of our prior can be seen in Figure 2.7.

During inference, we maximize the likelihood of the color reflectance image R by minimizing its cost under our learned model:

$$g_a(R) = \sum_i \mathrm{F}(\mathrm{W}_\mathrm{R} R_i) \tag{2.23}$$

where $\mathrm{F}(\mathrm{W}_\mathrm{R} R_i)$ is the value of F at the coordinates specified by the 3-vector $\mathrm{W}R_i$, the whitened reflectance at pixel i (W_R is the same as in Section 2.2.2). To make this function differentiable, we compute $\mathrm{F}(\cdot)$ using trilinear interpolation.

We trained our absolute color prior on the MIT Intrinsic Images dataset [35], and used that learned model in all experiments shown in this paper. However, the MIT dataset is very small and this absolute prior contains very many parameters (hundreds, in contrast to our other priors which are significantly more constrained), which suggests that we may be overfitting to the small set of reflectances in the MIT dataset. To address this concern, we trained an additional version of our absolute prior on the color reflectances in the OpenSurfaces dataset [14], which is a huge and varied dataset that is presumably a more accurate representation of real-world reflectances. Both models can be seen in Figure 2.7, where we see that the priors we learn for each dataset are somewhat different, but that both prefer lighter, desaturated reflectances. We ran some additional experiments using our OpenSurfaces model instead of our MIT model (not presented in this dissertation), and found that the outputs of each model were virtually indistinguishable. This is a testament to the robustness of our model, and suggests that we are not overfitting to the color reflectances in the MIT dataset.



(a) Training data (b) PDF (c) Samples

(d) Training data (e) PDF (f) Samples

Figure 2.7: A visualization of our "absolute" prior on color reflectance. We train two versions of our prior, one on the MIT Intrinsic Images dataset [35] that we use in our experiments (top row) and one on the OpenSurfaces dataset for comparison [14] (bottom row). In the first-column we have the log-RGB reflectance pixels in our training set, and in the second column we have a visualization of the 3D spline PDF that we fit to that data. In the third column we have samples from the PDF, where the x axis is sorted by cost (y axis is random). For both datasets, our model prefers less saturated, more earthy or subdued colors, and abhors brightly lit neon-like colors or very dark colors — the high-cost reflectances often do not even look like paint, but instead appear glowing and luminescent.

2.3 Priors on Shape

Our prior on shape consists of four components: 1) An assumption of smoothness (that shapes tend to bend rarely), which we will model by minimizing the variation of mean curvature. 2) An assumption of isotropy of the orientation of surface normals (that shapes are just as likely to face in one direction as they are another) which reduces to a well-motivated "fronto-parallel" prior on shapes. 3) An prior on the orientation of the surface

normal near the boundary of masked objects, as shapes tend to face outward at the occluding contour. 4) An optional prior that the shape should resemble some noisy or incomplete external observation, such as an estimate of depth derived from stereo or a depth sensor. Formally, our shape prior $f(Z)$ is a weighted combination of four costs:

$$f(Z) = \lambda_k f_k(Z) + \lambda_i f_i(Z) + \lambda_c f_c(Z) + \lambda_o f_o(Z, \hat{Z}) \tag{2.24}$$

where $f_k(Z)$ is our smoothness prior, $f_i(Z)$ is our isotropy prior, f_c is our bounding contour prior, and $f_o(Z, \hat{Z})$ encourages Z to be similar to some observation $\hat{Z}$, all of which will be explained in detail in the following sections. The λ multipliers are learned through cross-validation on the training set.

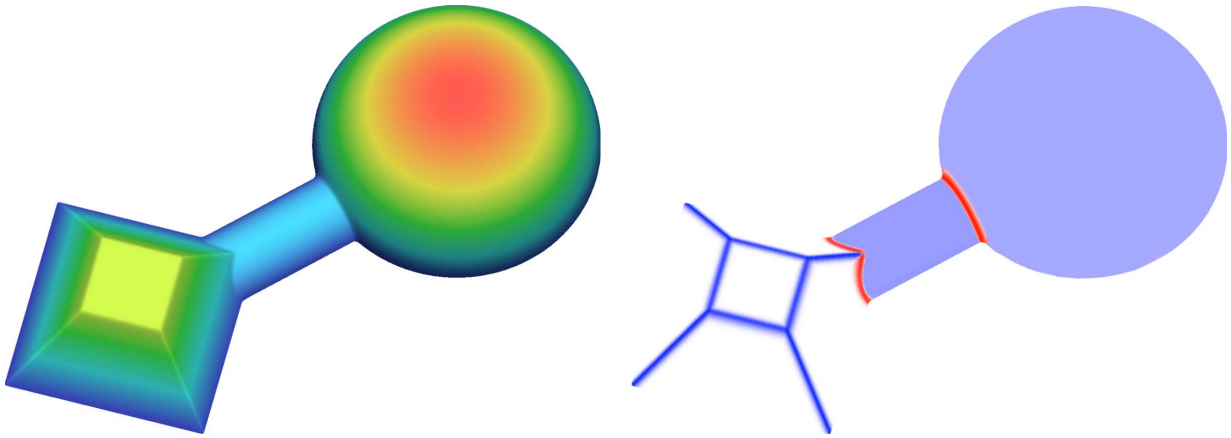
Most of our shape priors are imposed on intermediate representations of shape, such as mean curvature or surface normals. This requires that we compute these intermediate representations from a depth map, calculate the cost and the gradient of cost with respect to those intermediate representations, and backpropagate the gradients back onto the shape. In the supplementary material we explain in detail how to efficiently compute these quantities and backpropagate through them.

2.3.1 Smoothness

There has been much work on modeling the statistics of natural shapes [42, 77], with one overarching theme being that regularizing some function of the second derivatives of a surface is effective. However, this past work has severe issues with invariance to out-of-plane rotation and scale. Working within differential geometry, we present a shape prior based on the variation of mean curvature, which allows us to place smoothness priors on Z that are invariant to rotation and scale.

To review: mean curvature is the divergence of the normal field. Planes and soap films have 0 mean curvature everywhere, spheres and cylinders have constant mean curvature everywhere, and the sphere has the smallest total mean curvature among all convex solids with a given surface area [37]. See Figure 2.8 for a visualization. Mean curvature is a measure of curvature in *world coordinates*, not image coordinates, so (ignoring occlusion) the marginal distribution of $H(Z)$ is invariant to out-of-plane rotation of Z — a shape is just as likely viewed from one angle as from another. In comparison, the Laplacian of Z and the second partial derivatives of Z can be made large simply due to foreshortening, which means that priors placed on these quantities [77] would prefer certain shapes simply due to the angle from which those shapes are observed — clearly undesirable.

But priors on raw mean curvature are not scale-invariant. Were we to minimize $|H(Z)|$, then the most likely shape under our model would be a plane, while spheres would be unlikely. Were we to minimize $|H(Z) - \alpha|$ for some constant α, then the most likely shape under our model would be a sphere of a certain radius, but larger or smaller spheres, or a resized image of the same sphere, would be unlikely. Clearly, such scale sensitivity is an undesirable property for a general-purpose prior on natural shapes. Inspired by previous work



(a) some shape Z (b) mean curvature $H(Z)$

Figure 2.8: A visualization of a shape and its mean curvature (blue = positive, red = negative, white = 0). Planes and soap films have 0 mean curvature, spheres and cylinders have constant mean curvature, and mean curvature varies where shapes bend.

on minimum variation surfaces [60], we place priors on the local variation of mean curvature. The most likely shapes under such priors are surfaces of constant mean curvature, which are well-studied in geometry and include soap bubbles and spheres of any size (including planes). Priors on the variation of mean curvature, like priors on raw mean curvature, are invariant to rotation and viewpoint, as well as concave/convex inversion.

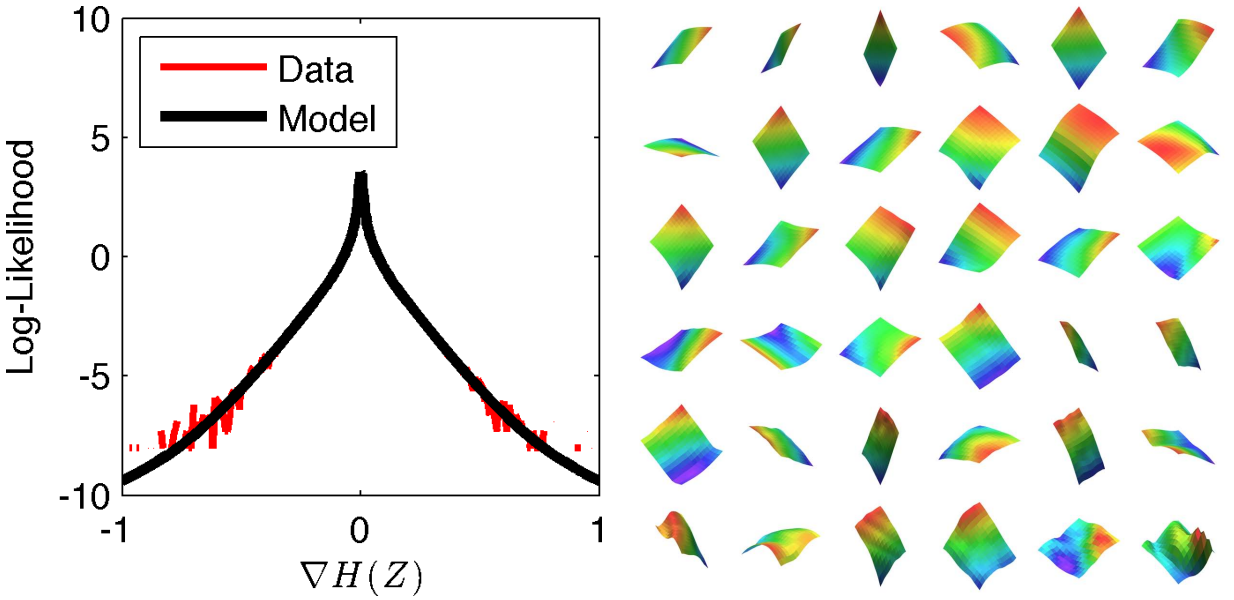
Mean curvature is defined as the average of principle curvatures: $H = \frac{1}{2}(\kappa_1 + \kappa_2)$. It can be approximated on a surface using filter convolutions that approximate first and second partial derivatives, as show in [15].

$$H(Z) = \frac{\left(1 + Z_x^2\right) Z_{yy} - 2 Z_x Z_y Z_{xy} + \left(1 + Z_y^2\right) Z_{xx}}{2\left(1 + Z_x^2 + Z_y^2\right)^{3/2}} \tag{2.25}$$

In Section 2.3.2 we detail how to calculate and differentiate $H(Z)$ efficiently. Our smoothness prior for shapes is a Gaussian scale mixture on the local variation of the mean curvature of Z:

$$f_k(Z) = \sum_i \sum_{j \in N(i)} c\left(H(Z)_i - H(Z)_j \,;\, \boldsymbol{\alpha}_k,\, \boldsymbol{\sigma}_k\right) \tag{2.26}$$

Notation is similar to Equation 2.4: $N(i)$ is the 5×5 neighborhood around pixel i, $H(Z)$ is the mean curvature of shape Z, and $H(Z)_i - H(Z)_j$ is the difference between the mean curvature at pixel i and pixel j. $c\left(\cdot\,; \boldsymbol{\alpha}, \boldsymbol{\sigma}\right)$ is defined in Equation 2.5, and is the negative log-likelihood (cost) of a discrete univariate Gaussian scale mixture (GSM), parametrized by $\boldsymbol{\alpha}$ and $\boldsymbol{\sigma}$, the mixing coefficients and standard deviations, respectively, of the Gaussians in the mixture. The mean of the GSM is 0, as the most likely shapes under our model



(a) Smoothness (b) Samples

Figure 2.9: To encourage shapes to be smooth, we model the variation in mean curvature of shapes using a Gaussian scale mixture, shown in 2.9(a). In 2.9(b) we show patches of shapes in our training data, sorted from least costly (upper left) to most costly (lower right). Likely shapes under our model look like soap-bubbles, and unlikely shapes look contorted.

should be smooth. We set $M = 40$ (the GSM has 40 discrete Gaussians), and $\boldsymbol{\alpha}_k$ and $\boldsymbol{\sigma}_k$ are learned from our training set using expectation-maximization. The log-likelihood of our learned model can be seen in Figure 2.9(a), and the likelihoods it assigns to different shapes can be visualized in Figure 2.9(b). The learned GSM is very heavy tailed, which encourages shapes to be mostly smooth, and occasionally very non-smooth — or equivalently, our prior encourages shapes to bend rarely.

2.3.2 Mean Curvature

The definition of our smoothness priors on shapes in Section 2.3.1 is somewhat complicated, as we are placing priors on $H(Z)$, and intermediate representation of Z, rather than on Z itself. We must therefore be able to efficiently compute $H(Z)$, and backpropagate the gradient of some loss defined with respect to $H(Z)$ back onto Z.

Mean curvature on a surface is a function of the first and second partial derivatives of that surface.

$$H(Z) = \frac{\left(1 + Z_x^2\right) Z_{yy} - 2 Z_x Z_y Z_{xy} + \left(1 + Z_y^2\right) Z_{xx}}{2\left(1 + Z_x^2 + Z_y^2\right)^{3/2}} \tag{2.27}$$

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

$$\begin{gathered} Z_x = Z * h_3^x, \quad Z_y = Z * h_3^y \\ Z_{xx} = Z * h_3^{xx}, \; Z_{yy} = Z * h_3^{yy}, \; Z_{xy} = Z * h_3^{xy} \\ h_3^x = \frac{1}{8}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad h_3^y = \frac{1}{8}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \\ h_3^{xy} = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \quad h_3^{yy} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ -2 & -4 & -2 \\ 1 & 2 & 1 \end{bmatrix}, \quad h_3^{xx} = \frac{1}{4}\begin{bmatrix} 1 & -2 & 1 \\ 2 & -4 & 2 \\ 1 & -2 & 1 \end{bmatrix} \end{gathered} \tag{2.28}$$

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

$$\begin{aligned} M &= \sqrt{1 + Z_x^2 + Z_y^2} \\ N &= (1 + Z_x^2)Z_{yy} - 2Z_x Z_y Z_{xy} + (1 + Zy^2)Z_{xx} \\ D &= 2M^3 \\ H(Z) &= N/D \end{aligned} \tag{2.29}$$

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.

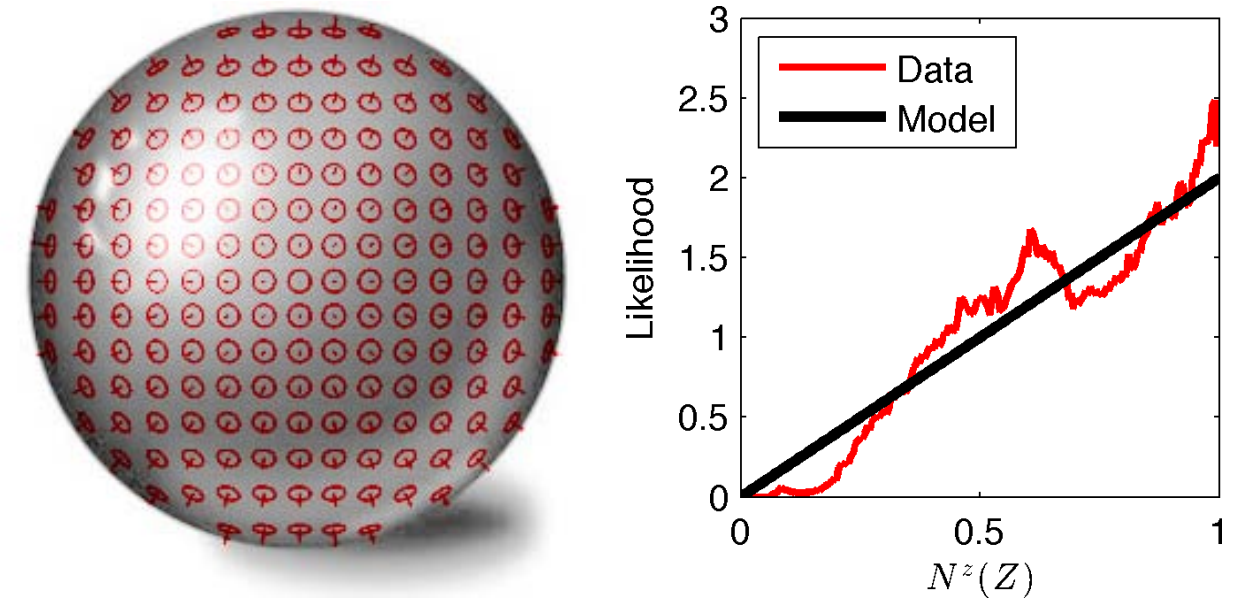
$$\begin{aligned} F_x &= 2(Z_x Z_{yy} - Z_{xy} Z_y) - \frac{3Z_x N}{M^2} \\ F_y &= 2(Z_{xx} Z_y - Z_x Z_{xy}) - \frac{3Z_y N}{M^2} \\ F_{xx} &= 1 + Z_y^2 \\ F_{yy} &= 1 + Z_x^2 \\ F_{xy} &= -2Z_x Z_y \end{aligned} \tag{2.30}$$

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:

$$\begin{aligned} B &= \frac{\nabla_{H(Z)} f}{D} \\ \nabla_Z f &= (BF_x) \star h_3^x + (BF_y) \star h_3^y \\ &+ (BF_{xx}) \star h_3^{xx} + (BF_{yy}) \star h_3^{yy} + (BF_{xy}) \star h_3^{xy} \end{aligned} \tag{2.31}$$

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

This allows us to compute an arbitrary prior on $H(Z)$ and efficiently compute the gradient of Z with respect to that prior, which is necessary for the gradient-based optimization we will perform in Section 2.6.



(a) An isotropic shape

(b) Our isotropy prior

Figure 2.10: We assume the surfaces of shapes to be isotropic — equally likely to face in any orientation, like in a sphere. However, observing an isotropic shape imposes a bias, as observed surfaces are more likely to face the observer than to be perpendicular to the observer (as shown by the red gauge figure "thumbtacks" placed on the sphere in 2.10(a)). We undo this bias by imposing a prior on N^z, shown in 2.10(b), which coarsely resembles our training data.

2.3.3 Surface Isotropy

Our second prior on shapes is motivated by the observation that shapes tend to be oriented isotropically in space. That is, it is equally likely for a surface to face in any direction. This assumption is not valid in many settings, such as man-made environments (which tend to be composed of floors, walls, and ceilings) or outdoor scenes (which are dominated by the ground-plane). But this assumption is more true for generic objects floating in space, which tend to resemble spheres (whose surface orientations are truly isotropic) or sphere-like shapes — though there is often a bias on the part of photographers towards imaging the front-faces of objects. Despite its problems, this assumption is still effective and necessary.

Intuitively, one may assume that imposing this isotropy assumption requires no effort: if our prior assumes that all surface orientations are equally likely, doesn't that correspond to a constant cost for all surface orientations? However, this ignores the fact that once we have observed a surface in space, we have introduced a bias: observed surfaces are much more likely to face the observer ($N^z \approx 1$) than to be perpendicular to the observer ($N^z \approx 0$). We must therefore impose an isotropy prior to undo this bias.

We will derive our isotropy prior analytically. Assume surfaces are oriented uniformly, and

that the surfaces are observed under orthogonal perspective with a view direction $(0, 0, -1)$. It follows that all N^z (the z-component of surface normals, relative to the viewer) are distributed uniformly between 0 and 1. Upon observation, these surfaces (which are assumed to have identical surface areas) have been foreshortened, such that the area of each surface in the image is N^z. Given the uniform distribution over N^z and this foreshortening effect, the probability distribution over N^z that we should expect at a given pixel in the image is proportional to N^z. Therefore, maximizing the likelihood of our uniform distribution over orientation in the world is equivalent to minimizing the following in the image:

$$f_i(Z) = -\sum_{x,y} \log\left(N^z_{x,y}(Z)\right) \tag{2.32}$$

Where $N^z_{x,y}(Z)$ is the z-component of the surface normal of Z at position (x, y) (defined in Section 2.5).

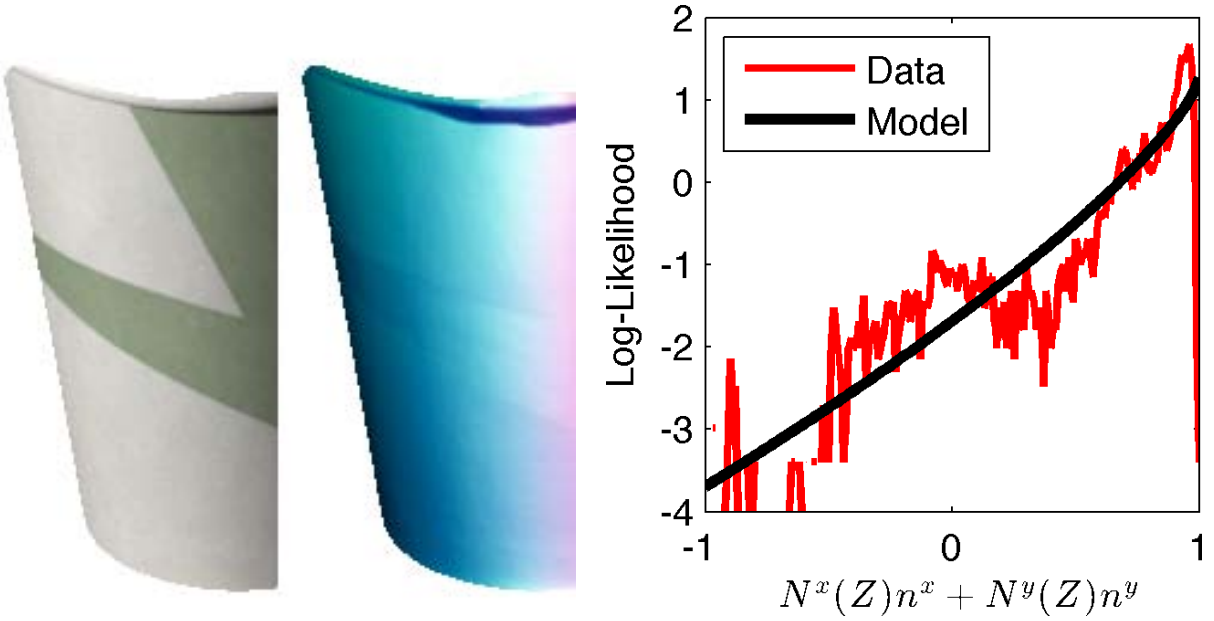
Though this was derived as an isotropy prior, the shape which maximizes the likelihood of this prior is not isotropic, but is instead (because of the nature of MAP estimation) a fronto-parallel plane. This gives us some insight into the behavior of this prior — it serves to as a sort of "fronto-parallel" prior. This prior can therefore be thought of as combating the bas-relief ambiguity [11] (roughly, that absolute scale and orientation are ambiguous), by biasing our shape estimation towards the fronto-parallel members of the bas-relief family.

Our prior on N^z is shown in Figure 2.10(b) compared to the marginal distribution of N^z in our training data. Our model fits the data well, but not perfectly. We experimented with learning distributions on N^z empirically, but found that they worked poorly compared to our analytical prior. We attribute this to the aforementioned photographer's bias towards fronto-parallel surfaces, and to data sparsity when N^z is close to 0.

It is worth noting that $-\log(N^z)$ is proportional to the surface area of Z. Our prior on slant therefore has a helpful interpretation as a prior on minimal surface area: we wish to minimize the surface area of Z, where the degree of the penalty for increasing Z's surface area happens to be motivated by an isotropy assumption. This notion of placing priors on surface area has been explored previously [29], but not in the context of isotropy. And of course, this connection relates our model to the study of minimal surfaces in mathematics [37], though this connection is somewhat tenuous as the fronto-parallel planes favored by our model are very different from classical minimal surfaces such as planes and soap films.

2.3.4 The Occluding Contour

The occluding contour of a shape (the contour that surrounds the silhouette of a shape) is a powerful cue for shape interpretation [49] which often dominates shading cues [59], and algorithms have been presented for coarsely estimating shape given contour information [19]. At the occluding contour of an object, the surface is tangent to all rays from the vantage point. Under orthographic projection (which we assume), this means the z-component of the normal is 0, and the x and y components are determined by the contour in the image. In



(a) A cropped object an its normals

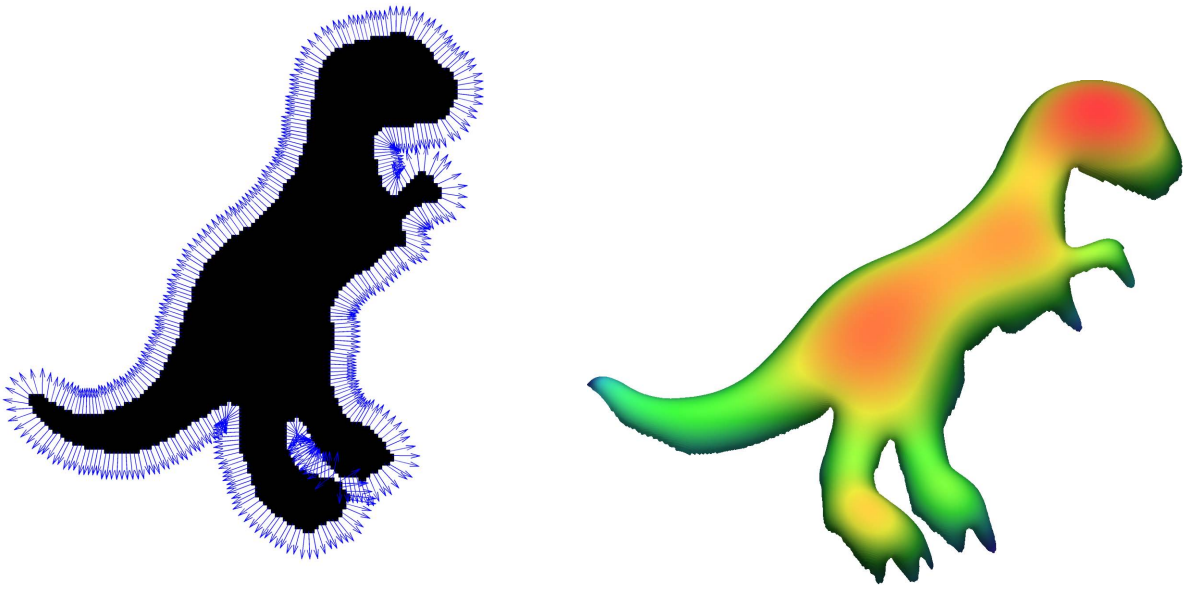
(b) Our occluding contour prior

Figure 2.11: In 2.11(a) we have an image and surface normals of a subset of a cup, in our dataset. The side of this cup are "limbs", points where the surface normal faces outward and is perpendicular to the occluding contour, while the top of the cup are "edges", sharp discontinuities where the surface is oriented arbitrarily. Our heavy-tailed prior over surface orientation at the occluding contour in 2.11(b) models the behavior of limbs, but is robust to the outliers caused by edges.

principle, this property is absolutely true, but in practice the occluding contour of a surface tends to be composed of limbs (points where the surface is tangent to rays from the vantage point, like the smooth side of a cylinder) and edges (an abrupt discontinuity of the surface, like the top of a cylinder or the edge of a piece of paper) [57]. See Figure 2.11(a) for an example of a shape which contains both phenomena. Of course, this taxonomy is somewhat false — all edges are limbs, but some are so small that they appear to be edges, and some are just small enough relative to the image resolution that the "limb" assumption begins to break down.

We present a "soft" version of a limb constraint, one which captures the "limb"-like behavior we expect to see but which can be violated by edges or small limbs. Because our dataset consists of masked objects, identifying the occluding contour C is trivial (see Figure 2.12(a)). For each point i on C, we estimate n_i, the local normal to the occluding contour in the image plane. Using those we regularize the surface normals in Z along the boundary by minimizing the following loss:

$$f_c(Z) = \sum_{i \in C} (1 - (N_i^x(Z)n_i^x + N_i^y(Z)n_i^y))^{\gamma_c} \tag{2.33}$$



(a) occluding contour normals (b) shape-from-contour output

Figure 2.12: A subset of our model that includes only our priors on shape is equivalent to a shape-from-contour model. Given only the normals of the silhouette of the object in 2.12(a), we can produce the coarse estimate of the shape of the object in 2.12(b).

Where $N(Z)$ is the surface normal of Z, as defined in Section 2.5. We set $\gamma_c = 0.75$, which fits the training data best, and which performs best in practice. The inner product of n_i and N_i (both of which are unit-norm) is 1 when both vectors are oriented in the same direction, in which case the loss is 0. If the normals do not agree, then some cost is incurred. This cost corresponds to a heavy-tailed distribution (shown in Figure 2.11(b)) which encourages the surface orientation to match the orientation of the occluding contour at limbs, allows surface normals to violate this assumption at edges.

This occluding-contour prior, when combined with our priors on smooth and isotropic shapes, allows us to easily define an ablation of our entire model that corresponds to a shape-from-contour algorithm: we simply optimize with respect to these shape priors, and ignore our priors on reflectance and illumination, thereby ignoring all but the silhouette of the input image. An example of the output of our shape-from-contour model can be seen in Figure 2.12(b), and this model is evaluated quantitatively against our complete SIRFS model in Section 2.7.

2.3.5 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 [11] and humans [50] 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 [16, 5].

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

$$f_o(Z, \hat{Z}) = \sum_i \left(((Z * b(\sigma_Z))_i - \hat{Z}_i)^2 + \epsilon^2 \right)^{\frac{\gamma_o}{2}} \tag{2.34}$$

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.

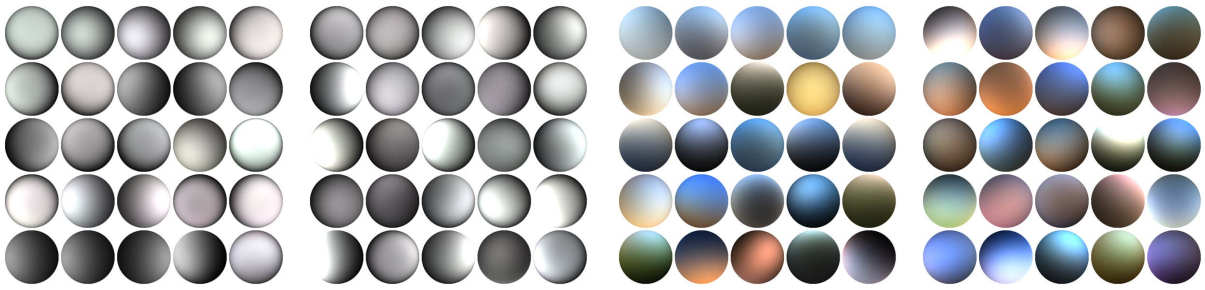
2.4 Priors over Illumination

Because illumination is unknown, we must regularize it during inference. Our prior on illumination is extremely simple: we fit a multivariate Gaussian to the spherical-harmonic illuminations in our training set. During inference, the cost we impose is the (non-normalized) negative log-likelihood under that model:

$$h(L) = \lambda_L (L - \boldsymbol{\mu}_L)^{\mathrm{T}} \Sigma_L^{-1} (L - \boldsymbol{\mu}_L) \tag{2.35}$$

where $\boldsymbol{\mu}_L$ and Σ_L are the parameters of the Gaussian we learned, and λ_L is the multiplier on this prior (learned on the training set).

We use a spherical-harmonic (SH) model of illumination, so L is a 9 (grayscale) or 27 (color, 9 dimensions per RGB channel) dimensional vector. In contrast to traditional SH



(a) "Laboratory" Data/Samples (b) "Natural" Data/Samples

Figure 2.13: We use two datasets: the "laboratory"-style illuminations of the MIT intrinsic images dataset [35] which are harsh, mostly-white, and well-approximated by point sources, and a dataset of "natural" illuminations, which are softer and much more colorful. Shown here are some illuminations from the training sets of our two datasets, and samples from a multivariate Gaussian fit to each training set (our illumination prior from Section 2.4), rendered on Lambertian spheres. In each visualization the illuminations are sorted from least costly (upper left) to most costly (lower right) according to either our "Laboratory" or "Natural" illumination priors.

illumination, we parametrize log-shading rather than shading. This choice makes optimization easier as we don't have to deal with "clamping" illumination at 0, and it allows for easier regularization, as the space of log-shading SH illuminations is surprisingly well-modeled by a simple multivariate Gaussian while the space of traditional SH illumination coefficients is not.

See Figure 2.13 for examples of SH illuminations in our different training sets, as well as samples from our model. The illuminations in Figure 2.13 come from two different datasets (see Section 2.7) for which we build two different priors. We see that our samples look similar to the illuminations in the training set, suggesting that our model fits the data well. The illuminations in these visualizations are sorted by their likelihoods under our priors, which allows us to build an intuition for what these illumination priors encourage. More likely illuminations tend to be lit from the front and are usually less saturated and more ambient, while unlikely illuminations are often lit from unusual angles and tend to exhibit strong shadowing and colors.

2.5 Linearization and Rendering

SIRFS assumes a rendering engine $S(Z, L)$, which takes as input some depth map Z and some spherical harmonic illumination model L and produces a log-shading image. Such a rendering engine is necessary to model shading, as it allows us to relate the shape Z to the reflectance image R. For $S(Z, L)$ we will assume Lambertian reflectance using spherical harmonic illumination, though $S(Z, L)$ is entirely modular, and so the specific choice of what

$S(\cdot)$ to use has little to do with the rest of SIRFS. The only requirement for $S(\cdot)$ is that we can compute it and its analytical derivative efficiently, as our gradient-based optimization requires that we be able to backpropagate gradients with respect to $S(Z, L)$ onto Z and L. In this section we will detail how to calculate $S(Z, L)$ and backpropagate across $S(Z, L)$ 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:

$$N^x = \frac{Z * h_3^x}{B}, \quad N^y = \frac{Z * h_3^y}{B}, \quad N^z = \frac{1}{B}$$
$$B = \sqrt{1 + (Z * h_3^x)^2 + (Z * h_3^y)^2} \tag{2.36}$$

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

$$\begin{aligned} F_{11} &= (1 - N^x \times N^x) \times N^z \\ F_{22} &= (1 - N^y \times N^y) \times N^z \\ F_{13} &= -(N^x \times N^z \times N^z) \\ F_{23} &= -(N^y \times N^z \times N^z) \\ F_{12} &= -(N^x \times N^y \times N^z) \end{aligned} \tag{2.37}$$

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:

$$\begin{aligned} S(\mathbf{n}_i, L) &= [\mathbf{n}_i; 1]^{\mathrm{T}} \mathrm{M} [\mathbf{n}_i; 1] \\ \mathrm{M} &= \begin{bmatrix} c_1 L_9 & c_1 L_5 & c_1 L_8 & c_2 L_4 \\ c_1 L_5 & -c_1 L_9 & c_1 L_6 & c_2 L_2 \\ c_1 L_8 & c_1 L_6 & c_3 L_7 & c_2 L_3 \\ c_2 L_4 & c_2 L_2 & c_2 L_3 & c_4 L_1 - c_5 L_7 \end{bmatrix} \end{aligned} \tag{2.38}$$

$$\begin{aligned} c_1 &= 0.429043 \quad c_2 = 0.511664 \\ c_3 &= 0.743125 \quad c_4 = 0.886227 \quad c_5 = 0.247708 \end{aligned}$$

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:

$$A_i = I_i - S(\mathbf{n}_i, L) \tag{2.39}$$

After calculating $g(A)$ and $\nabla_A g(A)$, (or prior on reflectance, as described in Section 2.2) we can backpropagate the gradient onto Z as follows:

$$\begin{aligned} D_S &= -\nabla_A g(A) \\ D_x &= B^x \times F_{11} + B^y \times F_{12} + B^z \times F_{13} \\ D_y &= B^x \times F_{12} + B^y \times F_{22} + B^z \times F_{23} \\ \nabla_Z g(A) &= (D_S \times D_x) \star h_3^x + (D_S \times D_y) \star h_3^y \end{aligned} \tag{2.40}$$

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:

$$\begin{aligned} J_i = [&c4, 2c_2 N_i^y, 2c_2 N_i^z, 2c_2 N_i^x, 2c_1 N_i^x N_i^y, 2c_1 N_i^y N_i^z, \\ &c_3 N_i^z N_i^z - c_5, 2c_1 N_i^x N_i^z, c_1(N_i^x N_i^x - N_i^y N_i^y)] \end{aligned}$$

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

$$\nabla_L g(A) = J^{\mathrm{T}} D_S \tag{2.41}$$

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

2.6 Optimization

To estimate shape, illumination, and reflectance, we must solve the optimization problem in Equation 2.2. This is a challenging optimization problem, and naive gradient-based optimization with respect to Z and L fails badly. We therefore present an effective multi-scale optimization technique, which is similar in spirit to multigrid methods [75], but extremely general and simple to implement. We will describe our technique in terms of optimizing $a(X)$, where $a(\cdot)$ is some loss function and X is some signal.

Let us define $\mathcal{G}$, which constructs a Gaussian pyramid from a signal. Because Gaussian pyramid construction is a linear operation, we will treat $\mathcal{G}$ as a matrix. Instead of minimizing $a(X)$ directly, we minimize $b(Y)$, where $X = \mathcal{G}^{\mathrm{T}} Y$:

$$\begin{aligned} &[\ell, \nabla_Y \ell] = b(Y): \\ &\quad X \leftarrow \mathcal{G}^{\mathrm{T}} Y \text{ // reconstruct signal} \\ &\quad [\ell, \nabla_X \ell] \leftarrow a(X) \text{ // compute loss \& gradient} \\ &\quad \nabla_Y \ell \leftarrow \mathcal{G} \nabla_X \text{ // backpropagate gradient} \end{aligned} \tag{2.42}$$

We initialize Y to a vector of all 0's, and then solve for $\hat{X} = \mathcal{G}^{\mathrm{T}}\left(\arg\min_Y b(Y)\right)$ using L-BFGS. Any arbitrary gradient-based optimization technique could be used, but L-BFGS worked best in our experience.

The choice of the filter used in constructing our Gaussian pyramid is crucial. We found that 4-tap binomial filters work well, and that the choice of the magnitude of the filter dramatically affects multiscale optimization. If the magnitude is small, then the coefficients of the upper levels of the pyramid are so small that they are effectively ignored, and optimization fails (and in the limit, a filter magnitude of 0 reduces our model to single-scale optimization). Conversely, if the magnitude is large, then the coarse scales of the pyramid are optimized and the fine scales are ignored. The filter that we found worked best is: $\frac{1}{\sqrt{8}}[1, 3, 3, 1]$, which has twice the magnitude of the filter that would normally be used for Gaussian pyramids. This increased magnitude biases optimization towards adjusting coarse scales before fine scales, without preventing optimization from eventually optimizing fine scales. This filter magnitude does not appear to be universally optimal — different tasks appear to have different optimal filter magnitudes. Note that this technique is substantially different from standard coarse-to-fine optimization, in that *all* scales are optimized simultaneously. As a result, we find much lower minima than standard coarse-to-fine techniques, which tend to keep coarse scales fixed when optimizing over fine scales. Optimization is also much faster than comparable coarse-to-fine techniques.

To optimizing Equation 2.2 we initialize Z and L to $\vec{0}$ ($L = \vec{0}$ is equivalent to an entirely ambient, white illumination) and optimize with respect to a vector that is a concatenation of $\mathcal{G}^{\mathrm{T}}Z$ and a whitened version of L. We optimize in the space of whitened illuminations because the Gaussians we learn for illumination mostly describe a low-rank subspace of SH coefficients, and so optimization in the space of unwhitened illumination is ill-conditioned. We pre-compute a whitening transformation for Σ_L and μ_L, and during each evaluation of the loss in gradient descent we unwhiten our whitened illumination, compute the loss and gradient, and backpropagate the gradient onto the whitened illumination. After optimizing Equation 2.2 we have a recovered depth map $\hat{Z}$ and illumination $\hat{L}$, with which we calculate a reflectance image $\hat{R} = I - S(\hat{Z}, \hat{L})$. When illumination is known, L is fixed. Optimizing to near-convergence (which usually takes a few hundred iterations) for a 1-2 megapixel grayscale image takes 1-5 minutes on a 2011 Macbook Pro, using a straightforward Matlab/C implementation. Optimization takes roughly twice as long if the image is color.

We use this same multiscale optimization scheme with L-BFGS to solve the optimization problems in Equations 2.20 and 2.22, though we use different filter magnitudes for the pyramids. Naive single-scale optimization for these problems works poorly.

2.6.1 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 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.

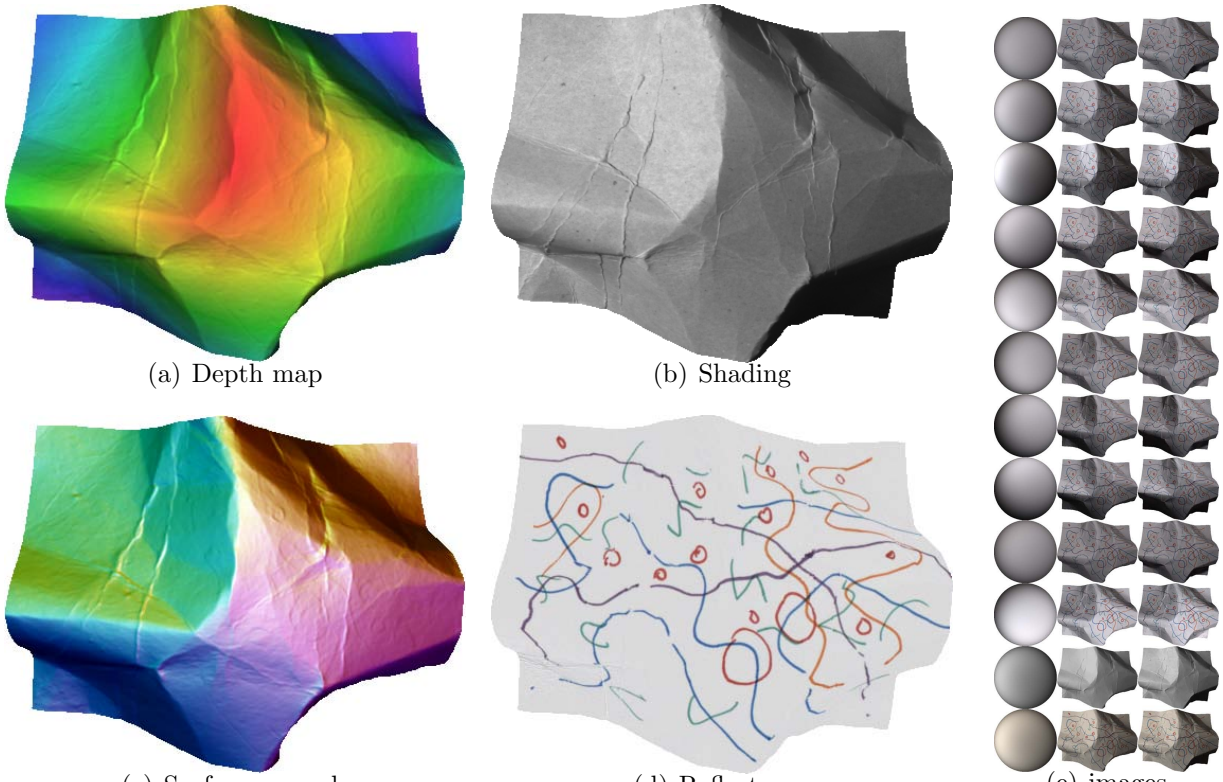
The prior for absolute reflectance can be computed efficiently using the same bilateral-grid trick used for entropy: splat the signal into a histogram, compute the loss of the histogram, and then backpropagate onto the data. For even more efficiency, we can use the same histogram for both entropy and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

2.7 MIT-Berkeley Intrinsic Images Dataset

Quantitatively evaluating the accuracy of our model is challenging, as there are no pre-existing datasets with ground-truth shape, surface normals, shading, reflectance, and illumination. Thankfully, the MIT Intrinsic Images dataset [35] provides ground-truth shading and reflectance for 20 objects (one object per image), and includes many additional images of each object under different illumination conditions. Given this, we have created the MIT-Berkeley Intrinsic Images dataset, an augmented version of the MIT Intrinsic Images dataset in which we have used photometric stereo on the additional images of each object to estimate the shape of each object and the spherical harmonic illumination for each image. Some example objects in our dataset can be seen in Figures 2.14 and 2.15. In all of our experiments, we use the following test-set: cup2, deer, frog2, paper2, pear, potato, raccoon, sun, teabag1, turtle. The other 10 objects are used for training.

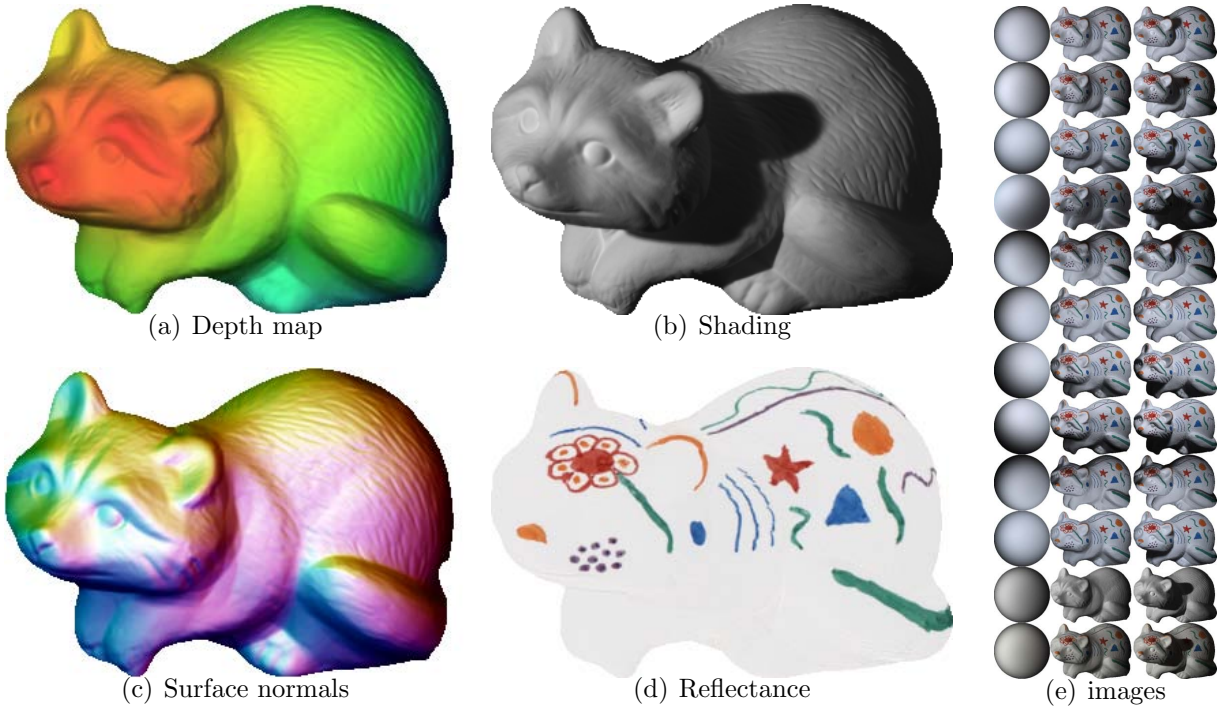
2.7.1 Photometric Stereo

Now 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



(a) Depth map (b) Shading (c) Surface normals (d) Reflectance (e) images

Figure 2.14: An object from our dataset. In 2.14(a), 2.14(b), 2.14(c), and 2.14(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 [35], and the shape and surface normals were produced by our photometric stereo algorithm. In 2.14 we have three columns, where the first contains the images from the MIT Intrinsic Images dataset [35], 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 2.14 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.



(a) Depth map (b) Shading (c) Surface normals (d) Reflectance (e) images

Figure 2.15: Another object from our dataset, shown in the same format as Figure 2.14. Note that our illumination model cannot capture the cast shadows in the input images, which is why our renderings are shadowless.

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[11]. 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.

$$s^*_{i,j} = \exp(I_{i,j} - R_i) \tag{2.43}$$

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:

$$L_j \leftarrow \arg\min_L \sum_i |\exp(S(\mathbf{n}_i, L)) - s^*_{i,j}| \tag{2.44}$$

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 $\mathbf{n}_i$.

$$\mathbf{n}_i \leftarrow \arg\min_{\mathbf{n}} \sum_j |\exp(S(\mathbf{n}, L_j)) - s^*_{i,j}| \tag{2.45}$$

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[11] 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 work, 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 2.14 and 2.15.

2.7.2 Error Metrics

An additional difficulty in evaluation is the choice of error metrics. Constructing error metrics for specific intrinsic scene properties such as a depth map or a reflectance image is challenging, as naive choices such as mean-squared-error often correspond very poorly with the perceptual salience of an error. Additionally, constructing a single error metric that describes all errors in each intrinsic scene property is difficult. We therefore present six different error metrics that have been designed to capture different kinds of important errors for each intrinsic scene property. We will use the geometric mean of six error metrics: two for shape, one for illumination, one for shading, one for reflectance, and one for high-frequency shading and reflectance.

Our first shape error metric is:

$$Z\text{-MAE}(\hat{Z}, Z^*) = \frac{1}{n}\min_{\beta} \sum_{x,y} \left| \hat{Z}_{x,y} - Z^*_{x,y} + b \right| \tag{2.46}$$

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:

$$N\text{-MAE}(\hat{N}, N^*) = \frac{1}{n} \sum_{x,y} \arccos\left(\hat{N}_{x,y} \cdot N^*_{x,y}\right) \tag{2.47}$$

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:

$$L\text{-MSE}(\hat{L}, L^*) = \frac{1}{n}\min_{\alpha} \sum_{x,y} ||\alpha V(\hat{L})_{x,y} - V(L^*)_{x,y}||_2^2 \tag{2.48}$$

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:

$$S\text{-MSE}(\hat{s}, s^*) = \frac{1}{n}\min_{\alpha} \sum_{x,y} \left\|\alpha\hat{s}_{x,y} - s^*_{x,y}\right\|_2^2 \tag{2.49}$$

$$R\text{-MSE}(\hat{r}, r^*) = \frac{1}{n}\min_{\alpha} \sum_{x,y} \left\|\alpha\hat{r}_{x,y} - r^*_{x,y}\right\|_2^2 \tag{2.50}$$

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 work 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 [35], 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:

$$RS\text{-MSE}(\hat{s}, \hat{r}, s^*, r^*) = \frac{1}{2}\left(\frac{e(\hat{s}, s^*)}{e(\hat{s}, 0)} + \frac{e(\hat{r}, r^*)}{e(\hat{r}, 0)}\right) \tag{2.51}$$

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:

$$e(\hat{x}, x^*) = \sum_{w \in W} \min_{\alpha} \|\alpha\hat{x}_w - x^*_w\|_2^2 \tag{2.52}$$

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

$$\min_{\alpha} \|\alpha\hat{x} - x^*\|_2^2 \tag{2.53}$$

The closed-form solution to this problem is:

$$\left\| \left(\frac{\hat{x}^{\mathrm{T}} x^*}{\hat{x}^{\mathrm{T}} \hat{x}} \right) \hat{x} - x^* \right\|_2^2 \tag{2.54}$$

To summarize these individual error metrics, we report an "average" error metric, which is the geometric mean of the previous six error metrics. For each error metric and the average metric, we report the geometric mean of error across the test-set images. The use of the geometric mean prevents the average error from being dominated by individual error metrics with large dynamic ranges, or by particularly challenging images.

2.7.3 Shape From Shading

For the purposes of comparing SIRFS to intrinsic image techniques, we need a shape-from-shading technique for producing depth maps from recovered shading images. Thankfully, 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:

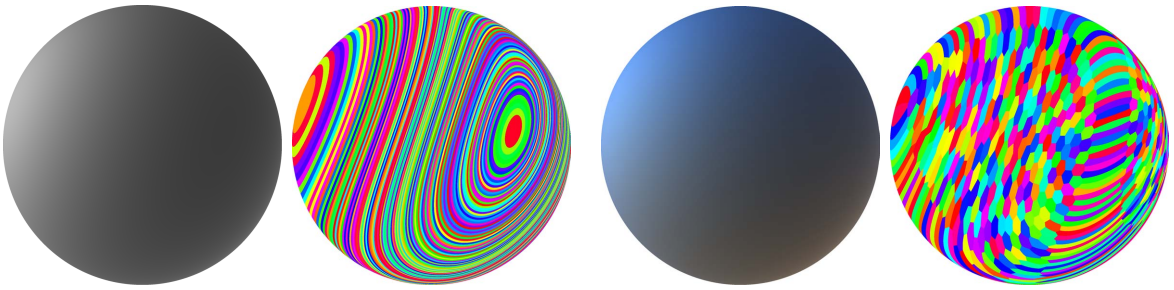
$$\underset{Z}{\text{minimize}} \qquad \lambda|I - S(Z, L)| + f(Z) \tag{2.55}$$

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 as they were defined in SIRFS. 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.

2.7.4 Color / Illumination Conditions

Thought the MIT dataset has a great deal of variety in terms of the kinds of objects used, the illumination in the dataset is very "laboratory"-like — lights are white, and are placed



(a) Achromatic illumination (b) Chromatic illumination

Figure 2.16: Chromatic illumination dramatically helps shape estimation. Achromatic isophotes (K-means clusters of log-RGB values) are very elongated, while chromatic isophotes are usually more tightly localized. Therefore, under achromatic lighting a very wide range of surface orientations appear similar, but under chromatic lighting only similar orientations appear similar.

at only a few locations relative to the object. See Figure 2.13(a) for examples of these "laboratory" illuminations. In contrast, natural illuminations exhibit much more color and variety: the sun is yellow, outdoor shadows are often tinted blue, man-made illuminants have different colors, and indirect illumination from colored objects may cause very colorful illuminations. To acquire some illumination models that are more representative of the variety seen in the natural world, we took all of the environment maps from the sIBL Archive[3], expanded that set of environment maps by shifting and mirroring them and varying their contrast and saturation (saturation is only ever decreased, never increased) and produced spherical harmonic illuminations from the resulting environment maps. After removing similar illuminations, the illuminations were split into training and test sets. See Figure 2.13(b) for examples of these "natural" illuminations. Each object in the MIT dataset was randomly assigned an illumination (such that training illuminations were assigned to training objects, etc), and each object was re-rendered under its new illumination, using that object's ground-truth shape and reflectance. We will refer to this new pseudo-synthetic dataset of naturally illuminated objects as our "natural" illumination dataset, and we will refer to the original MIT images as the "laboratory" illumination dataset. From our experience applying our model to real-world images, these "natural" illuminations appear to be much more representative of the sort of illumination we see in uncontrolled environments, though the dataset is heavily biased towards more colorful illuminations. We attribute this to a photographer's bias towards "interesting" environment maps in the sIBL Archive.

[3] `http://www.hdrlabs.com/sibl/archive.html`

2.7.5 Evaluation

Given our dataset, we will evaluate our model on the task of recovering all intrinsic scene properties from a single image of a masked object, under three different conditions: I: the input is a grayscale image and the illumination is "laboratory"-like, II: the input is a color image and the illumination is "laboratory"-like, and III: the input is a color image and the illumination is "natural". For all tasks, we use the same training/test split, and for each task we tune a different set of hyperparameters on the training set ($\lambda_s, \lambda_e, \lambda_a, \sigma_R, \lambda_k, \lambda_i, \lambda_c, \lambda_o$, and λ_L), and fit a different prior on illumination (as in Section 2.4). Hyperparameters are tuned using coordinate descent to minimize our "average" error metric for the training set. For each task, we compare SIRFS against several intrinsic images algorithms (meant to decompose an image into shading and reflectance components), upon which we've run our previously-detailed shape-from-shading algorithm on the shading image. For the sake of a generous comparison, the SFS algorithm uses our shape priors, which boosts each baseline's performance. We also compare against a "naive" algorithm, which is a baseline in which $Z = \vec{0}$ and $L = \vec{0}$. Because the intrinsic image baselines do not estimate illumination, we use $L = \vec{0}$ as their prediction. We were forced to use different baseline techniques for different tasks, as some baselines do not have code available for running on new imagery, and some code that was designed for color images crashes when run on grayscale images.

We also compare against several ablations of our model in which components have been removed: models B-H omit priors by simply setting their λ hyperparameters to 0, and models I and J omit our multiscale optimization over Z and our whitened optimization over L respectively. Model K is a shape-from-contour technique, in which only our shape-priors are non-zero and $L = \vec{0}$, so the only effective input to the model is the silhouette of the object (for this baseline, the hyperparameters have been completely re-tuned on the training set). We also compare against two extensions of SIRFS: model A+L, in which the ground-truth illumination is known (and fixed during optimization), and model A+S, in which we provide a blurry version of the ground-truth shape (convolved with a Gaussian kernel with $\sigma = 30$) as input and use our prior from Section 2.3.5 to incorporate that information (in all other models, that prior is ignored by setting its λ multiplier to 0). Model S shows the performance of just the blurry ground-truth shape provided as input to model A+S, for reference. The performance of SIRFS relative to some of these baselines and extensions can be see in Tables 2.1-2.3, and in Figures 2.17-2.28.

From Tables 2.1-2.3, we see that SIRFS outperforms all baseline techniques. For grayscale images, the improvement is substantial: our error is roughly half that of the best technique. For color images under "laboratory" illumination, our recovered shading and reflectance images are only slightly better than those of the best-performing intrinsic image technique [33], but our recovered shape and surface normals are significantly better, demonstrating the value of a unified technique over a piecewise system that first does intrinsic images, and then does shape from shading. For color images under "natural" illumination, SIRFS outperforms all baseline models by a very large margin, it is the only model that can reason well about color illumination and (implicitly) color shading. From our ablation study,

we see that each prior contributes positively to performance, though the improvement we get from each prior is greater in the grayscale case than in the color/natural case. This makes sense, as color images under natural illumination contain much more information than in grayscale images, and so the "likelihood" dominates our priors during inference. Our ablation study also shows that our multiscale optimization is absolutely critical to performance. Surprisingly, our shape-from-contour baseline performs very well in terms of our shape/normal error metrics. This is probably just a reflection of the fact that all models are bad at absolute shape reconstruction, due to the inherent ambiguity in shape-from-shading, and so the overly-smooth shape predicted by the shape-from-contour model, by virtue of being smooth and featureless, has a low error relative to the more elaborate depth maps produced by other models. Of course, the shape-from-contour model performs poorly on all other error metrics, as we would expect. This analysis of the inherent difficulty of shape estimation is further demonstrated by model A+S, which includes external shape information, and which therefore performs much better in terms of our shape/normal error metrics, but surprisingly performs similarly to model A (basic SIRFS) in terms of all other error metrics. From the performance of model A+L we see that knowing the illumination of the scene a-priori does not help much when the illumination is laboratory-like, but helps a great deal when the illumination is "natural" — which makes sense, as more-varied illumination simply makes the reconstruction task more difficult. One surprising conclusion we can draw is that, though the intrinsic image baselines perform worse in the presence of "natural" illumination, SIRFS actually performs *better* in natural illumination, as it can exploit color illumination to better disambiguate between shading and reflectance (Figure 2.2), and produce higher-quality shape reconstructions (Figure 2.16). This finding is consistent with recent work regarding shape-from-shading under natural illumination [45]. However, we should mention that some of the improved performance in the natural illumination task may be due to the fact that the images are pseudo-synthethic (their shading images were produced using our spherical-harmonic rendering) and so they are Lambertian and contain no cast shadows.

In Figure 2.29, we demonstrate a simple graphics application using the output of our model, for a color image under laboratory illumination. Given just the output of our model from a single image, we can synthesize novel images in which the shape, reflectance, illumination, or orientation of the object has been changed. The output is not perfect — the absolute shape is often very incorrect, as we saw in Tables 2.1-2.3, which is due to the inherent ambiguity and difficulty in estimating shape from shading. But such shape errors are usually only visible when rotating the object, and this inherent ambiguity in shape perception often works in our favor when only manipulating reflectance, illumination, or fine-scale shape — low-frequency errors in shape-estimation made by our model are often not noticed by human observers, because both the model and the human are bad at using shading to estimate coarse shape.

I. Grayscale Images, Laboratory Illumination

	Algorithm	Z-MAE	N-MAE	S-MSE	R-MSE	RS-MSE	L-MSE	Avg.
(1)	Naive Baseline	25.56	0.7223	0.0571	0.0426	0.0353	0.0484	0.2061
(2)	Retinex [35, 40] + SFS	67.15	0.8342	0.0311	0.0265	0.0289	0.0484	0.2002
(3)	Tappen *et al.* [74] + SFS	41.96	0.7413	0.0354	0.0252	0.0285	0.0484	0.1835
(4)	Shen *et al.* [71] + SFS	45.57	0.8293	0.0493	0.0427	0.0436	0.0484	0.2348
(A)	SIRFS	31.00	0.5343	**0.0156**	**0.0177**	**0.0209**	**0.0103**	**0.0998**
(B)	SIRFS, no R-smoothness	27.25	0.5361	0.0267	0.0255	0.0290	0.0152	0.1279
(C)	SIRFS, no R-parsimony	23.53	0.4862	0.0224	0.0261	0.0228	0.0167	0.1170
(D)	SIRFS, no R-absolute	24.02	0.5023	0.0190	0.0201	0.0222	0.0122	0.1037
(E)	SIRFS, no Z-smoothness	29.05	0.5783	0.0241	0.0227	0.0337	0.0125	0.1254
(F)	SIRFS, no Z-isotropy	98.07	0.7560	0.0200	0.0198	0.0268	0.0104	0.1419
(G)	SIRFS, no Z-contour	34.29	0.7676	0.0208	0.0207	0.0232	0.0231	0.1351
(H)	SIRFS, no L-gaussian	26.75	0.5929	0.0270	0.0212	0.0327	0.1940	0.1964
(I)	SIRFS, no Z-multiscale	25.58	0.7233	0.0571	0.0426	0.0353	0.0414	0.2009
(J)	SIRFS, no L-whitening	33.93	0.5837	0.0207	0.0208	0.0256	0.0119	0.1171
(K)	Shape-from-Contour	**18.96**	**0.4192**	0.0571	0.0426	0.0353	0.0484	0.1791
(S)	shape observation	4.83	0.1952	-	-	-	-	-
(A+S)	SIRFS + shape observation	3.72	0.2414	0.0128	0.0176	0.0210	0.0096	0.0586
(A+L)	SIRFS + known illumination	27.32	0.4944	0.0175	0.0179	0.0225	-	-

Table 2.1: Performance on our "Grayscale/Laboratory" dataset variant. We evaluate the complete SIRFS model, and we compare SIRFS to several baseline techniques, several ablations of SIRFS, and two extensions in which additional information is provided.

II. Color Images, Laboratory Illumination

	Algorithm	Z-MAE	N-MAE	S-MSE	R-MSE	RS-MSE	L-MSE	Avg.
(1)	Naive Baseline	25.56	0.7223	0.0577	0.0455	0.0354	0.0489	0.2092
(2)	Retinex [35, 40] + SFS	85.34	0.8056	0.0204	0.0186	0.0163	0.0489	0.1658
(3)	Tappen *et al.* [74] + SFS	41.96	0.7413	0.0361	0.0379	0.0347	0.0489	0.2040
(4)	Shen *et al.* [71] + SFS	55.95	0.8529	0.0528	0.0458	0.0398	0.0489	0.2466
(5)	Gehler *et al.* [33] + SFS	53.36	0.6844	0.0106	0.0101	0.0131	0.0489	0.1166
(A)	SIRFS	19.24	**0.3914**	**0.0064**	**0.0098**	**0.0125**	0.0096	**0.0620**
(B)	SIRFS, no R-smoothness	19.23	0.4046	0.0125	0.0163	0.0214	0.0092	0.0824
(C)	SIRFS, no R-parsimony	19.45	0.4312	0.0096	0.0149	0.0140	0.0091	0.0731
(D)	SIRFS, no R-absolute	22.98	0.4288	0.0085	0.0113	0.0135	0.0095	0.0704
(E)	SIRFS, no Z-smoothness	19.28	0.4367	0.0114	0.0116	0.0219	**0.0088**	0.0773
(F)	SIRFS, no Z-isotropy	84.08	0.7013	0.0117	0.0128	0.0160	0.0103	0.1063
(G)	SIRFS, no Z-contour	32.59	0.7351	0.0103	0.0146	0.0173	0.0444	0.1186
(H)	SIRFS, no L-gaussian	20.81	0.4631	0.0199	0.0140	0.0183	0.1272	0.1358
(I)	SIRFS, no Z-multiscale	25.62	0.7237	0.0574	0.0453	0.0353	0.0401	0.2022
(J)	SIRFS, no L-whitening	24.96	0.4766	0.0106	0.0156	0.0188	0.0138	0.0894
(K)	Shape-from-Contour	**18.96**	0.4192	0.0577	0.0455	0.0354	0.0489	0.1818
(S)	shape observation	4.83	0.1952	-	-	-	-	-
(A+S)	SIRFS + shape observation	3.40	0.2126	0.0070	0.0111	0.0153	0.0063	0.0420
(A+L)	SIRFS + known illumination	18.58	0.3761	0.0076	0.0120	0.0146	-	-

Table 2.2: Performance on our "Color/Laboratory" dataset variant. We evaluate the complete SIRFS model, and we compare SIRFS to several baseline techniques, several ablations of SIRFS, and two extensions in which additional information is provided.

III. Color Images, Natural Illumination

	Algorithm	Z-MAE	N-MAE	S-MSE	R-MSE	RS-MSE	L-MSE	Avg.
(1)	Naive Baseline	25.56	0.7223	0.0283	0.0266	0.0125	0.0371	0.1364
(2)	Retinex [35, 40] + SFS	26.76	0.5851	0.0174	0.0174	0.0083	0.0371	0.1066
(3)	Tappen *et al.* [74] + SFS	53.87	0.7255	0.0255	0.0280	0.0268	0.0371	0.1740
(4)	Gehler *et al.* [33] + SFS	37.66	0.6398	0.0162	0.0150	0.0105	0.0371	0.1149
(A)	SIRFS	28.21	0.4057	0.0055	0.0046	**0.0036**	0.0103	0.0469
(B)	SIRFS, no R-smoothness	28.41	0.4192	0.0061	0.0057	0.0062	0.0104	0.0546
(C)	SIRFS, no R-parsimony	28.90	0.4184	0.0073	0.0064	0.0041	0.0107	0.0540
(D)	SIRFS, no R-absolute	20.63	**0.3538**	0.0068	0.0058	0.0039	0.0091	**0.0466**
(E)	SIRFS, no Z-smoothness	24.68	0.4441	0.0087	0.0062	0.0095	0.0099	0.0618
(F)	SIRFS, no Z-isotropy	50.49	0.4015	**0.0046**	**0.0039**	0.0037	**0.0086**	0.0475
(G)	SIRFS, no Z-contour	41.27	0.7036	0.0094	0.0083	0.0062	0.0256	0.0843
(H)	SIRFS, no L-gaussian	20.22	0.3937	0.0100	0.0088	0.0075	0.0483	0.0796
(I)	SIRFS, no Z-multiscale	25.64	0.7205	0.0279	0.0279	0.0124	0.0291	0.1316
(J)	SIRFS, no L-whitening	51.74	0.9430	0.0140	0.0106	0.0066	0.0777	0.1246
(K)	Shape-from-Contour	**19.55**	0.4253	0.0283	0.0266	0.0125	0.0371	0.1194
(S)	shape observation	4.83	0.1952	-	-	-	-	-
(A+S)	SIRFS + shape observation	3.17	0.1471	0.0034	0.0032	0.0030	0.0049	0.0206
(A+L)	SIRFS + known illumination	10.28	0.1957	0.0018	0.0014	0.0022	-	-

Table 2.3: Performance on our "Color/Natural" dataset variant. We evaluate the complete SIRFS model, and we compare SIRFS to several baseline techniques, several ablations of SIRFS, and two extensions in which additional information is provided.

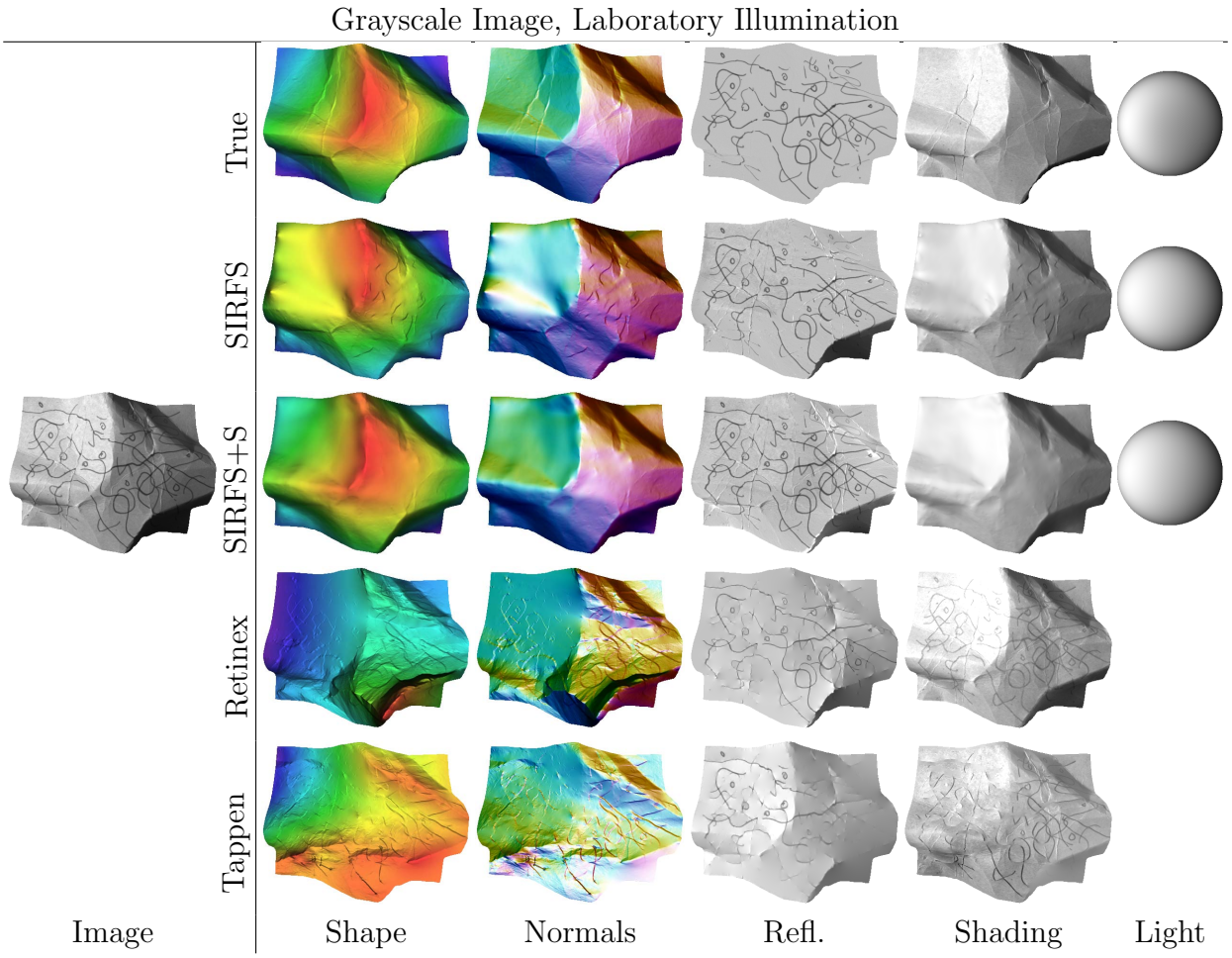


Figure 2.17: An image from our dataset, for our "Grayscale/Laboratory" dataset variant. We present the ground-truth, the output of SIRFS, the output of SIRFS+S (which uses external shape information), and the two best-performing intrinsic image techniques (for which we do SFS on the recovered shading to recover shape).

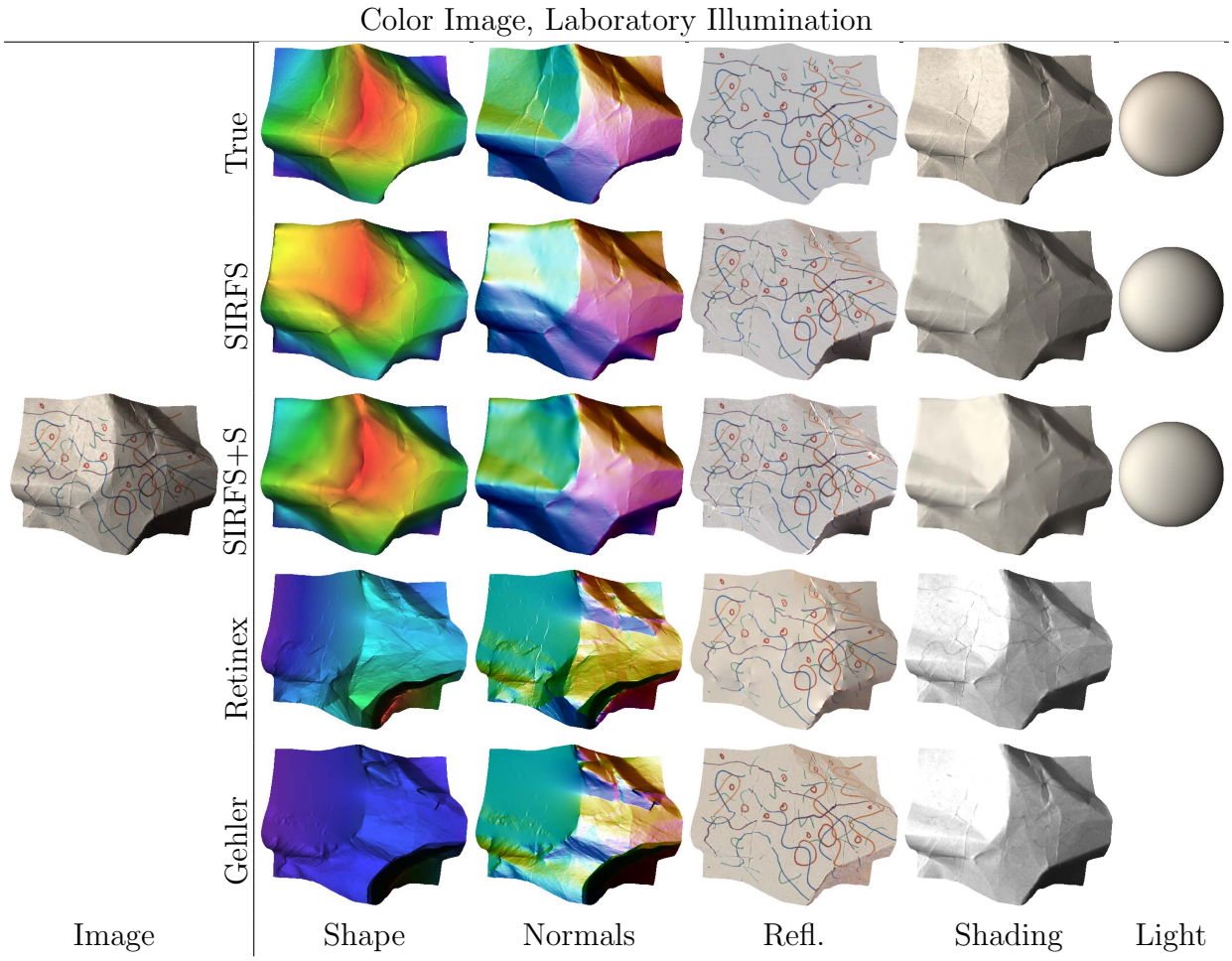


Figure 2.18: An image from our dataset, for our "Color/Laboratory" dataset variant. We present the ground-truth, the output of SIRFS, the output of SIRFS+S (which uses external shape information), and the two best-performing intrinsic image techniques (for which we do SFS on the recovered shading to recover shape).

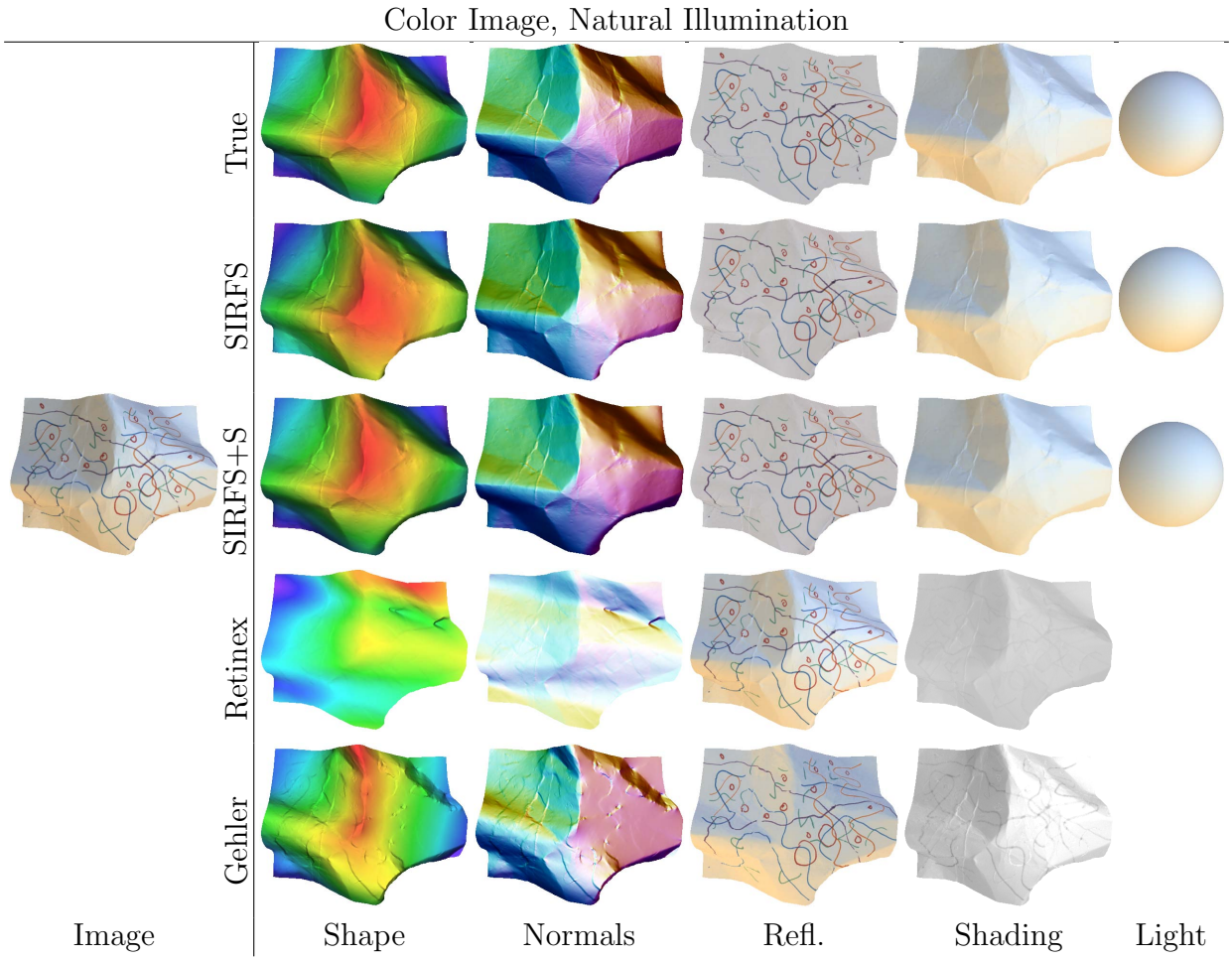


Figure 2.19: An image from our dataset, for our "Color/Natural" dataset variant. We present the ground-truth, the output of SIRFS, the output of SIRFS+S (which uses external shape information), and the two best-performing intrinsic image techniques (for which we do SFS on the recovered shading to recover shape).

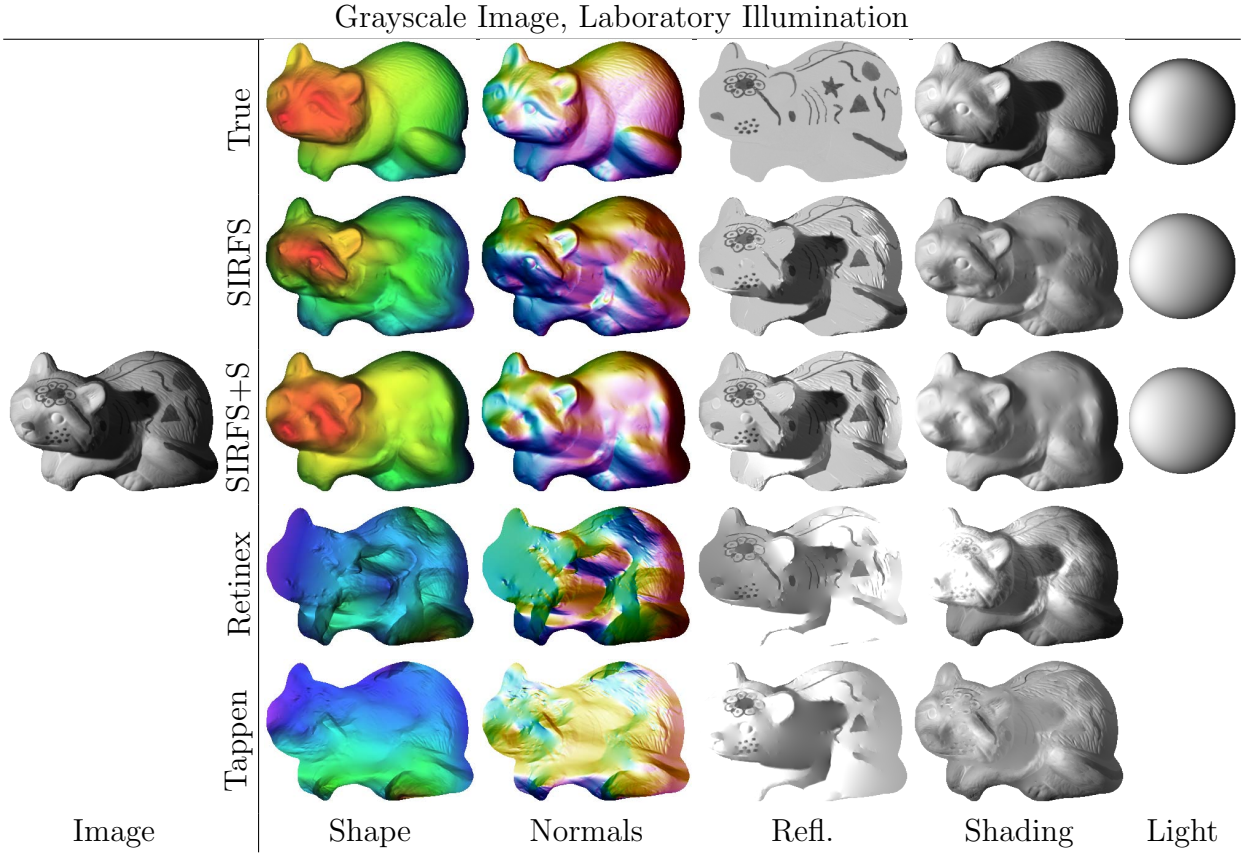


Figure 2.20: An image from our dataset, for our "Grayscale/Laboratory" dataset variant. We present the ground-truth, the output of SIRFS, the output of SIRFS+S (which uses external shape information), and the two best-performing intrinsic image techniques (for which we do SFS on the recovered shading to recover shape).

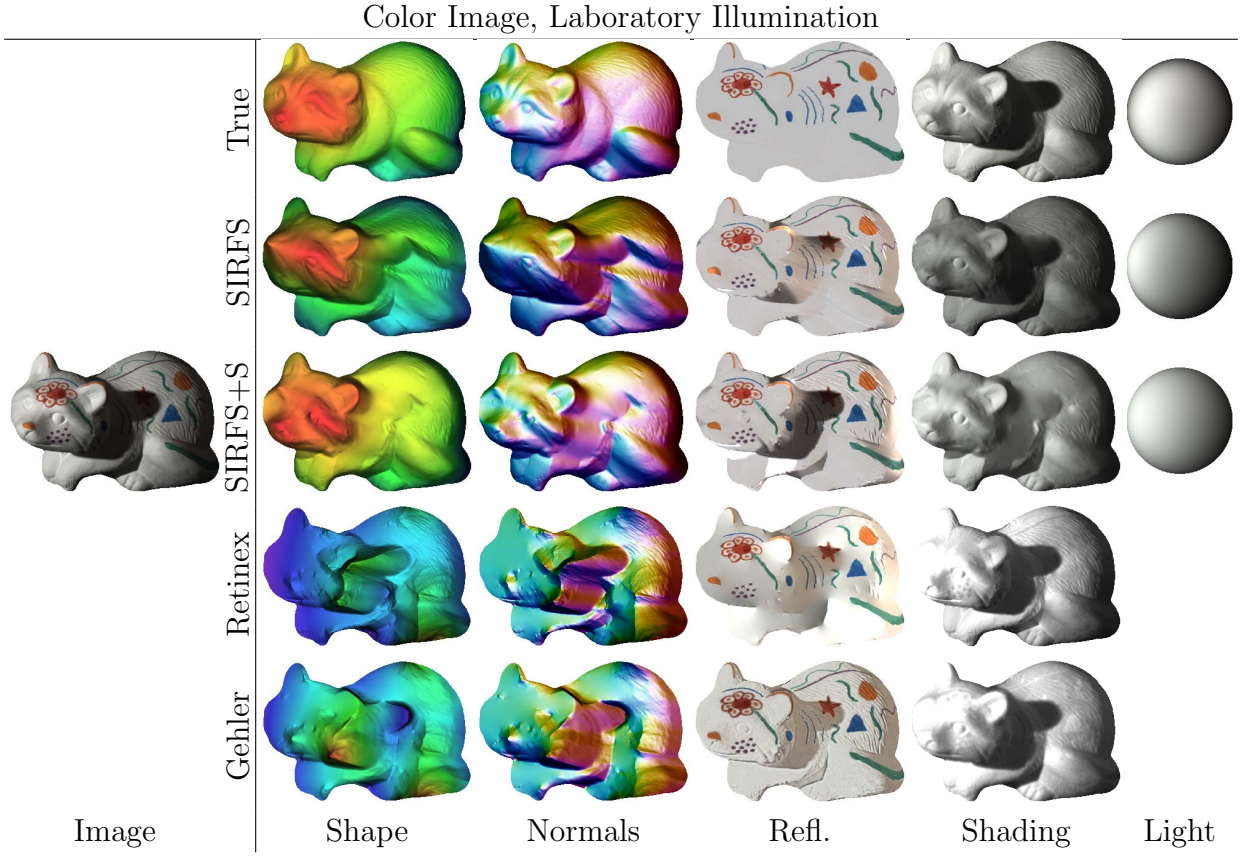


Figure 2.21: An image from our dataset, for our "Color/Laboratory" dataset variant. We present the ground-truth, the output of SIRFS, the output of SIRFS+S (which uses external shape information), and the two best-performing intrinsic image techniques (for which we do SFS on the recovered shading to recover shape).

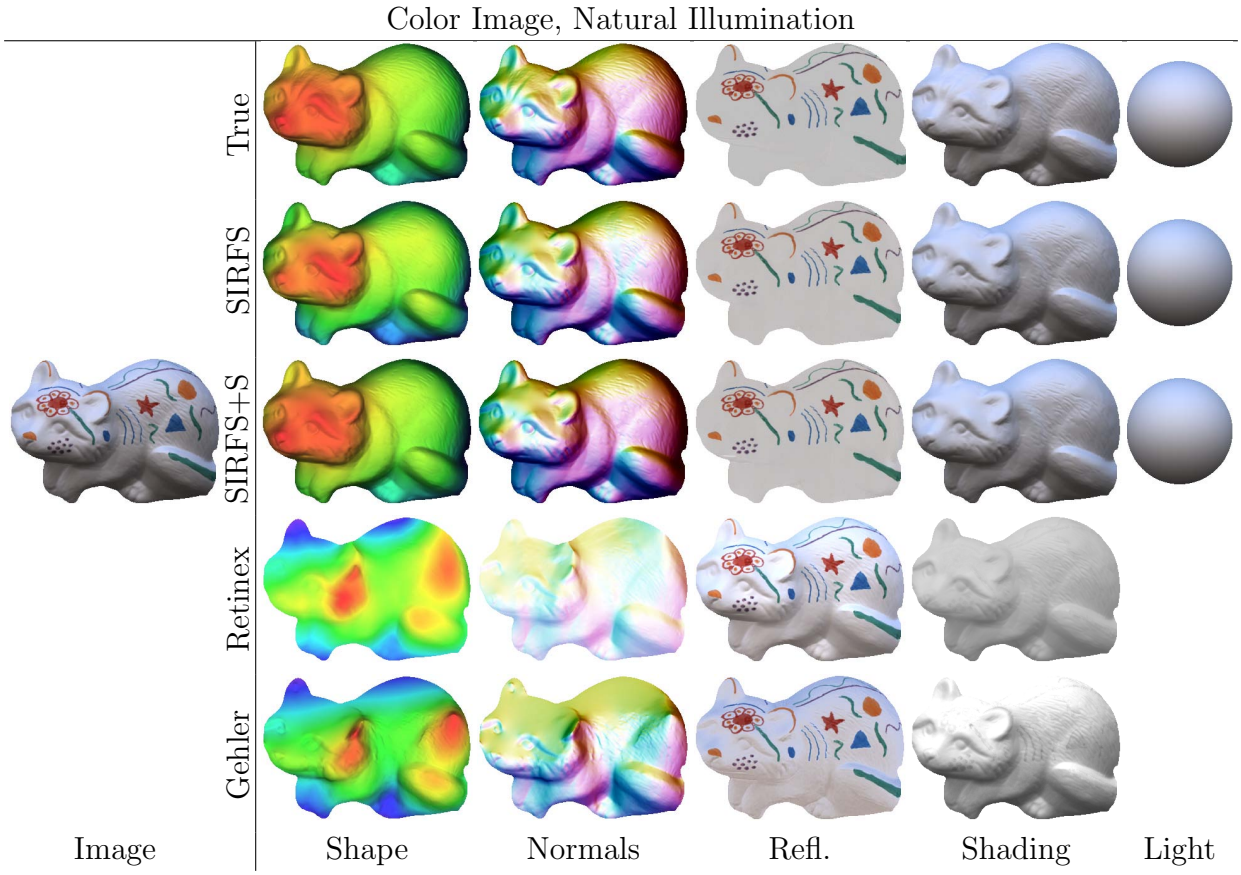


Figure 2.22: An image from our dataset, for our "Color/Natural" dataset variant. We present the ground-truth, the output of SIRFS, the output of SIRFS+S (which uses external shape information), and the two best-performing intrinsic image techniques (for which we do SFS on the recovered shading to recover shape).

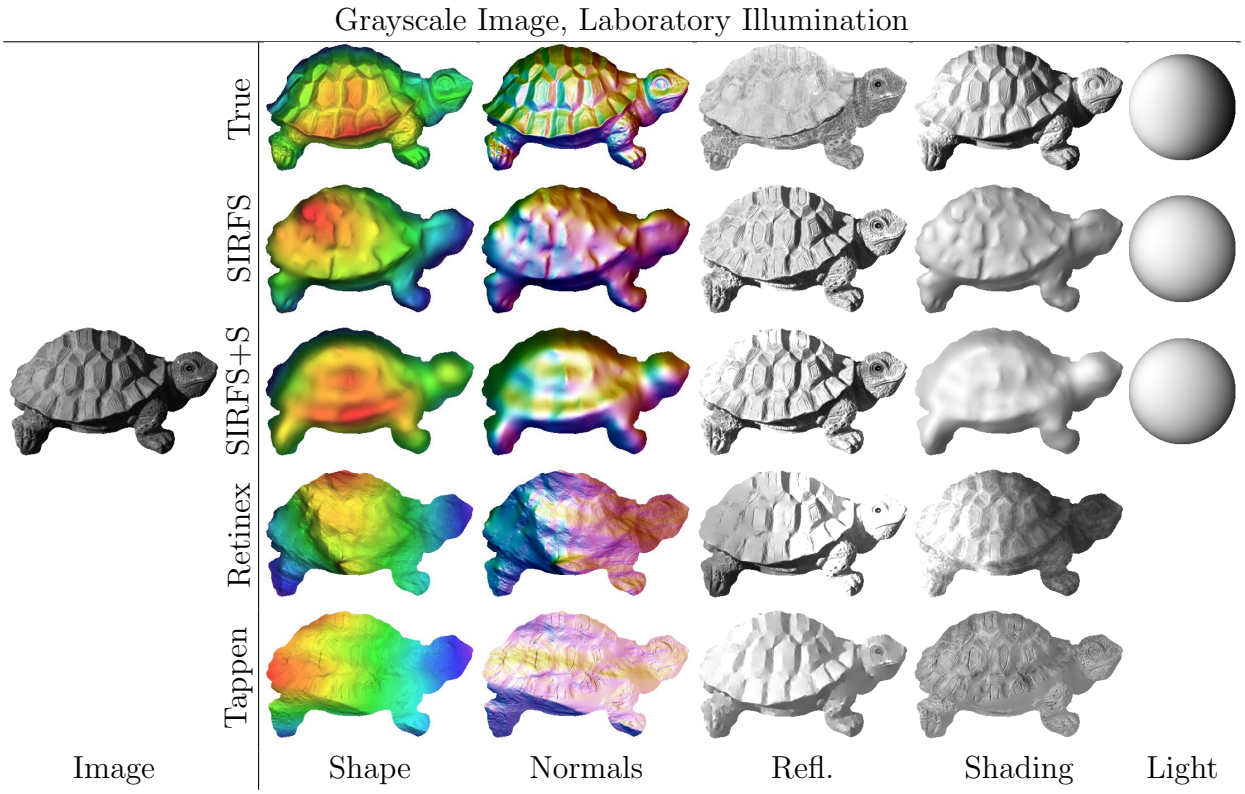


Figure 2.23: An image from our dataset, for our "Grayscale/Laboratory" dataset variant. We present the ground-truth, the output of SIRFS, the output of SIRFS+S (which uses external shape information), and the two best-performing intrinsic image techniques (for which we do SFS on the recovered shading to recover shape).

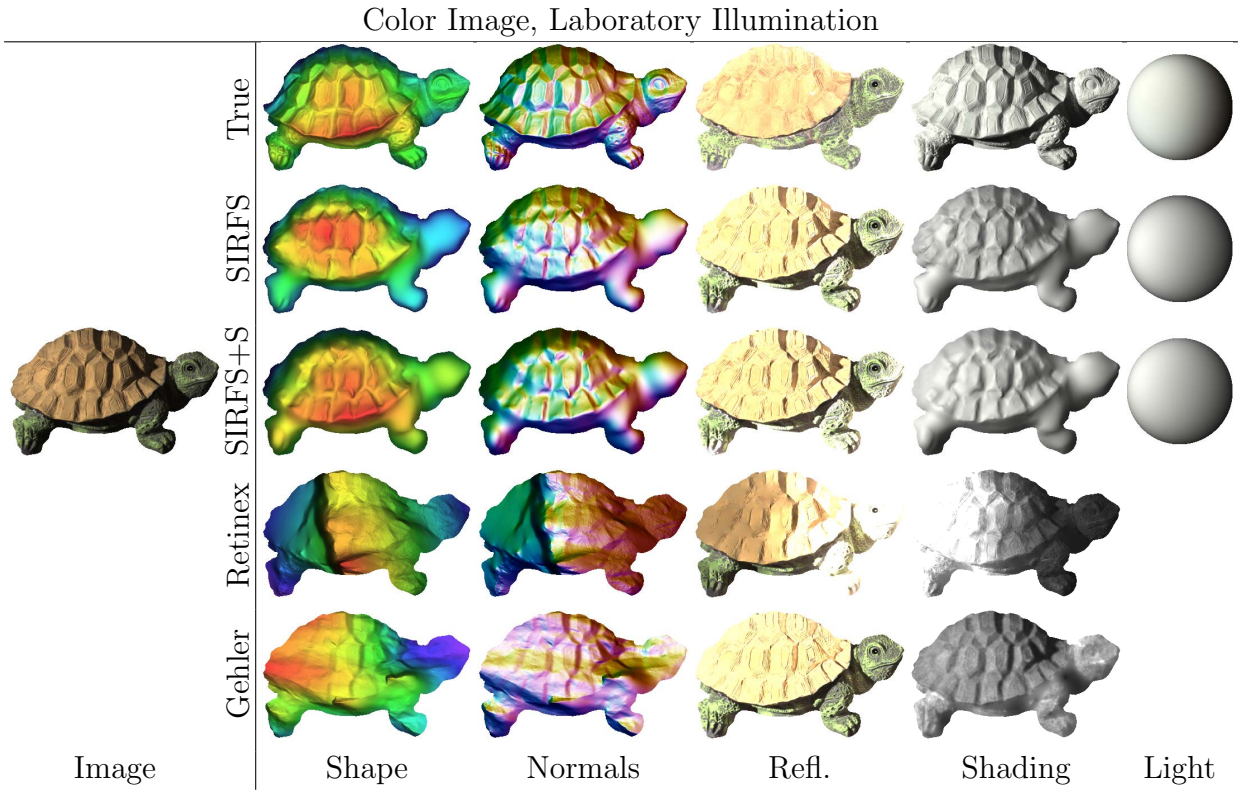


Figure 2.24: An image from our dataset, for our "Color/Laboratory" dataset variant. We present the ground-truth, the output of SIRFS, the output of SIRFS+S (which uses external shape information), and the two best-performing intrinsic image techniques (for which we do SFS on the recovered shading to recover shape).

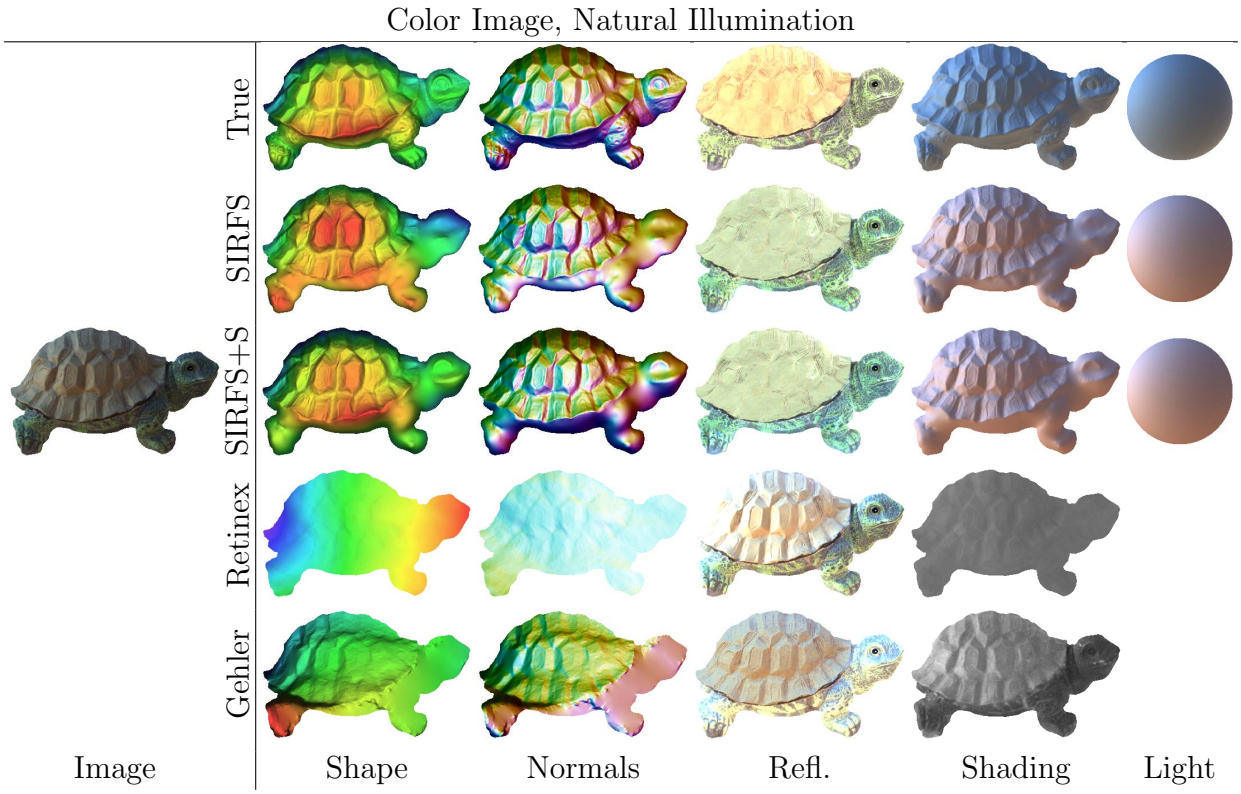


Figure 2.25: An image from our dataset, for our "Color/Natural" dataset variant. We present the ground-truth, the output of SIRFS, the output of SIRFS+S (which uses external shape information), and the two best-performing intrinsic image techniques (for which we do SFS on the recovered shading to recover shape).

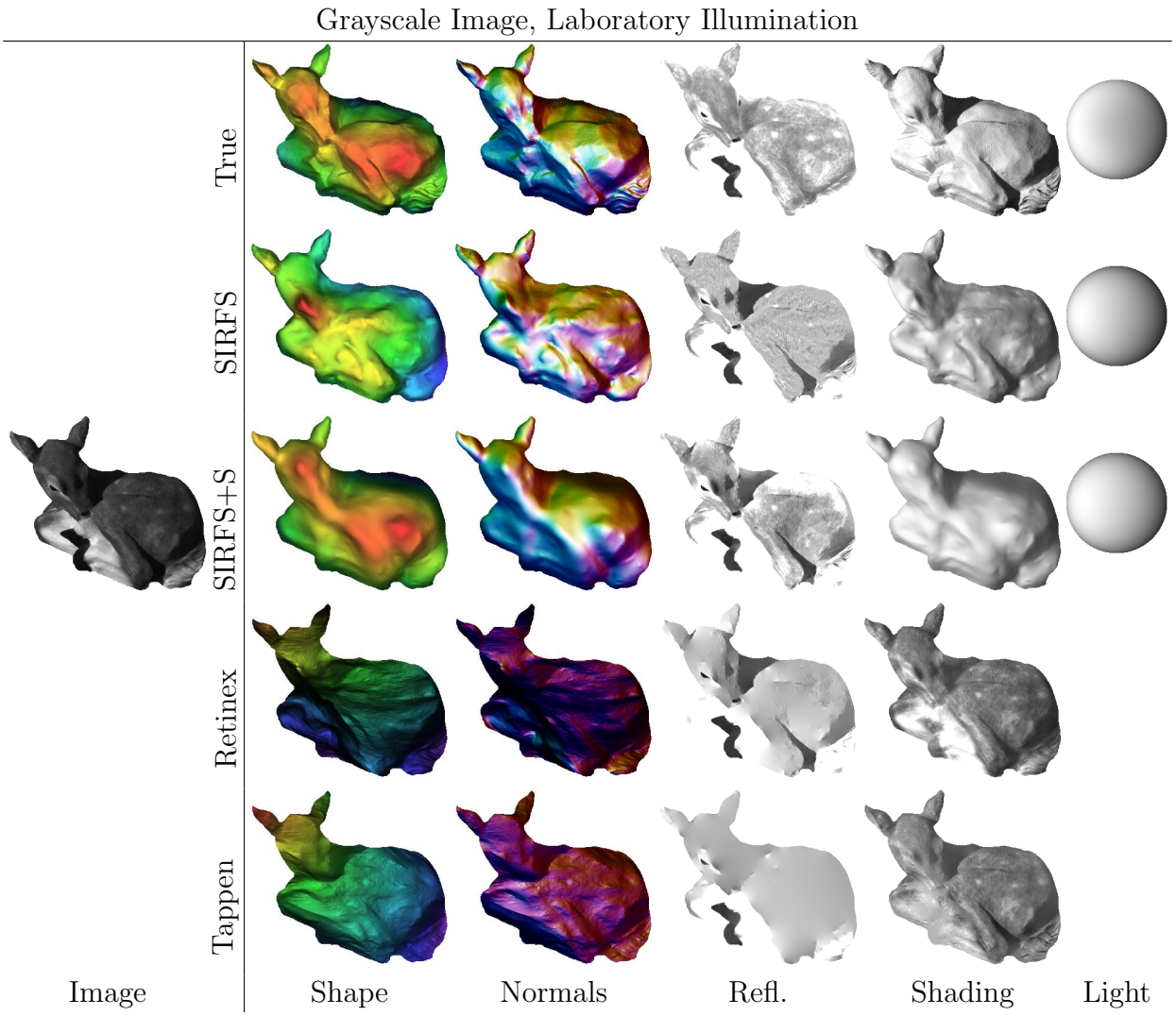


Figure 2.26: An image from our dataset, for our "Grayscale/Laboratory" dataset variant. We present the ground-truth, the output of SIRFS, the output of SIRFS+S (which uses external shape information), and the two best-performing intrinsic image techniques (for which we do SFS on the recovered shading to recover shape).

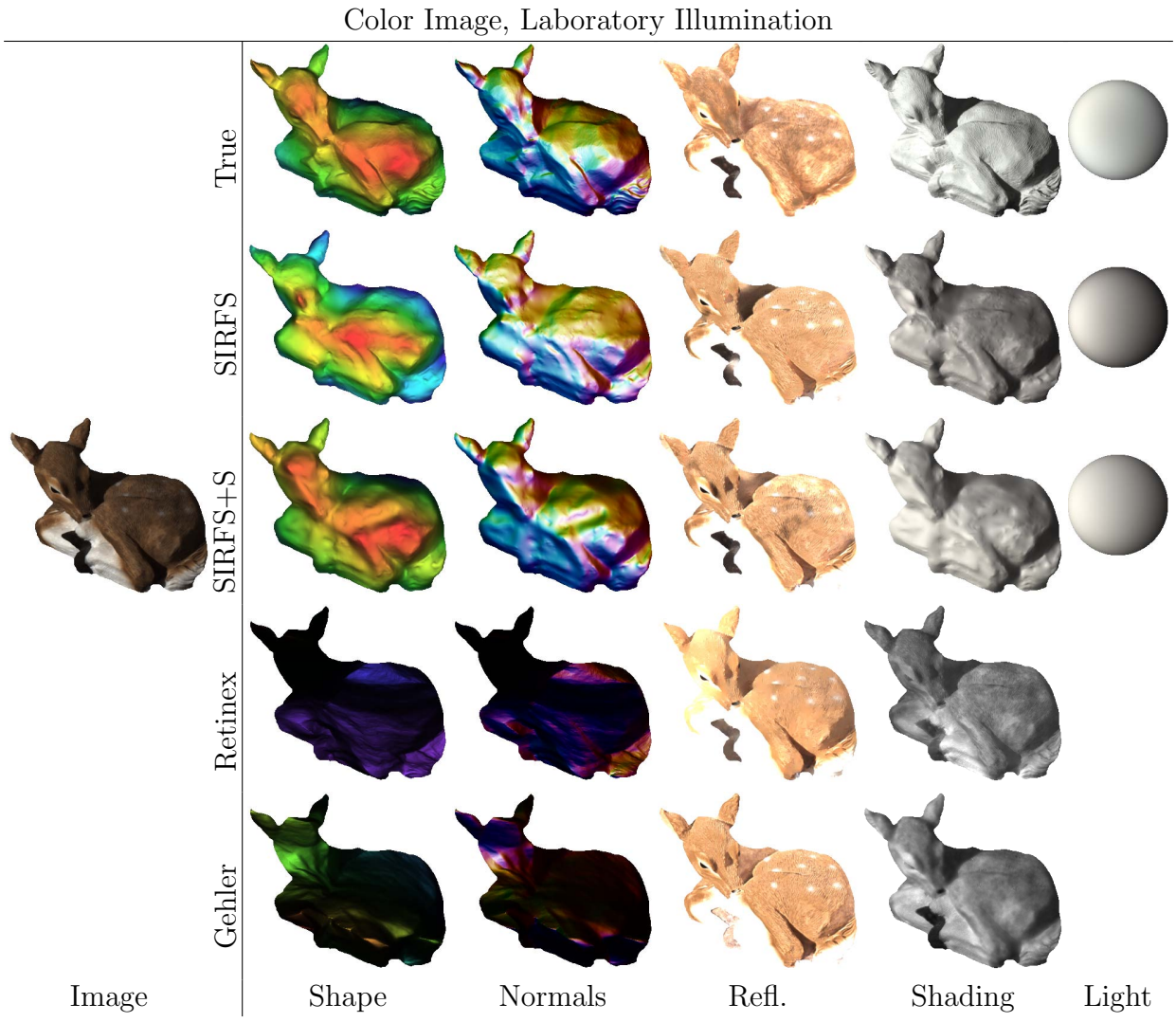


Figure 2.27: An image from our dataset, for our "Color/Laboratory" dataset variant. We present the ground-truth, the output of SIRFS, the output of SIRFS+S (which uses external shape information), and the two best-performing intrinsic image techniques (for which we do SFS on the recovered shading to recover shape).

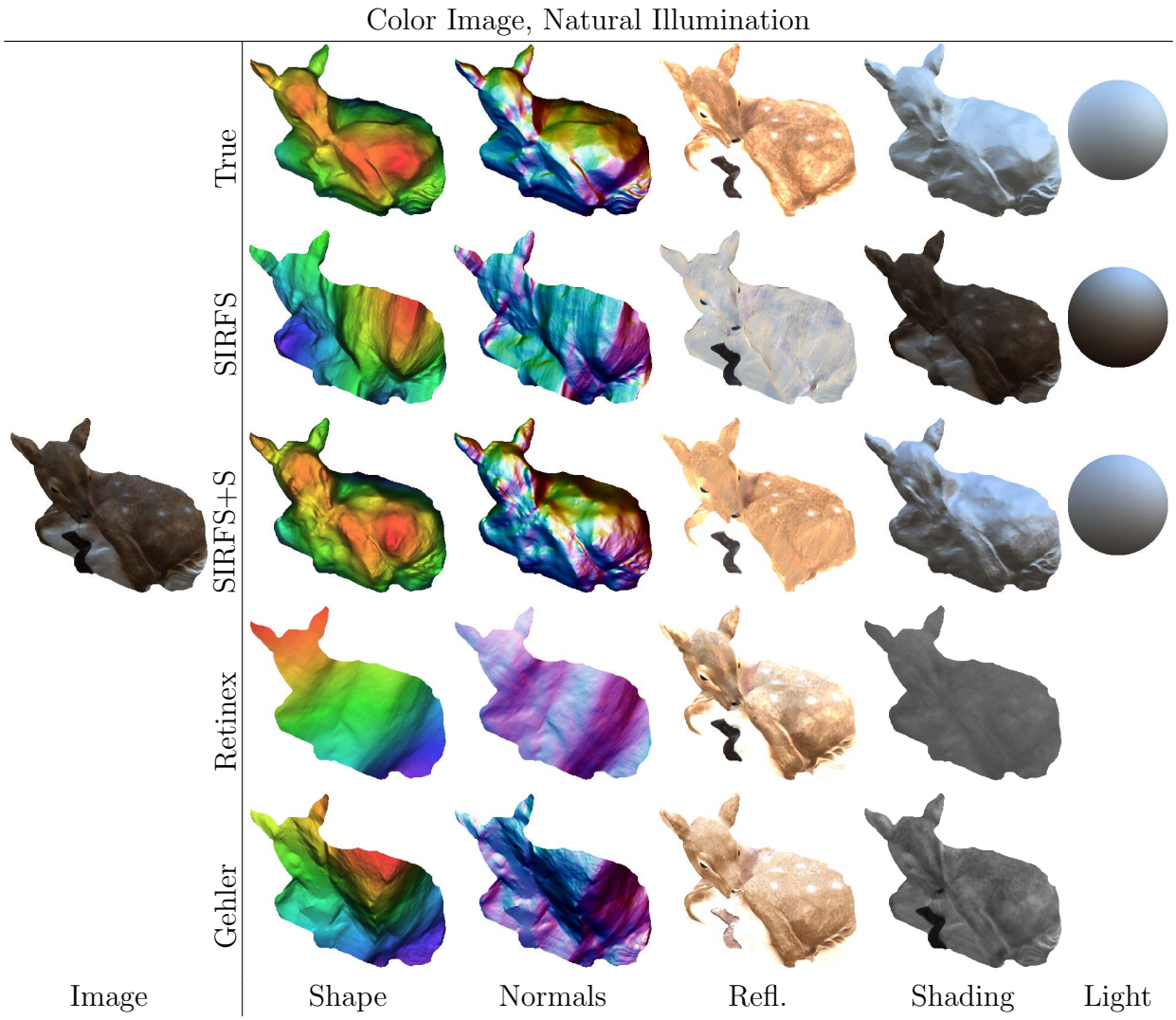
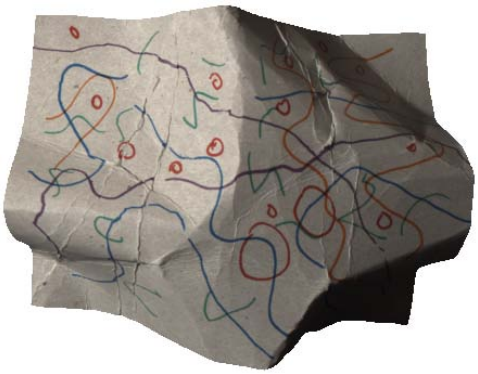
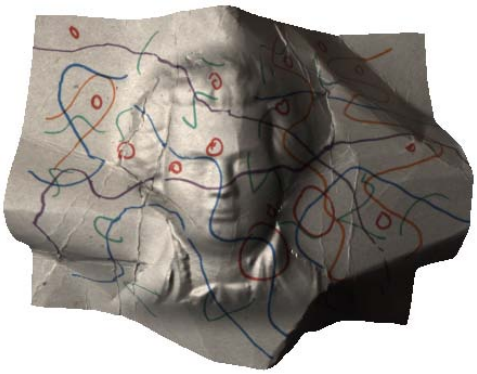


Figure 2.28: An image from our dataset, for our "Color/Natural" dataset variant. We present the ground-truth, the output of SIRFS, the output of SIRFS+S (which uses external shape information), and the two best-performing intrinsic image techniques (for which we do SFS on the recovered shading to recover shape).

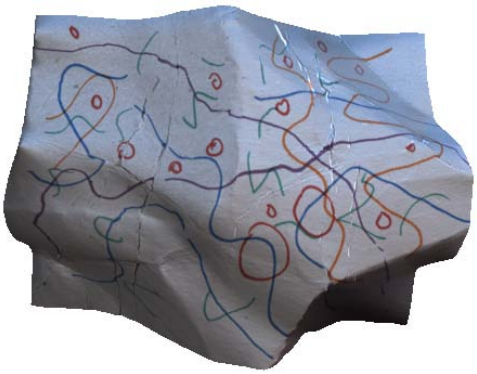


(a) Input Image

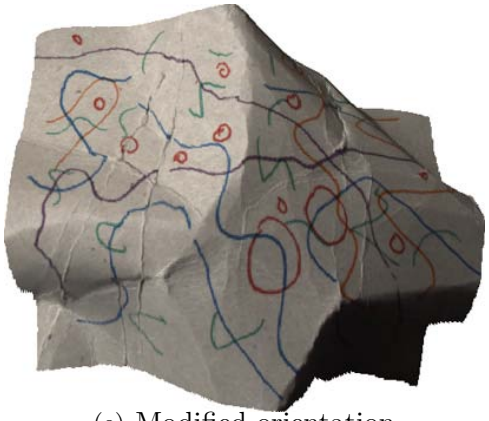


(b) Modified shape

(c) Modified reflectance



(d) Modified light



(e) Modified orientation

Figure 2.29: Our system has obvious graphics applications. Given only a single image, we can estimate an object's shape, reflectance, or illumination, modify any of those three scene properties (or simply rotate the object), and then re-render the object.

2.8 Real-World Images

Though our model quantitatively performs very well on the MIT-Berkeley Intrinsic Images dataset, this dataset is not very representative of the variety of natural objects in the world — materials are very Lambertian, many reflectances are very synthetic-looking, and illumination is not very varied. We therefore present an additional experiment in which we ran our model on arbitrary masked images of natural objects. We acquired many images (some with an iPhone camera, some with a DSLR, some downloaded from the internet), manually cropped the object in the photo, and used them as input to our model. In Figure 2.33 we visualize the output of our model: the recovered shape, normals, reflectance, shading, and illumination, a synthesized view of the object from a different angle, and a synthesized rendering of the object using a different (randomly generated) illumination. We did two experiments: one in which we used a grayscale version of the input image and our laboratory illumination model, and one in which we used the color input image and our natural illumination model. We use the same code and hyperparameters for all images in the two constituent tasks, where our hyperparameters are identical to those used in the previous experiments with the MIT-Berkeley Intrinsic Images dataset.

We see that our model is often able to produce extremely compelling shading and reflectance images, and qualitatively correct illumination. Our recovered shape and surface normals are often somewhat wrong, as evidenced by the new synthesized views of each object, but our "relit" objects are often very compelling. The most common mistakes made in shading/reflectance estimation are usually due to our model assuming that the dominant color of the object is due to illumination, not reflectance (such as in the two pictures of faces) which we believe is due to biases in our training data towards white reflectances and colorful illumination.

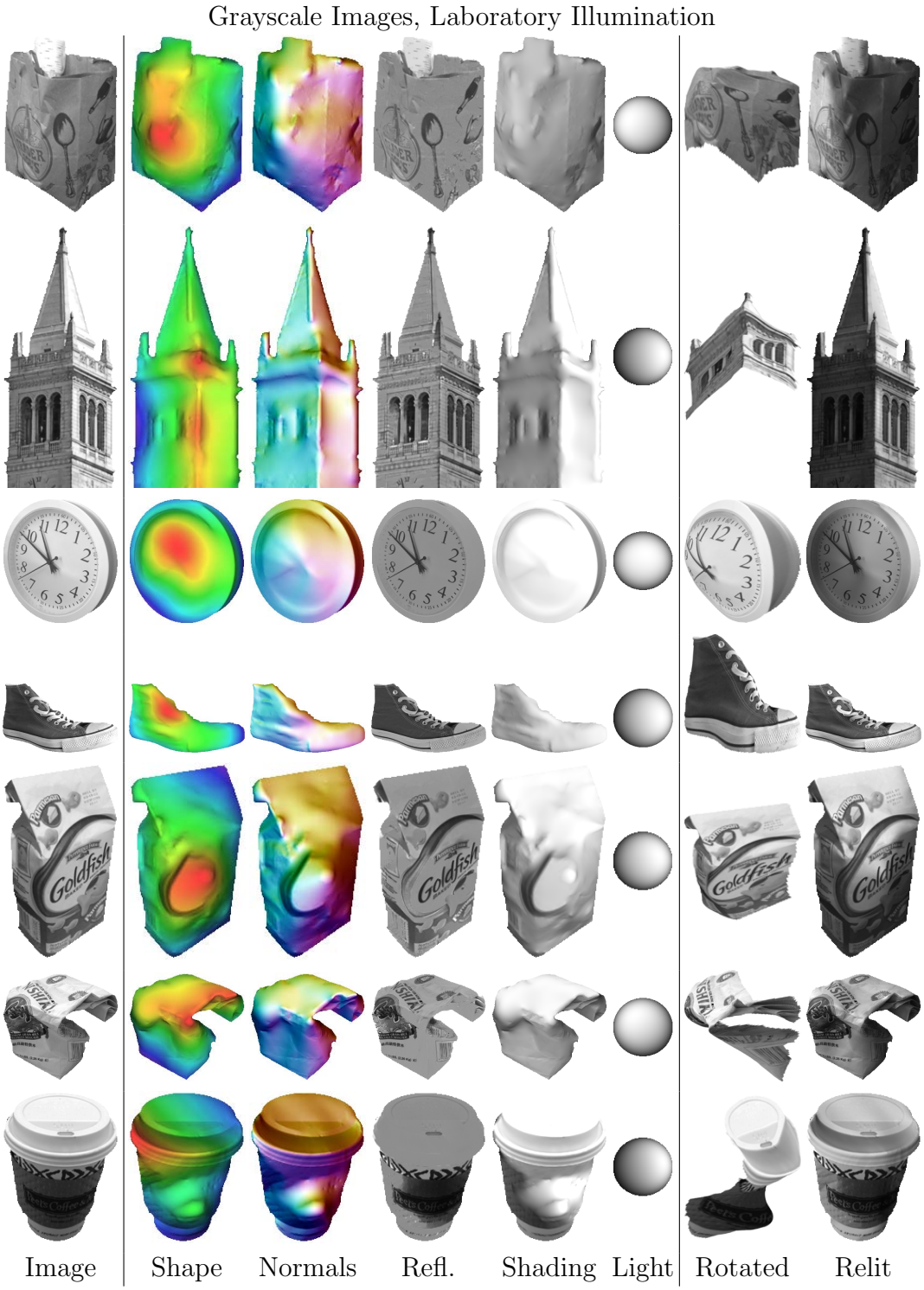


Figure 2.30: Some examples of the output of our model on real, manually cropped grayscale images of objects.

Figure 2.31: More examples of the output of our model on real, manually cropped grayscale images of objects.

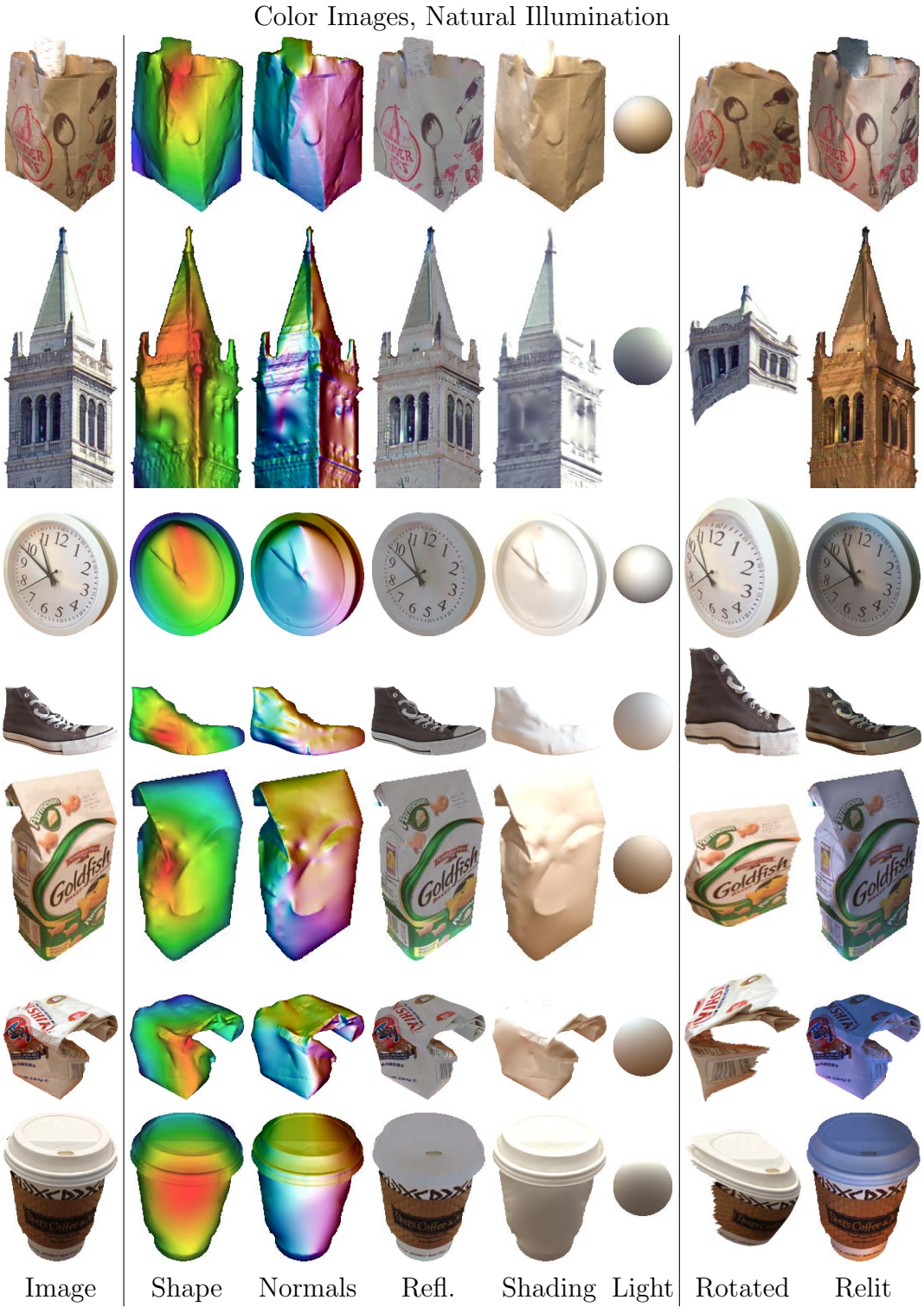


Figure 2.32: Some examples of the output of our model on real, manually cropped color images of objects.

Figure 2.33: More examples of the output of our model on real, manually cropped color images of objects.

2.9 Limitations

Of course, our model has some limitations. Because shading is an inherently poor cue for low-frequency shape estimation [11, 48] SIRFS often makes mistakes in coarse shape estimation. To address this, we have presented a method for incorporating some external observation of shape, such as one from a stereo algorithm or a depth sensor, and we have demonstrated that by incorporating some low-frequency external shape observation (such as what a stereo algorithm or a depth sensor would provide) we can produce high-quality shape estimates. We assume that materials are Lambertian, which is often a reasonable approximation but can causes problems for objects with specularities. Thankfully, because of the modular nature of our algorithm, our simple Lambertian rendering engine can easily be replaced by a more sophisticated model. Another limitation of our technique is that our priors on shape and reflectance are independent of the category of object present in the scene. We see this as a strength of our model, as it means that our priors are general enough to generalize across object categories, but presumably an extension of our model which uses object recognition techniques to produce class-specific priors should perform better.

The most glaring limitation of SIRFS is that the images we use each consist of a single, masked object, while real-world natural scenes contain severe occlusion and support relationships. We also assume illumination is global, and we ignore illumination issues such as cast shadows, mutual illumination, or other sources of spatially-varying illumination [29, 30]. To address these two issues of occlusion and spatially-varying illumination in natural scenes, in Chapter 3 we present an extension of SIRFS that is integrated with image segmentation techniques, by generalizing SIRFS to a mixture model of shapes and lights which are embedded in a soft segmentation of a scene.

Chapter 3

Scene-SIRFS

SIRFS is severely limited by its assumption that input images are segmented images of single objects, illuminated under a single global model of illumination. Natural images, in contrast, contain many shapes which may occlude or support one another, as well as complicated, spatially-varying illumination in the form of shadows, attenuation, and interreflection.

In this chapter, we address the problem of inferring a *mixture* of shapes and a *mixture* of illuminations (and implicitly, a shading image and a reflectance image) which explain a natural *scene*. Initially, this problem may seem trivial: why not use segmentation techniques to decompose an image into its constituent objects or illuminations, and then apply SIRFS to each segment? But this is a classic "chicken-or-the-egg" problem, as we cannot reliably segment an image into its constituent shapes and illuminations without first inferring shape and illumination, and vice versa. Additionally, regions of a scene viewed in isolation are often ambiguous, which suggests that information must be shared between regions. We must therefore unify the problems of reconstruction (inferring intrinsic scene properties) and reorganization (grouping an image into meaningful regions), by jointly optimizing over a mixture of shapes, a mixture of illuminations, and the corresponding embedding of each mixture component in the image.

For our technique to work, our shape and light mixtures must respect the structure of the image. We therefore embed our mixtures in the normalized Laplacian of the image, building on normalized cuts [72], as well as Laplacian embeddings [12] and spectral graph theory [23]. This is motivated by the observation that variation in shape and illumination tends to produce gradients and contours in the image, and so our mixtures of shapes and illuminations should be embedded in a space that respects such image variation.

Using shading cues to infer shape, as we are attempting, is understood to work poorly for recovering low-frequency (coarse) shape information [16]. Thankfully, depth data from sensors such as the Kinect [32] is becoming increasing commonplace, and is complementary to shading: binocular disparity (the principle by which the Kinect computes depth) is accurate at coarse scales and inaccurate at fine scales. We will therefore assume the input to our model is an RGB-D image, where "D" is the depth map produced by a sensor such as the Kinect. This makes our problem easier, but in no way trivial — depth maps from sensors

such as the Kinect are noisy and incomplete for many reasons: occlusion of the structured light, dark objects, sensor noise, alignment errors, quantization, and the inherent physical limitations of binocular disparity. Attempts to use raw depth maps from the Kinect for photometric applications therefore often fail badly. See Figures 3.5-3.10 for demonstrations of how noisy these depth maps are compared to the depth maps that our model produces.

In Figures 3.5-3.7 we show the output of our model on an RGB-D image from the NYU Depth Dataset [73]. Our model's depth map is a clear improvement over the raw sensor depth map (missing regions have been filled in, noise has been removed, detail has been added), our output shading and reflectance images look better than those of the best "intrinsic image" algorithms, our shape mixture has separated the bed in the foreground from the walls in the background, and our recovered mixture of illuminations captures the complicated illumination in the scene produced by the lamp. Even "mistakes" produced by our model are compelling: our model has attempted to reconstruct the shape of the contents of the photos on the wall, and has modeled these contents with a different illumination environment than the rest of the scene, similarly to how a human might perceive an image within an image. See the supplementary material for dozens of additional examples of our output.

Some past work has addressed similar problems to our own. Forsyth [29] used a spatially-varying model of illumination to address complicated illumination and interreflection, but did not address reflectance or scene-like shape occlusion. Yu *et al.* [79] and Karsch *et al.* [46] have attempted to recover the reflectance and illumination of a scene, but assume known geometry and multiple images, or a user annotation of geometry and illumination, respectively. Hoeim *et al.* [38] and Saxena *et al.* [70] present algorithms for determining the "spatial layout" of a scene, but these shape estimates are coarse, and these models do not recover illumination, reflectance, or shading. Lee *et al.* [52] produces shading and reflectance images given RGB-D data, but requires a video and a fused depth map, and does not produce an illumination model or a refined shape.

This chapter is as follows: in Section 2.1 we review SIRFS, in Section 3.1 we formalize Scene-SIRFS, and in Section 3.2 we introduce the embedding used by our shape and illumination mixtures. In Sections 3.3 and 3.4 we present our priors on shape and illumination (our shape prior incorporates the input depth map from the Kinect), and in Section 3.5 we show how we optimize the resulting inference problem. In Sections 3.6.1 and 3.6.2 we present experiments on pseudo-synthetic and real RGB-D data.

3.1 Problem Formulation

The problem formulation of Scene-SIRFS is:

$$\begin{aligned} \underset{R,\boldsymbol{Z},\boldsymbol{\psi},\boldsymbol{L},\boldsymbol{\omega}}{\text{minimize}} \quad & g(R) + \sum_{n=1}^{|\boldsymbol{Z}|} f'(Z^n, U^n) + h'\left(\sum_{m=1}^{|\boldsymbol{L}|} V^m L^m\right) \\ \text{subject to} \quad & I = R + S'(\boldsymbol{Z}, \boldsymbol{U}, \boldsymbol{L}, \boldsymbol{V}) \\ & U^n = \frac{\exp(B\psi^n)}{\sum_{n'} \exp(B\psi^{n'})}, \forall_n \qquad V^m = \frac{\exp(B\omega^m)}{\sum_{m'} \exp(B\omega^{m'})}, \forall_m \end{aligned} \tag{3.1}$$

Where $\boldsymbol{Z} = \{Z^n\}$, $\boldsymbol{U} = \{U^n\}$, $\boldsymbol{L} = \{L^m\}$, and $\boldsymbol{V} = \{V^m\}$. This is similar to Equation 2.2, except that we have sets of shapes and lights instead of a single shape and light, and we have introduced $\boldsymbol{U}$ and $\boldsymbol{V}$, two sets of "images" that define distributions over shapes and lights, respectively. We can think of $\boldsymbol{U}$ as a "visibility" map or a soft relaxation of a $2.5D$ shape representation: if $U^n_{i,j} = 1$, then $Z^n_{i,j}$ is visible at pixel (i, j). Similarly, $\boldsymbol{V}$ is the "ownership" of each illumination in $\boldsymbol{L}$, such that if $V^m_{i,j} = 1$ then pixel (i, j) is entirely illuminated by L^m. Our prior on shape is now a sum of priors over individual depth maps, where each Z^n in $\boldsymbol{Z}$ is regularized independently (see Section 3.3). In contrast, our prior on illumination is over the expected illumination of the entire scene, the per-pixel weighted combination of each illumination (see Section 3.4). Our shape and light mixture probabilities $\boldsymbol{U}$ and $\boldsymbol{V}$ are "images" (where each image corresponds to one mixture component) parametrized by the matrices $\boldsymbol{\psi}$ and $\boldsymbol{\omega}$, respectively, where each column (ψ^n or ω^m) is a 17-dimensional vector parametrizing the "ownership" of that shape or light mixture in the scene. The probabilities $\boldsymbol{U}$ and $\boldsymbol{V}$ are the product of each weight matrix with B (the eigenvectors of the normalized Laplacian of the RGB image, explained in later) passed through a softmax function[1]. We use 8 shapes and illuminations in our mixtures for all experiments ($|\boldsymbol{L}| = |\boldsymbol{Z}| = 8$) though this is arbitrary. See Figures 3.5-3.7 for a visualization of these mixtures.

For the purpose of optimization, we need to define the normal field of this mixture of shapes $N'(\boldsymbol{Z}, \boldsymbol{U})$. We cannot use the surface normals of the expected depth map $N(\sum Z^n U^n)$ as this cannot model depth-discontinuities. We also cannot use the expected surface normals of the mixture of shapes $\sum U^n N(Z^n)$ as this normal field may have vectors of non-unit length. We will therefore linearize each Z^n into a set of partial derivatives in x and y, take the expectation of those with respect to $\boldsymbol{U}$, and then construct a normal field from those expected partial derivatives. This gives us a proper normal field where each Z^n's influence

[1] in a slight abuse of notation, $\boldsymbol{U}$ and $\boldsymbol{V}$ are simultaneously treated as sets of images and as matrices whose columns are vectorized images.

at pixel (i, j) is proportional to $U^n_{i,j}$. Formally:

$$N'(\boldsymbol{Z}, \boldsymbol{U}) = \left\{ \frac{D^x}{D^m}, \frac{D^y}{D^m}, \frac{1}{D^m} \right\}$$
$$D^x = \sum_{n=1}^{|\boldsymbol{Z}|} U^n (Z^n * h^x), \quad D^y = \sum_{n=1}^{|\boldsymbol{Z}|} U^n (Z^n * h^y)$$
$$D^m = \sqrt{1 + (D^x)^2 + (D^y)^2}$$
$$h^x = \frac{1}{8} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad h^y = \frac{1}{8} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{3.2}$$

Let $S'(\cdot)$ be our rendering engine for our mixtures, which computes the normal field of the mixture of shapes and renders it such that the spherical harmonic illumination at pixel (i, j) is a linear combination of all L^m, weighted by $V^m_{i,j}$:

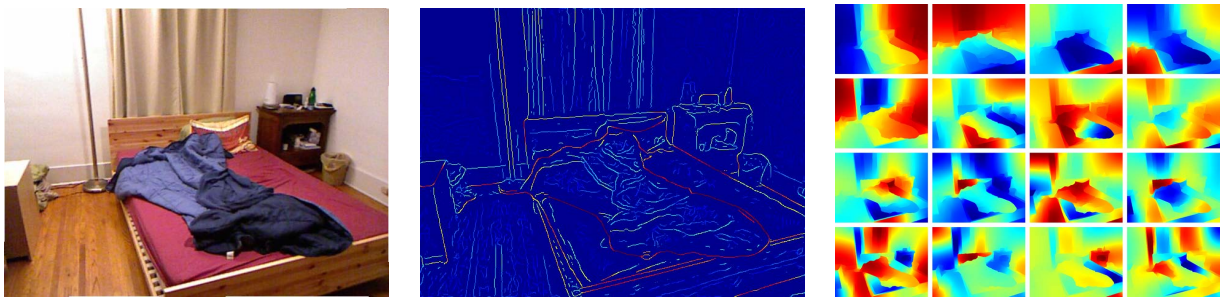
$$S'(\boldsymbol{Z}, \boldsymbol{U}, \boldsymbol{L}, \boldsymbol{V}) = S\left(N'(\boldsymbol{Z}, \boldsymbol{U}), \sum_{m=1}^{|\boldsymbol{L}|} V^m L^m \right) \tag{3.3}$$

Where $S(\cdot)$ is the rendering engine in SIRFS.

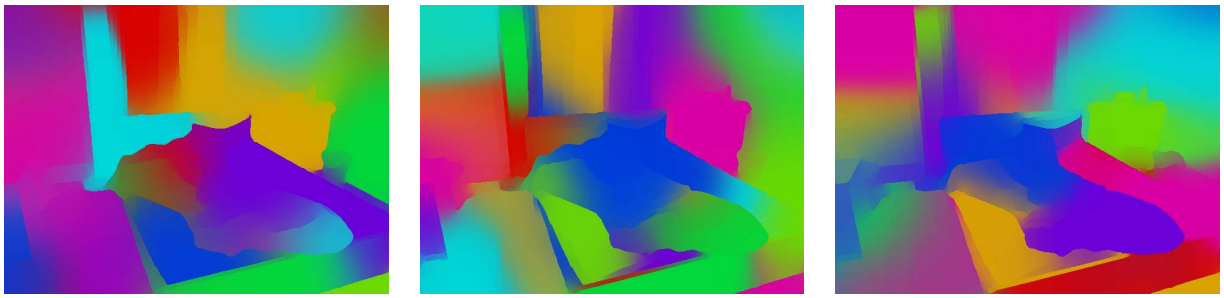
Though the spatially varying illumination parametrized by $\{\boldsymbol{L}, \boldsymbol{V}\}$ is capable of explaining away shadows, specularities, and interreflections, no attempt has been made to ensure that the illumination is globally consistent. Though this may seem unsettling, the human visual system has similar properties: people tend not to notice inconsistent shadows or impossible illumination [62].

3.2 Mixture Embedding

Using a mixture of shapes and illuminations is necessary to model depth discontinuities and spatially varying illumination, both of which tend to produce variation in the image in the form of contours, intensity variation, texture gradients, etc. It therefore follows that we should embed the shape and light mixtures in some space where the "ownership" of each mixture adheres to the segmentation of the scene. This simplifies inference, as we restrict our attention to only mixtures of the shapes and lights that are supported by evidence in the image. This is similar to the motivation for the use of superpixels as a substrate for inference in CRF-type models, though our experience suggests that superpixels are a poor embedding for this task, as they are too "hard". We will instead embed each mixture component in a more "soft" embedding: the eigenvectors of the normalized Laplacian of a graph corresponding to the input RGB image [72]. We construct our embedding as follows: given an image, first we compute the multiscale Pb of that image [4, 55]. We then form an affinity matrix from mPb using the intervening contour cue [53], and compute the 17



(a) Input Image (b) Multiscale Pb (c) Eigenvector basis



(d) A random mixture (e) Another random mixture (f) Another random mixture

Figure 3.1: A visualization of the embedding used in our shape and light mixtures. In 3.1(a), we have an input image. In 3.1(b) we have the output of multiscale Pb on the input image, and in 3.1(c) we have the 16 smallest eigenvectors (ignoring the eigenvector that is all 1's) of mPb using the intervening contour cue [4]. Each shape's and light's "ownership" of the image is parametrized by a 17-dimensional vector, which is projected onto the eigenvector basis and passed through a softmax function to yield the probability of each pixel belonging to each mixture component. 3.1(d), 3.1(e), and 3.1(f) are visualizations of three random mixtures with 8 components (such as $\boldsymbol{U}$ or $\boldsymbol{V}$) where the weight vectors ($\boldsymbol{\psi}$ or $\boldsymbol{\omega}$) are generated randomly (sampled from a Gaussian).

eigenvectors $\{\mathbf{u}_i\}$ corresponding to the smallest eigenvalues (the first eigenvector is all 1's). For eigenvectors 2 through 17, we subtract the mean from each $\mathbf{u}_i$ and divide each by its standard deviation, and then concatenate these normalized eigenvectors into a matrix B, with one column per-pixel. B is our embedding space, in that each mixture component is defined by a 17-dimensional vector, whose inner product with B defines how dominant that mixture component is at every pixel in the input image. A similar embedding is used in [56], for the purpose of combining recognition and segmentation. See Figure 3.1 for a visualization of this embedding.

It may seem unusual that we construct our embedding using only RGB information, instead of using the complete RGB-D image. We do this because the depth images are often mis-aligned and noisy enough that it is challenging to construct a single accurate contour signal from both sources of information. Using only the image to create an embedding

circumvents the noise in the depth map and forces the reconstructed shape to be aligned with the image.

Our prior on reflectance $g(\cdot)$ is exactly the same as in SIRFS. In Sections 3.3 and 3.4 we will define $f'(\cdot)$ and $h'(\cdot)$, our priors on our shape and illumination mixtures, respectively.

3.3 Shape Priors and Kinect Images

Our prior on shape is a modification of that of SIRFS. We use a linear combination of the smoothness and isotropy terms $f_k(Z)$ and $f_i(Z)$ from Sections 2.3.2 and 2.3.3 and we overwrite the $f_o(Z, U)$ term from Section 2.3.5 to incorporate knowledge from the raw sensor depth map produced by the Kinect $\hat{Z}$:

$$f'(Z, U) = \lambda_k f_k(Z) + \lambda_f f_i(Z) + \lambda_o f_o(Z, U) \tag{3.4}$$

Where $f_k(Z)$ minimizes the local variation of mean curvature of Z, encouraging Z to be smooth, and $f_i(Z)$ minimizes the slant of Z, encouraging Z to be fronto-parallel. We introduce $f_o(Z, U)$, which encourages Z to be similar to the raw sensor depth map if Z is thought to be "visible" according to U. Crucially, we apply this prior to each individual depth map in our mixture rather than to some average depth map. This encourages the scene's constituent depth maps to be smooth while allowing the expected depth map implied by the mixture to vary abruptly, thereby allowing us to model depth discontinuities and occlusion.

We use version 2 of the NYU Depth Dataset [73], which consists of RGB images and aligned Kinect depth maps. Because Kinect depth maps often have gaps, the dataset also provides inpainted depth maps. We will use the raw depth maps rather than the inpainted ones, as our algorithm will implicitly denoise and inpaint depth during inference. In addition to gaps, Kinect depth maps have different kinds of noise. First: the depth and RGB images are often not well-aligned — not enough to matter for most recognition tasks, but enough to affect photometric or reconstruction tasks. Second: the disparity recovered by the Kinect is often noisy, presumably due to sensor noise or errors in the Kinect's stereo algorithm. Third: the disparity is quantized, which leads to step-like artifacts in depth.

We must construct a loss function to encourage our recovered depth Z to resemble the raw sensor depth $\hat{Z}$. First, let us approximate the upper bound of the error introduced by quantizing the disparity corresponding to $\hat{Z}$:

$$Z^{err}_{i,j} = (1.4233 \times 10^{-5})\hat{Z}^2_{i,j} + 2 \tag{3.5}$$

where $\hat{Z}$ and Z^{err} are in centimeters. The first term is derived from the baseline of the Kinect, and the second term is additional ad-hoc slack. We assume that if the difference between $Z_{i,j}$ and $\hat{Z}_{i,j}$ at pixel (i, j) is less than $Z^{err}_{i,j}$, then that difference is due to quantization and therefore should be ignored. Errors greater than $Z^{err}_{i,j}$ will be robustly penalized, as they probably are due to sensor noise or alignment errors. Our loss function is:

$$f_o(Z, U) = \sum_{i,j} U_{i,j} \max\left(0, \left|Z_{i,j} - \hat{Z}_{i,j}\right| - Z^{err}_{i,j}\right)^{\alpha_o} \tag{3.6}$$

Minimizing this is equivalent to assuming noise is uniformly distributed over a region of size $2Z_i^{err}$ and is hyper-Laplacian outside of that range. The loss is proportional to $U_{i,j}$, which means that $Z_{i,j}$ need only resemble $\hat{Z}_{i,j}$ if our model believes that this depth map is in the foreground at pixel (i, j). α_o controls the shape of the tail of the distribution, and is tuned with cross-validation on the training set (along with λ_k, λ_f, and λ_o), which sets $\alpha_o = 0.7$.

3.4 Illumination Priors

Our prior on illumination is a simple extension of the illumination prior of Section 2.4 to a mixture model, in which we regularize the expectation of a set of illuminations instead of a single illumination. Given $\boldsymbol{L}$ (our set of spherical harmonic illuminations) and $\boldsymbol{V}$ (our set of "images" that define a per-pixel distribution over our illuminations), we can compute the expectation of this model at each pixel of the image:

$$\bar{L}_{i,j} = \sum_{m=1}^{|\boldsymbol{L}|} V_{i,j}^m L^m \tag{3.7}$$

Where $\bar{L}_{i,j}$ is a 27-dimensional vector describing the effective illumination at pixel (i, j) in the image. Our prior on illumination is the negative log-likelihood of a multivariate normal distribution, applied to each 27-dimensional "pixel" in $\bar{L}$:

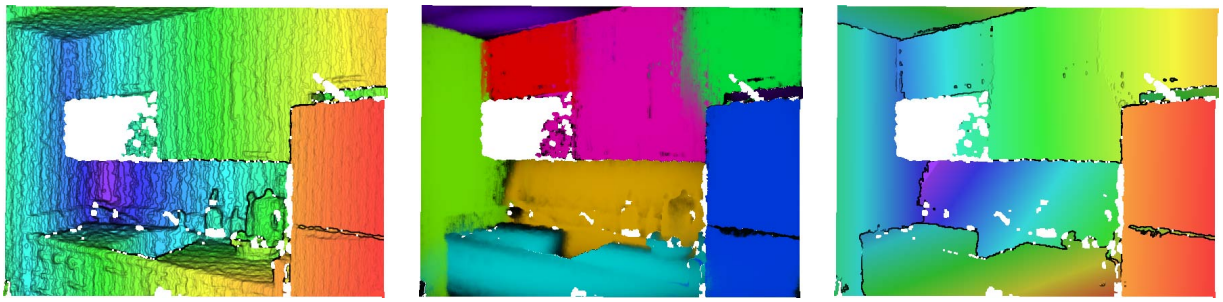
$$h'(\bar{L}) = \lambda_L \sum_{i,j} (\bar{L}_{i,j} - \boldsymbol{\mu}_L)^{\mathrm{T}} \Sigma_L^{-1} (\bar{L}_{i,j} - \boldsymbol{\mu}_L) \tag{3.8}$$

Where $\boldsymbol{\mu}_L$ and Σ_L are the parameters of the Gaussian we learn on the training set, and λ_L is the multiplier on this prior (learned through cross-validation on the training set).

3.5 Initialization & Optimization

Optimization of our model is similar to that of Section 2.6. We absorb the $I = R + S(\cdot)$ constraint in Equation 3.1 by rewriting $g(R)$ as $g(I - S(\cdot))$, thereby eliminating R as a free parameter. In optimization we internally represent each depth map Z^n as a pyramid, and whiten each illumination L^m according to $\{\boldsymbol{\mu}_L, \Sigma_L\}$. We vectorize those our pyramid-depths, whitened illuminations, and mixture weights $\{\boldsymbol{\psi}, \boldsymbol{\omega}\}$ into one state vector, and then minimize the loss in Equation 3.1 using L-BFGS.

This optimization problem is non-convex, and so it is sensitive to initialization. Because the scenes in the NYU dataset are mostly composed of planar surfaces, we will initialize each depth map Z^i in $\boldsymbol{Z}$ to a plane such that the scene is well-described by the set of planes. To do this, we fit a mixture of Gaussians to the (x, y, z) coordinates of each pixel in $\hat{Z}$ (in image coordinates) using EM with 50 random restarts. Once EM converges we have n multivariate Gaussians, each parametrized by a mean μ and a covariance matrix Σ. If a Gaussian does



(a) Kinect Depth Map (b) Mixture of Gaussians (c) Plane-Fit Depth Map

Figure 3.2: We initialize the depth maps in our shape mixture by fitting a mixture of Gaussians to the (x, y, z) coordinates of depth-map pixels, and then fitting a plane to each Gaussian. 3.2(a) shows the raw depth map, 3.2(b) shows the posterior probability of each pixel under each mixture component, and 3.2(c) shows the fitted planes composed into one depth map according to hard assignments under the mixture of Gaussians.

indeed describe a roughly-planar surface, then Σ will be elongated in two directions, and narrow in the third. This means that the Gaussian is well described by the plane satisfying $v^{\mathrm{T}}([x, y, z] - \mu) = 0$, where v is the eigenvector corresponding to the smallest eigenvalue of Σ. We initialize each surface in our mixture to its corresponding plane in our mixture of Gaussians, by solving for z at every pixel. See Figure 3.2 for a visualization.

This plane-fitting sometimes produces poor results on our synthetic dataset, because our synthetic scenes contain mostly fronto-parallel objects stacked on top of each other. Therefore, in our synthetic experiments we initialize the depth maps by doing K-means (with 50 random restarts) on just the z values in the scene, and then initializing each depth map to be a centroid, thereby constraining the initial depth-planes to be fronto-parallel.

3.6 Experiments

3.6.1 Pseudo-synthetic Dataset

The primary goal of this chapter is to produce a model that works well on actual Kinect images. However, it is extremely difficult to produce ground-truth shape, reflectance, shading, and illumination models for real-world natural scenes. Thankfully, using the MIT-Berkeley Intrinsic Images dataset we can compose pseudo-synthetic scenes that emulate natural scenes. We will do this by layering the objects in the dataset into a scene, and then rendering the resulting shapes using a randomly sampled spatially-varying illumination. For each scene we also produce a noisy Kinect-like depth map for use as input to our model. With a dataset of "scene"-like images which contain occlusion and spatially varying illumination, we can quantitatively evaluate how our model might perform on actual Kinect imagery. This dataset also

allows us to tune hyperparameters (on the training set) and compare our model to others (on the test set).

The creation of our dataset is as follows: The 20 objects in the MIT dataset are downsampled by a factor of two, such that our resulting scenes can be reasonably sized. We split the objects into training and test sets, using the same split as Section 2.7. We then synthesize 10 training and 10 test scenes, where training scenes are composed of training objects, and test scenes are composed of test objects. To generate a scene, we iteratively layer objects on top of each other, taking care to place each object in the largest part of the scene that is still empty. For each scene we generate a layered depth map, normal map, and reflectance image. Given these, we then synthesize a natural illumination to generate a shading image. All of our scenes are 256×256 pixels.

Natural scenes have complicated, spatially-varying illumination due to attenuation, shadowing, interreflection, etc. Therefore, to make this dataset a somewhat realistic surrogate for real images of natural scenes, we cannot simply use one global model of illumination, as was done in Section 2.7.4. We will instead synthesize our own spatially-varying illumination as a mixture of many attenuated point-light sources. For each scene, we generate 50 lights, each of whose position is generated in a uniform region twice the size of the volume enveloped by the objects in the scene, and whose color c_i is randomly sampled from the average RGB value of an image in the sIBL Archive[2]. For each light, we compute the distance d_i of each depth-map pixel to the light source and compute an attenuation $a_i = 1/(1+d_i^2/40000)$, where 40000 (which controls the amount of attenuation) is chosen manually such that the resulting scenes look reasonable. For each pixel, we compute the unit vector ℓ_i from that pixel's location in the scene to the light source's position. We then render the scene according to attenuated Lambertian reflectance $(c_i \times \max(0, a_i(n_i \cdot \ell_i)))$ for each light source, and take the sum of all of these renderings (after dividing by the max intensity) as the complete shading image. The final image, which is this shading image multiplied by the previously-generated reflectance image, will be the RGB input to our system.

In parallel with rendering these images, we render a "light probe" surface that has the same depths as the scene, but whose normal field is chosen to uniformly sample the space of orientations. We will evaluate the fidelity of our recovered illumination by rendering this "light probe" surface according to our recovered illumination, and comparing it to the ground-truth produced during this synthesis.

In Section 3.3 we described a probabilistic model of the noise present in Kinect depth maps, for use during inference. This noise model is not sufficient for our experiments: we must be able to synthetically generate noisy depth maps that are similar to Kinect depth maps, for the purposes of tuning our model parameters and empirically evaluating the accuracy of our model on pseudo-synthetic data. We therefore present a generative model to construct a noisy depth map: Given a "true" depth-map Z^* in centimeters, we first construct a disparity map. We replace each pixel in the disparity map with the bilinearly interpolated value of a location near that pixel where the shift is drawn from a normal distribution

[2] `http://www.hdrlabs.com/sibl/archive.html`

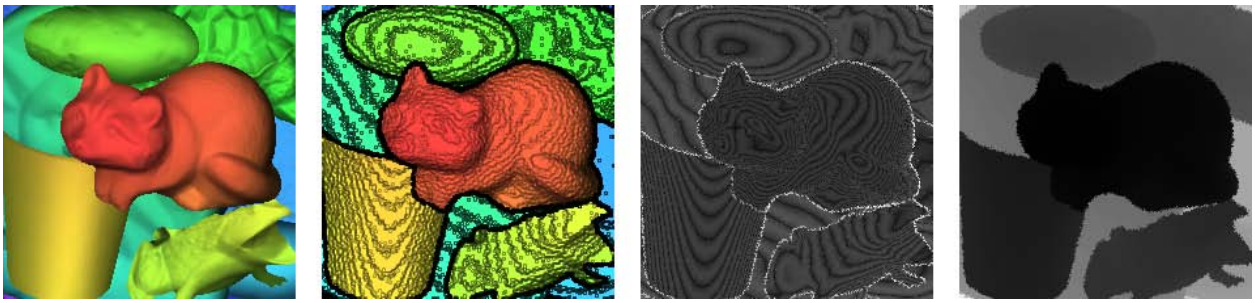


Figure 3.3: A visualization of how we introduce noise to Kinect images. In the first column we have a ground-truth depth-map, and in the second we have our corrupted version of it that we will use as a proxy for Kinect data. The third column shows the difference between the two, where we see the stripe-like noise introduced by quantization, as well as the noise near the boundaries of the objects introduced by shuffling and mis-aligning the image. The fourth column is a visualization of our error model, where we see error increases with depth, in accordance with our understanding of binocular stereo.

$\sigma = 1/2$, then we add IID Gaussian noise to each pixel of the disparity map ($\sigma = 1/6$), and then we translate all pixels in the disparity map by a shift drawn from a normal distribution ($\sigma = 1/4$). The first shuffling and injection of noise is intended to simulate sensor noise in the Kinect, and the second image-wide shuffling is intended to simulate a slightly inaccurate alignment between the image and the depth map, which we often see in even well-aligned Kinect data. We quantize the shuffled disparity by rounding it to the nearest integer, and then convert the disparity measurements back into depth measurements. Formally, our procedure for corrupting a ground-truth depth map is:

$$\hat{Z} \leftarrow \frac{35130}{\lfloor 35130/(\text{shuffle}(Z^*)) + \mathcal{N}(0, (1/6)^2) + 0.5 \rfloor} \tag{3.9}$$

The constant 35130 is derived from the baseline of the Kinect sensors. See Figure 3.3 for a visualization of these synthetic noisy depth maps. Though our model for generating noise is different from our model for inference in the face of noise, manual investigation of the data suggests that the models largely agree — the marginal distribution of the noisy depth maps generated from Equation 3.9 appears to match the prior posed earlier, in terms of the width of the uniform distribution and the shape of the hyper-laplacian tail. Perhaps more importantly, our synthetically noisy depth maps look similar to the real depth maps in the NYU Depth Dataset [73].

Table 3.1 compares our model's performance to other intrinsic images techniques and to ablations of our model. The shading and reflectance images produced by our model beat or match the best intrinsic image algorithms. The surface normals produced by our model have half of the error of the input, though for absolute depth error we do not improve. This is consistent with the limits of shading, as shading directly informs the surface normal, but only implicitly informs absolute depth. Our performance is similar to that of SIRFS in

terms of reflectance, but much better in all other respects. A naive extension of SIRFS to scenes (in which we use normalized cuts to segment each image into 16 segments and run SIRFS on each segment in isolation) performs similarly to basic SIRFS. The source of our advantage over SIRFS is shown through our ablations — removing either the shape or the illumination mixture components hurts performance on every error metric, and removing the Kinect depth map hurts performance on the depth and normal error metrics, though not the shading or reflectance metrics. The degenerate case of our model which only denoises the depth map and ignores the RGB image performs surprisingly well in terms of error relative to the ground-truth shape and normal field. However, we believe this mostly reflects a bias in our error metrics towards overly smooth shapes, which the shape-denoising ablation produces (see Figure 3.10).

Figure 3.4: A test-set scene from our pseudo-synthetic scene dataset. In 3.4(a) we have the input to our model: an RGB image and a noisy Kinect-like depth map. In 3.4(b) we have the depth map, surface normals, reflectance, shading, and spatially-varying illumination that our model produces, and the corresponding ground-truth scene properties on the bottom. In 3.4(c) and 3.4(d) we show the shading and reflectance images produced by the best-performing intrinsic image algorithms. See the supplementary material for additional similar figures.

Algorithm	Z-MAE	N-MAE	S-MSE	R-MSE	RS-MSE	L-MSE	Avg.
(1) Color Retinex [35, 40]	-	-	0.0230	0.0364	0.0354	-	-
(2) Tappen *et al.* 2005 [74]	-	-	0.0281	0.0337	0.0387	-	-
(3) Gehler *et al.* 2011 [33]	-	-	0.0181	0.0224	**0.0216**	-	-
(4) Kinect Only	**5.09**	0.5799	-	-	-	-	-
(5) SIRFS	114.82	0.6841	0.0181	0.0202	0.0289	0.0241	0.1647
(6) SIRFS + Segmentation	57.43	0.7600	0.0176	0.0200	0.0296	0.0210	0.1458
(A) Scene-SIRFS (Complete)	10.91	**0.2618**	**0.0101**	**0.0184**	0.0227	**0.0166**	**0.0764**
(B) Scene-SIRFS ($\lambda_o = 0$)	122.67	0.6454	0.0134	0.0203	0.0256	0.0199	0.1491
(C) Scene-SIRFS (No Initialization)	11.06	0.3000	0.0113	0.0233	0.0263	0.0176	0.0860
(D) Scene-SIRFS ($\|\boldsymbol{Z}\| = 1$)	22.72	0.5123	0.0179	0.0284	0.0348	0.0237	0.1302
(E) Scene-SIRFS ($\|\boldsymbol{L}\| = 1$)	11.64	0.2754	0.0163	0.0313	0.0269	0.0211	0.0988
(F) Scene-SIRFS ($\|\boldsymbol{Z}\| = \|\boldsymbol{L}\| = 1$)	24.59	0.5285	0.0292	0.0587	0.0523	0.0213	0.1708
(G) Scene-SIRFS (Z only)	9.82	0.2877	-	-	-	-	-
(H) Scene-SIRFS (Z only, $\|\boldsymbol{Z}\| = 1$)	24.69	0.5552	-	-	-	-	-

Table 3.1: Our results on the test set of our pseudo-synthetic dataset. Shown are the geometric means of six error metrics (detailed in the supplementary material) across the test set, and an "average" error (the geometric mean of the other error metrics). Z-MAE measures shape errors, N-MAE measures surface-normal errors, S-MSE, R-MSE, and RS-MSE measure shading and reflectance errors, and L-MSE measures illumination errors. If an algorithm does not produce a certain scene property, its error is left blank. (1)-(3) are intrinsic image algorithms, which produce shading and reflectance images from an RGB image, where (3) is the current state-of-the-art. (4) evaluates the error of the noisy Kinect-like depth maps we use as input. (5) is the SIRFS model that we build upon, and is equivalent to our model without any mixture models or a Kinect depth map. (6) is SIRFS run in isolation on the segments produced by normalized cuts. In addition to our complete model (A), we present several ablations. (B) has no Kinect information, (C) has no initialization, and (D)-(F) omit one or both of the shape or light mixtures. (G) is a shape-denoising algorithms in which we omit the RGB image and just optimize over shape with respect to our prior on shapes, and (H) is (G) with a single depth map, instead of a mixture model.

3.6.2 Kinect Data

To qualitatively evaluate our model, we sampled several images from version 2 of the NYU Depth Dataset [73], and ran them through our model (all using the same hyperparameter setting as in our pseudo-synthetic experiment). The output of our model can be seen in Figures 3.5-3.7. We compare against two intrinsic image algorithms: Retinex [35, 40] and Gehler *et al.* [33].

Our shading and reflectance images generally look much better than those produced by the intrinsic image algorithms, and our recovered depth and surface normals look much better than the input Kinect image. Our spatially varying illumination captures shadowing and interreflections, and looks reasonable. The primary cause of errors in our model appears to be over-smoothing of the depth map, which we believe is because the error metrics with which we cross-validate our model tend to favor conservative parameter settings, and because MAP estimation for tasks such as ours tends to produce overly conservative output [54].

One way to evaluate the accuracy of our model is to use it in graphics applications. In Figures 3.8 and 3.9 we use our output to re-render the input image under different camera viewpoints and under different illumination conditions. Our renderings look significantly better than renderings produced with the inpainted Kinect depth map provided by the NYU dataset. Changing the viewpoint with the raw Kinect depths creates jagged artifacts at the edges of shapes, while our depth (which is both denoised and better-aligned to the image) looks smooth and natural at object boundaries. Relighting the raw Kinect depth produces terrible artifacts, as the surface normals of the raw depth are very inaccurate due to noise and quantization, while relighting our output looks reasonable, as the surface normals are cleaner and reflectance has been separated from shading. In Figure 3.10 we see that the depth maps our model produces are less noisy than the NYU depth maps, and more detailed than the output of the shape-denoising ablation of our model, demonstrating the importance of the complete model.

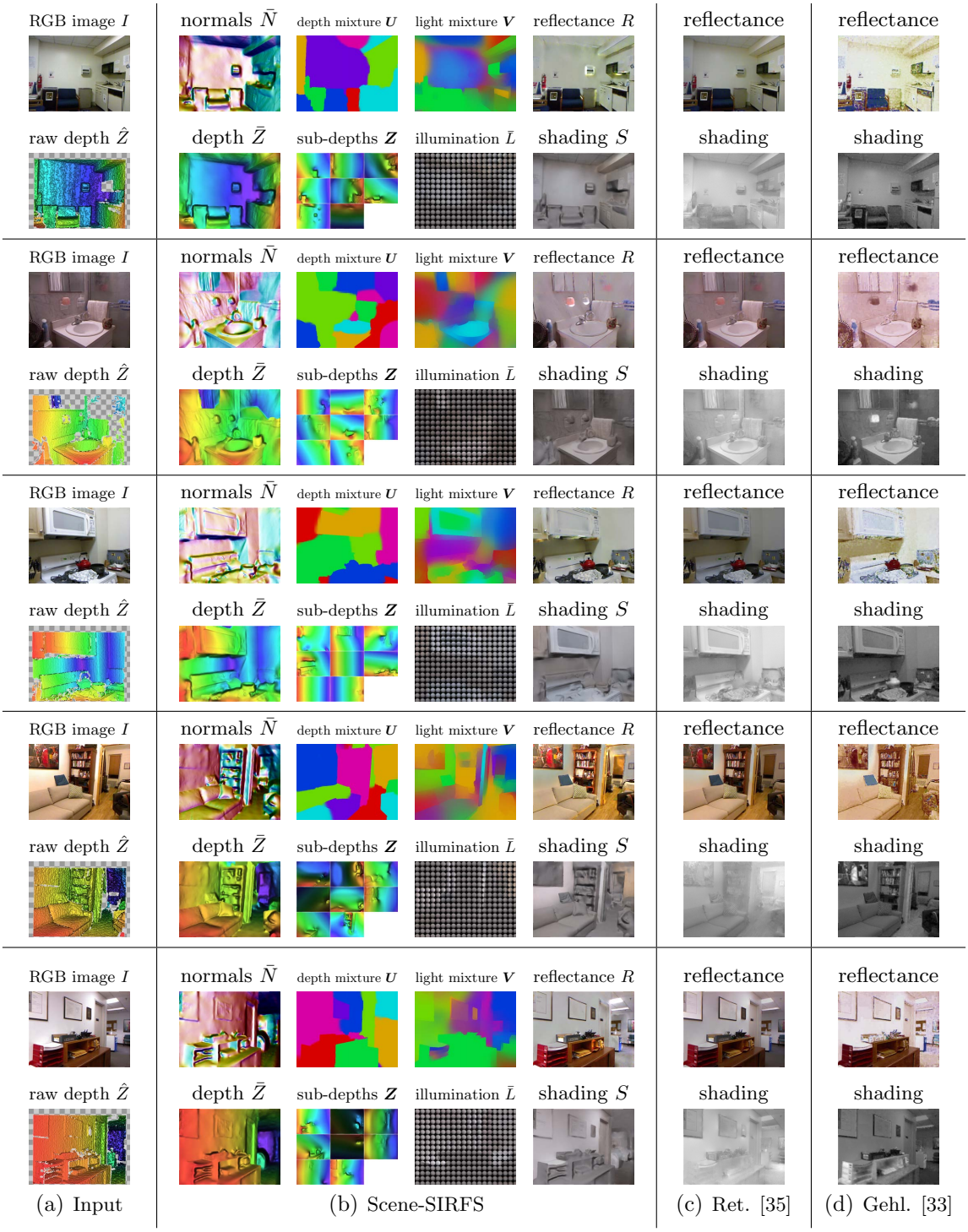


Figure 3.5: In 3.5(a) we have the input to our model: an RGB image and a Kinect depth map from the NYU Depth Dataset [73]. In 3.5(b) we have the output of our model. Mixtures are visualized with hue corresponding to component, and intensity corresponding to probability. Illumination is visualized by rendering a coarse grid of spheres.

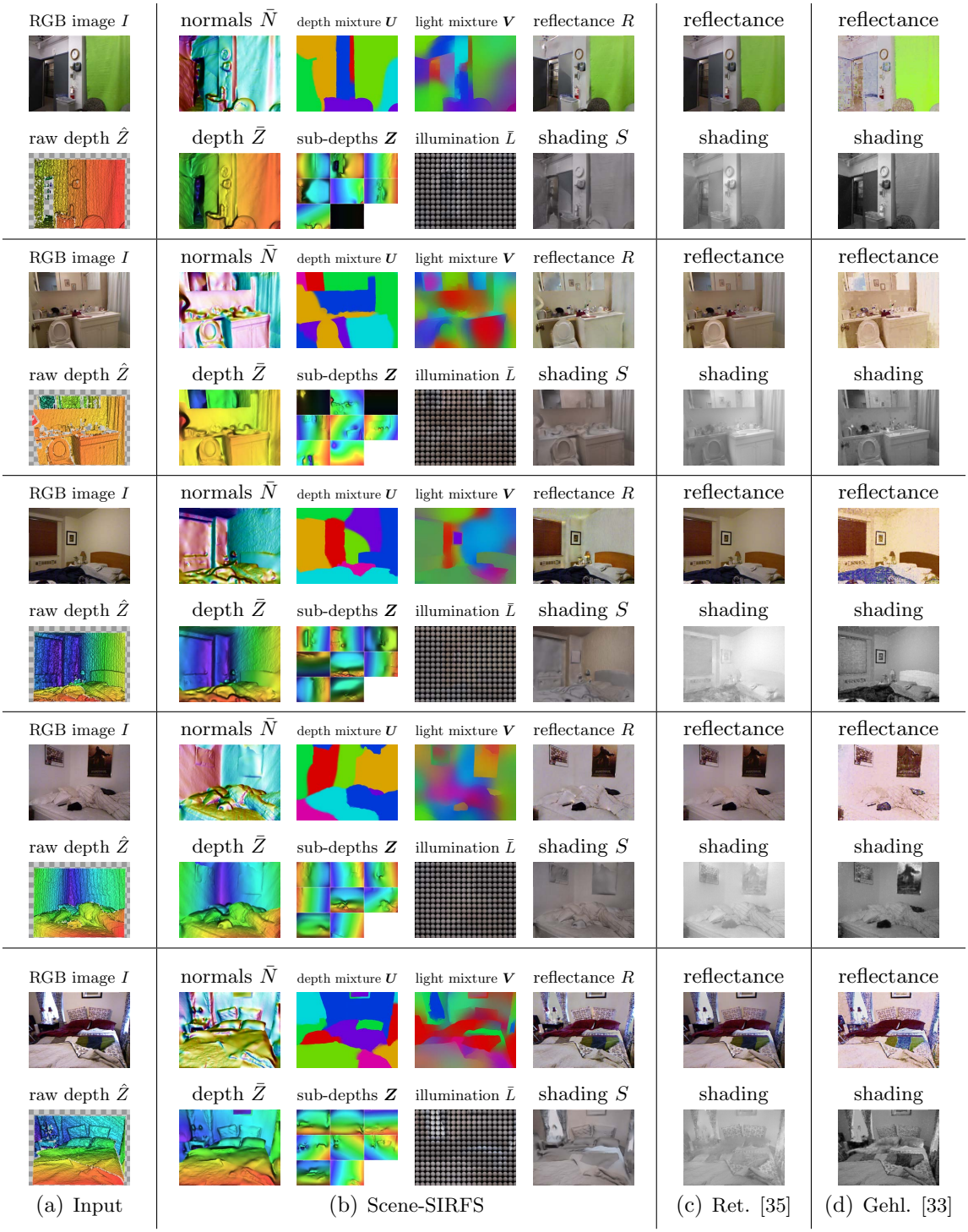


Figure 3.6: More results from the NYU Depth Dataset [73].

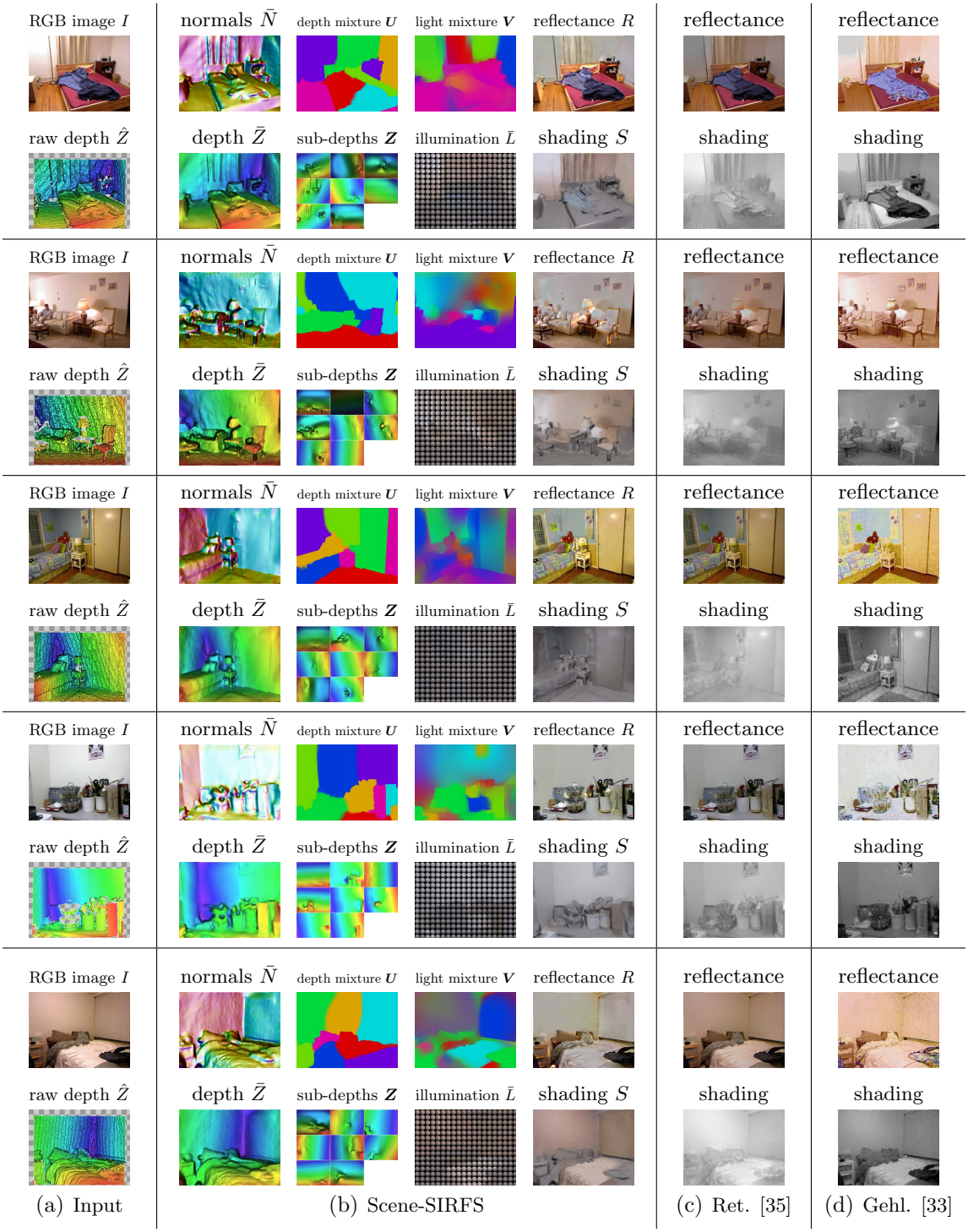
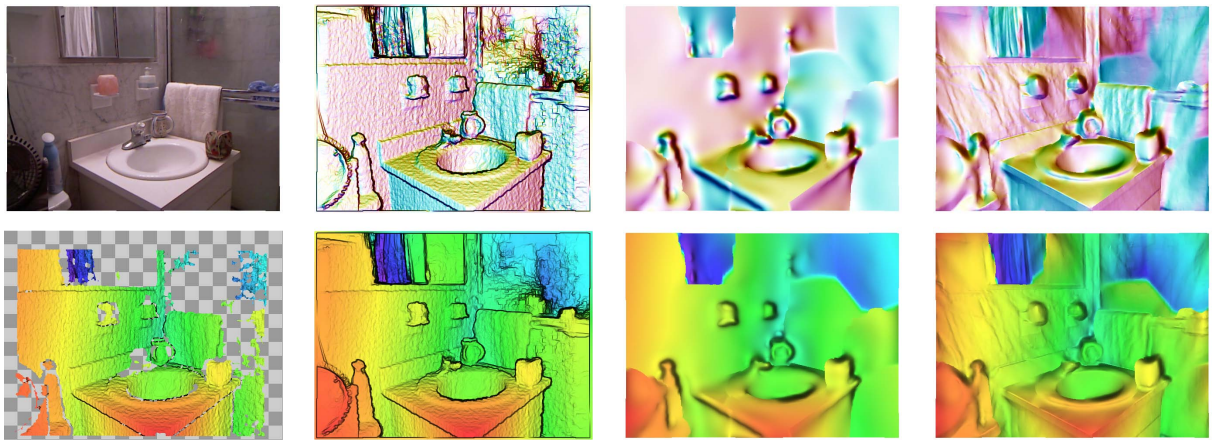


Figure 3.7: Even more results from the NYU Depth Dataset [73].

Figure 3.8: After a model is recovered, the camera can be moved and the input image (left) can be shown from a different viewpoint (right). Such a warping could be produced using just the smoothed Kinect depth maps provided in the NYU dataset (middle), but these images have jagged artifacts at surface and normal discontinuities. Both renderings, of course, contain artifacts in occluded regions.

Figure 3.9: After a model is recovered, the spherical harmonic illuminations can be replaced (here we use randomly generated illuminations) and the input image (left) can shown under a different illumination (right). The middle image is our attempt to produce similar re-lit images using only the inpainted depth maps in the NYU dataset, which look noticeably worse due to noise in the depth image and the fact that illumination and reflectance have not been decomposed.



(a) input (b) NYU depth (c) denoised depth (d) our depth

Figure 3.10: One output of our model is a denoised depth-map. In 3.10(a) we have the RGB-D input to our model, demonstrating how noisy and incomplete the raw Kinect depth map can be. 3.10(b) shows the inpainted normals and depth included in the NYU dataset [73], where holes have been inpainted but there is still a great deal of noise, and many fine-scale shape details are missing. 3.10(c) is from an ablation of our model in which we just denoise/inpaint the raw depth map ("model H" in our ablation study), and 3.10(d) is from our complete model. The NYU depth map is noisy at high frequencies and does not model depth discontinuities (hence the dark "slant" lines outlining each object), and our "denoising" model tends to oversmooth the scene, but our complete model has little noise while recovering much of the detail of the scene and correctly separating objects into different layers.

Chapter 4

Conclusion

We have presented SIRFS, a model which takes as input a single (masked) image of an object, and produces as output a reasonable estimate of the shape, surface normals, reflectance, shading, and illumination which produced that image. At the core of SIRFS is a series of priors on shape, reflectance, and illumination: surfaces tend to be isotropic and bend infrequency, reflectance images tend to be piecewise smooth and low-entropy, and illumination tends to be natural. Given these priors and our multiscale optimization technique, we can infer the most-likely explanation of a single image subject to our priors and the constraint that the image be explained. Our unified approach to this problem outperforms all previous solutions to its constituent problems of shape-from-shading and intrinsic image recovery on our challenging dataset, and produces reasonable results on arbitrary masked images of real-world objects in uncontrolled environments. This suggests that the shape-from-shading and intrinsic images problem formulations may be fundamentally limited, and attention should be refocused towards developing models that jointly reason about shape and illumination in addition to shading and reflectance.

We have also presented Scene-SIRFS, a variant of SIRFS that takes as input images of natural scenes rather than images of segmented objects. We have done this by generalizing SIRFS into a mixture model of shapes and illuminations, and by embedding those mixtures into a soft segmentation of an image. We additionally use the noisy depth maps in RGB-D data to improve low-frequency shape estimation. Scene-SIRFS addresses the primary shortcomings of SIRFS, in that it can model multiple, occluding objects, spatially-varying illumination, and can use the coarse shape estimate provided by the Kinect to overcome the problems associated with estimating shape from shading.

The output of our model can be used for graphics applications such relighting or reorienting the camera, and it is easy to imagine other applications such as inserting objects, modifying reflectances, or white balancing. Our model improves the initial depth map by removing noise, adding fine-scale shape detail, and aligning the depth to the RGB image, all of which presumably would be useful in any application involving RGB-D images. Perhaps most importantly, our model takes an important step towards solving one of the grand challenges in vision — inferring all intrinsic scene properties from a single image.

Bibliography

[1] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. Fast high-dimensional filtering using the permutohedral lattice. *Eurographics*, 2012.

[2] E.H. Adelson and A.P. Pentland. The perception of shading and reflectance. *Perception as Bayesian inference*, 1996.

[3] N.G. Alldrin, S.P. Mallick, and D.J. Kriegman. Resolving the generalized bas-relief ambiguity by entropy minimization. *CVPR*, 2007.

[4] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *TPAMI*, 2011.

[5] Jonathan T Barron and Jitendra Malik. High-frequency shape and albedo from shading using natural image statistics. *CVPR*, 2011.

[6] Jonathan T Barron and Jitendra Malik. Color constancy, intrinsic images, and shape estimation. *ECCV*, 2012.

[7] Jonathan T Barron and Jitendra Malik. Shape, albedo, and illumination from a single image of an unknown object. *CVPR*, 2012.

[8] Jonathan T Barron and Jitendra Malik. Intrinsic scene properties from a single rgb-d image. *CVPR*, 2013.

[9] H.G. Barrow and J.M. Tenenbaum. Recovering intrinsic scene characteristics from images. *Computer Vision Systems*, 1978.

[10] R. Basri, D. Jacobs, and I. Kemelmacher. Photometric stereo with general, unknown lighting. *IJCV*, 72(3):239–257, 2007.

[11] P.N. Belhumeur, D.J. Kriegman, and A.L. Yuille. The Bas-Relief Ambiguity. *IJCV*, 1999.

[12] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 2002.

[13] Matt Bell and William T. Freeman. Learning local evidence for shading and reflectance. *ICCV*, 2001.

[14] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Opensurfaces: A richly annotated catalog of surface appearance. *SIGGRAPH*, 2013.

[15] P.J. Besl and R.C. Jain. Segmentation through variable-order surface fitting. *TPAMI*, 1988.

[16] Andrew Blake, Andrew Zisserman, and Greg Knowles. Surface descriptions from stereo and shading. *IVC*, 1986.

[17] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. *SIGGRAPH*, 1999.

[18] Huseyin Boyaci, Katja Doerschner, Jacqueline L. Snyder, and Laurence T. Maloney. Surface color perception in three-dimensional scenes. *Visual Neuroscience*, 2006.

[19] Michael Brady and Alan Yuille. An extremum principle for shape from contour. *TPAMI*, 1983.

[20] David H. Brainard and William T. Freeman. Bayesian color constancy. *JOSA A*, 1997.

[21] Michael J. Brooks and Berthold K. P. Horn. *Shape from shading*. MIT Press, 1989.

[22] Jiawen Chen, Sylvain Paris, and Frédo Durand. Real-time edge-aware image processing with the bilateral grid. *SIGGRAPH*, 2007.

[23] Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[24] Ron O. Dror, Alan S. Willsky, and Edward H. Adelson. Statistical characterization of real-world illumination. *JOV*, 2004.

[25] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, and W.T. Freeman. Removing camera shake from a single photograph. *SIGGRAPH*, 2006.

[26] D.J. Field. Relations between the statistics of natural images and the response properties of cortical cells. *JOSA A*, 1987.

[27] G.D. Finlayson, S.D. Hordley, and P.M. Hubel. Color by correlation: a simple, unifying framework for color constancy. *TPAMI*, 2001.

[28] Graham D. Finlayson, Mark S. Drew, and Cheng Lu. Entropy minimization for shadow removal. *IJCV*, 2009.

[29] D. A. Forsyth. Variable-source shading analysis. *IJCV*, 2011.

[30] David Forsyth and Andrew Zisserman. Reflections on shading. *TPAMI*, 1991.

[31] David A. Forsyth. A novel algorithm for color constancy. *IJCV*, 1990.

[32] Barak Freedman, Alexander Shpunt, Meir Machline, and Yoel Arieli. Depth mapping using projected patterns. *US Patent*, 2009.

[33] Peter Gehler, Carsten Rother, Martin Kiefel, Lumin Zhang, and Bernhard Schoelkopf. Recovering intrinsic images with a global sparsity prior on reflectance. *NIPS*, 2011.

[34] Alan Gilchrist. *Seeing in Black and White*. Oxford University Press, 2006.

[35] Roger Grosse, Micah K. Johnson, Edward H. Adelson, and William T. Freeman. Ground-truth dataset and baseline evaluations for intrinsic image algorithms. *ICCV*, 2009.

[36] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge Univ Press, 2003.

[37] D. Hilbert and Cohn S. Vossen. *Geometry and the Imagination*. Chelsea Publishing Company, 1956.

[38] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Recovering surface layout from an image. *IJCV*, 2007.

[39] Berthold K. P. Horn. Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. Technical report, MIT, 1970.

[40] Berthold K. P. Horn. Determining lightness from an image. *Computer Graphics and Image Processing*, 1974.

[41] Berthold K. P. Horn. Obtaining shape from shading information. *The Psychology of Computer Vision*, 1975.

[42] Jinggang Huang, Ann B. Lee, and David Mumford. Statistics of range images. *CVPR*, 2000.

[43] Jinggang Huang and David Mumford. Statistics of natural images and models. *CVPR*, 1999.

[44] K. Ikeuchi and B.K.P. Horn. Numerical shape from shading and occluding boundaries. *Artificial Intelligence*, 1981.

[45] Micah K. Johnson and Edward H. Adelson. Shape estimation in natural illumination. *CVPR*, 2011.

[46] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. *SIGGRAPH Asia*, 2011.

[47] G.J. Klinker, S.A. Shafer, and T. Kanade. A physical approach to color image understanding. *IJCV*, 1990.

[48] Jan Koenderink, Andrea van Doorn, Chris Christou, and Joseph Lappin. Shape constancy in pictorial relief. *Perception*, 1996.

[49] J.J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 1984.

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

[51] Edwin H. Land and John J. McCann. Lightness and retinex theory. *JOSA*, 1971.

[52] Kyong Joon Lee, Qi Zhao, Xin Tong, Minmin Gong, Shahram Izadi, Sang Uk Lee, Ping Tan, and Stephen Lin. Estimation of intrinsic image sequences from image+depth video. *ECCV*, 2012.

[53] Thomas K. Leung and Jitendra Malik. Contour continuity in region based image segmentation. *ECCV*, 1998.

[54] A. Levin, Y. Weiss, F. Durand, and W.T. Freeman. Understanding and evaluating blind deconvolution algorithms. *CVPR*, 2009.

[55] Michael Maire, Pablo Arbelaez, Charless C. Fowlkes, and Jitendra Malik. Using contours to detect and localize junctions in natural images. *CVPR*, 2008.

[56] S. Maji, N.K. Vishnoi, and J. Malik. Biased normalized cuts. *CVPR*, 2011.

[57] Jitendra Malik. Interpreting line drawings of curved objects. *IJCV*, 1, 1987.

[58] Laurence T. Maloney and Brian A. Wandell. Color constancy: a method for recovering surface spectral reflectance. *JOSA A*, 1986.

[59] P. Mamassian, D. Kersten, and D. C. Knill. Categorical local-shape perception. *Perception*, 1996.

[60] Henry P. Moreton and Carlo H. Squin. Functional optimization for fair surface design. In *SIGGRAPH*, 1992.

[61] Ido Omer and Michael Werman. Color lines: Image specific color representation. *CVPR*, 2004.

[62] Y. Ostrovsky, P. Cavanagh, and P. Sinha. Perceiving illumination inconsistencies in scenes. *Perception*, 2005.

[63] Javier Portilla, Vasily Strela, Martin J. Wainwright, and Eero P. Simoncelli. Image denoising using scale mixtures of gaussians in the wavelet domain. *IEEE Trans. Image Process*, 2003.

[64] Jose C. Principe and Dongxin Xu. Learning from examples with quadratic mutual information. *Workshop on Neural Networks for Signal Processing*, 1998.

[65] R. Ramamoorthi and P. Hanrahan. An Efficient Representation for Irradiance Environment Maps. *CGIT*, 2001.

[66] T. Rindfleisch. Photometric method for lunar topography, 1966.

[67] Fabiano Romeiro and Todd Zickler. Blind reflectometry. *ECCV*, 2010.

[68] Stefan Roth and Michael J. Black. Fields of experts: A framework for learning image priors. *CVPR*, 2005.

[69] D.L. Ruderman and W. Bialek. Statistics of natural images: Scaling in the woods. *Physical Review Letters*, 1994.

[70] A. Saxena, M. Sun, and A.Y. Ng. Make3d: learning 3d scene structure from a single still image. *TPAMI*, 2008.

[71] Jianbing Shen, Xiaoshan Yang, Yunde Jia, and Xuelong Li. Intrinsic images using optimization. *CVPR*, 2011.

[72] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *TPAMI*, 2000.

[73] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. *ECCV*, 2012.

[74] Marshall F. Tappen, William T. Freeman, and Edward H. Adelson. Recovering intrinsic images from a single image. *TPAMI*, 2005.

[75] D Terzopoulos. Image analysis using multigrid relaxation methods. *TPAMI*, 1986.

[76] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. *ICCV*, 1999.

[77] Oliver Woodford, Philip Torr, Ian Reid, and Andrew Fitzgibbon. Global stereo reconstruction under second-order smoothness priors. *TPAMI*, 2009.

[78] R.J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical engineering*, 1980.

[79] Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: recovering reflectance models of real scenes from photographs. *SIGGRAPH*, 1999.

[80] R. Zhang, P.S. Tsai, J.E. Cryer, and M. Shah. Shape-from-shading: a survey. *TPAMI*, 1999.