Estimating Intrinsic Component Images using Non-Linear Regression

Marshall F. Tappen Edward H. Adelson William T. Freeman
MIT Computer Science and Artificial Intelligence Laboratory
Cambridge, MA 02139
{mtappen, adelson, billf}@csail.mit.edu

Abstract

Images can be represented as the composition of multiple intrinsic component images, such as shading, albedo, and noise images. In this paper, we present a method for estimating intrinsic component images from a single image, which we apply to the problems of estimating shading and albedo images and image denoising. Our method is based on learning estimators that predict filtered versions of the desired image. Unlike previous approaches, our method does not require unnatural discretizations of the problem. We also demonstrate how to learn a weighting function that properly weights the local estimates when constructing the estimated image. For shading estimation, we introduce a new training set of real-world images. The accuracy of our method is measured both qualitatively and quantitatively, showing better performance on the shading/albedo separation problem than previous approaches. The performance on denoising is competitive with the current state of the art.

1. Introduction

An image of a scene can be represented as the composition of a number of different images. At the image formation level, the image can be represented as the sum of a noise image and the true image of the scene. The scene image itself can also be represented as the composition of images that describe the characteristics of the surfaces in the scene. Research in the area of lightness perception [18, 14, 4, 9, 2, 15] has modelled the scene image as the composition of a shading image, which describes the illumination and shape of the surface, and an albedo image, which describes the reflectance of the surface. Figure 1 shows an example of this shading and albedo image decomposition. The noise, shading, and albedo images can be viewed as special types of *intrinsic images* [3], because each of them represents an intrinsic characteristic of the scene – if the camera is included as part of the scene. They are a special type of intrinsic image because the observed image is created directly from these images using pointwise sums and multiplications. In this paper, we will refer to these special intrinsic images as the *intrinsic components* of the observed image.

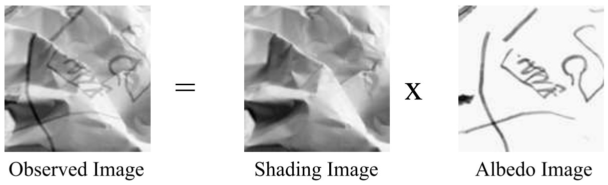


Figure 1. An example of how an image of crumpled paper with scribbling on it can be expressed as the product of two intrinsic component images, a shading image and an albedo image.

The ability to decompose an image into its intrinsic components is useful because applications involving image understanding often rely exclusively on one of the intrinsic characteristics of the scene. For example, recovering the shape of a surface with shape-from-shading techiques requires image data with no changes in albedo. A pure albedo image would be useful for graphics applications if the image is to be altered by changing the color of surfaces.

In this paper, we propose treating the problem of decomposing an image into its intrinsic components as a non-linear regression problem. To recover the shading image, we estimate it directly from the observed image. The high-dimensionality of images makes standard regression techniques infeasible, so we describe how this regression can be accomplished using local, low-dimensional estimators. Our method is an especially significant improvement for recovering shading and albedo images because it avoids the overly-simplified models and artificial discretizations of the problem, discussed in Section 3. Unlike previous work in this area, we demonstrate the improvement of our method quantitatively. We also apply our intrinsic component estimation procedure to a second problem, denoising images, and show it to be competitive with recently published methods for denoising.

While previous methods have also used local estimates to construct an estimate of an intrinsic component image, these methods have not addressed the issue of how to handle uncertainty in the local estimates. In this paper, we demonstrate how to learn a weighting function that weights the different local estimates in order to produce the best estimate possible. This significantly reduces the error in the estimates of the component images.

2. Basic Approach

Viewed as generic non-linear regression, estimating an intrinsic component image with N pixels is an N-dimensional regression problem. Even for small images, this dimensionality is too large to use black-box regression algorithms. To overcome this dimensionality issue, we use local estimation based on small image patches.

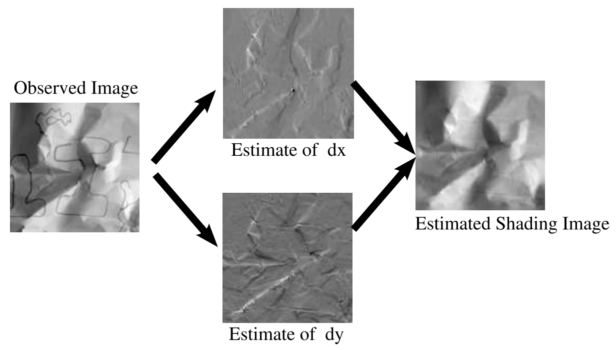


Figure 2. This diagram describes our basic approach to recovering intrinsic component images. In this example, we recover the shading image of a piece of crumpled paper by first estimating the horizontal and vertical derivatives of the shading image. The estimate of the shading image is produced by finding the image that best satisfies these constraints in the least squares sense.

Using local image patches to estimate *pixel* values for an intrinsic component may not be feasible. Consider an image of surface with two large regions of differing albedo. If the region with darker albedo is larger than the image patch used to estimate the intrinsic component image, it is impossible to correctly recover the shading of that area. In the case of reflectance, the classic Retinex algorithm overcomes this by making local estimates of the *image derivatives*, rather than the pixel values. These derivative estimates are then used to estimate the image by solving Poisson's equation.

We adopt a generalization of this approach, allowing a richer set of linear filters than just derivatives. Our steps are:

1. Estimate a set of local linear constraints, such as the derivatives, using local image data. The estimators will utilize non-linear regression, learned from training data.

2. Solve for the image that best satisfies these constraints, by using a method akin to a pseudoinverse.

Figure 2 shows an example of the process.

If the linear constraints are computed at every point, they can be thought of as the desired outputs of linear filters applied to the estimated image. The choice of filters depends on the image statistics. We want filters that capture and distinguish the characteristic statistics of the intrinsic components. In the case of classic Retinex, the assumption is that shading images are smooth, while reflectance images are piecewise constant, so that the processes can be easily separated in the derivative domain. Since real images are more complex, we need to learn more sophisticated statistics in order to do the separation. In general, we may choose to estimate the outputs of any linear operators that capture relevant statistics.

Our main interest here is in separating albedo and shading. However, the same approach can be used for many image estimation problems where the statistics of different image classes are distinct. A good example is denoising: we can apply our technique by training the non-linear regressor on natural images with and without noise. As we will show, our results are competitive with state of the art denoising methods.

3. Previous Work on Recovering Shading and Albedo

The Retinex algorithm [15] was one of the first approaches to estimating the lightness of surfaces. Using a simplified model of intrinsic image statistics, the Retinex algorithm assumes that image derivatives with a large magnitude are caused by changes in the albedo of the surface, while derivatives with a small magnitude are caused by changes in illumination. Under this model, a shading image can be constructed by calculating the derivatives of the observed image, eliminating derivatives with a large magnitude, then reconstructing the image [12]. The disadvantage of this model is that this simple characterization of shading does not hold for many surfaces.

In [18], Tappen et al. generalize this algorithm by learning a classifer that labeled each horizontal and vertical image derivative as either an albedo change or a change in the shading image. The algorithm collects local color information and gray-scale information using an AdaBoost classifier. To disambiguate locally ambiguous areas of the image, this local information is propagated along edges in the image using an MRF. The classifier is trained using examples of shading and albedo changes.

Training the classifier allows it handle surfaces that violated the assumptions of the Retinex. However, it also requires presenting examples that are *either* shading or albedo changes. Training a classifier on a set of real images would require an unnatural division of the image derivatives into the two classes. In Section 7.2, we show that this can lead to significant artifacts.

Like these two approaches, our method first estimates the filtered versions of the intrinsic component image. The first major advantage of our method over Retinex is that we use training data to learn to predict the derivatives of the shading and albedo images, rather than basing the estimates on a simple model of the world. This enables our method to produce better estimates of the images. Our model also lacks the discretization inherent to methods such as [18]. The second significant advantage between our work and these previous methods is the weighting of the estimates. Both of the previous methods assume that the estimates of the horizontal and vertical derivatives should be equally weighted in the pseudo-inverse step. In Section 6, we describe how to learn a function to optimally set the weights of each estimate, downweighting unreliable estimates.

4. Locally Estimating Constraint Values

As described in Section 2, an estimate of an intrinsic component image is produced by first estimating filtered versions of that image. We limit the dimensionality of the estimation problem by only using a patch of the observed image to estimate a particular pixel of the filtered intrinsic component image.

In order to capture the statistics of real-world surfaces, the estimator is trained from images of real surfaces and the corresponding intrinsic component image. Formally, for each filter, we created a training set of M pairs, $(o_1, c_1) \ldots (o_M, c_M)$, where o_i is a vector created by unwrapping a patch of the observed image and c_i is the value of the filtered intrinsic component image at the same pixel as the center of the observed image patch. The estimator, $r(o)$, is trained to minimize the squared error, E, in the predictions of c_i from o_i:

$$E = \sum_i^M (r(o_i) - c_i)^2 \tag{1}$$

4.1. Using a Mixture of Experts Estimator

With the goal of a simple, yet powerful approach, the estimator, $r(o)$, is parameterized as a set of N prototype patches, $p_1 \ldots p_N$, a set of linear regression coefficients, $f_1 \ldots f_N$, associated with each prototype point, and a scale factor h. Given an observed image patch, o, the estimate of the filtered intrinsic component image, $r(o)$, is computed as:

$$r(o) = \frac{\sum_{i=1}^N \left(e^{-\sum_j (p_i^j - o^j)^2}\right)\left(f_i^T o\right)}{\sum_{i=1}^N e^{-\sum_j (p_i^j - o^j)^2}} \tag{2}$$

where the superscript j indexes into the patches o and p_i.

This estimator can be viewed as a Mixture of Experts estimator. [13] A prototype patch p_i and the corresponding regression coefficients, f_i, can be viewed as an expert that describes how to predict c from o for observed image patches that are similar to p_i. The actual estimate, $r(o)$, is an average of these individual predictions, weighted by the distance from the o to the prototype patch.

We chose to base the estimator on patches taken from the training set to give the estimator flexibility in capturing the statistics of the intrinsic component images. Limiting the number of experts in the estimator is important because of the computational demands of our approach. Estimating an intrinsic component image involves estimating pixel values at *every* pixel in the image. The estimator must be computationally efficient for estimates to be available in reasonable amounts of time. Fixing the number of patches in the estimator is straight-forward method of controlling the computation needed to produce estimates.

4.2. Choosing Prototype Patches

Training this Mixture of Experts estimator consists of choosing the N prototype patches, $p_1 \ldots p_N$, and the corresponding linear regression coefficients, $f_1 \ldots f_N$. The prototype patches are added one at a time until N patches have been added to the set of prototype patches. This minimizes the error criterion in a stage-wise fashion by repeatedly adding an expert, or weak learner, and setting the paramters for that stage, and thus can be categorized as a boosting algorithm [10]. We refer to this algorithm for choosing the prototype patches and regression coefficents as ExpertBoost.

Algorithm 1 describes the ExpertBoost algorithm for adding patches to the prototype database. In Steps 1-6, the algorithm is initialized by setting the first prototype to be an arbitrary training sample. The associated linear regression coefficients are chosen to be the optimal linear regression from the observations to the ground-truth filter responses, c. At each iteration in the algorithm, the training example with the largest error is added as the next prototype in Step 8. The regression coefficients associated with that prototype is chosen in Steps 9-13 to minimize the squared error between the new estimates and the $c_1 \ldots c_M$. To prevent the estimator from focusing solely on relatively small regions with higher error, we modified step 8 to occasionally choose the next expert at random.

We found that the Mixture of Experts estimator produced good results, while including relatively few training examples in the estimator. For comparison, in the experiments detailed in Section 7, we also trained a Support Vector Regression estimator using libSVM [8]. The SVR estimator required nearly twice as many support vectors to achieve comparable prediction performance.

Algorithm 1 ExpertBoost

```
Require: $N$, the number of prototype patches to be added to the model. A set of $M$ training examples, $(o_1, c_1) \ldots (o_M, c_m)$. A scale parameter, $h$
 1: $i \leftarrow 1$
 2: Choose an arbitrary observation to be the initial prototype, $p_1$.
 3: $f_1 \leftarrow \arg\min_f \sum_{m=1}^M (f_1 \cdot o_m - c_m)^2$
 4: for $m = 1 \ldots M$ do
 5:     set $r_m^i$ to the linear estimate of $c_m$ from $o_m$
 6: end for
 7: for $i = 2 \ldots N$ do
 8:     Add a patch, $o_n$, to the prototype database by setting patch $p_i = o_n$, where $n = \arg\max_n (r_n^i - c_n)^2$
 9:     for $m = 1 \ldots M$ do
10:         $d_k \leftarrow e^{-\sum_j (o_m^j - p_n^j)^2}$, where $p_i^j$ denotes value $j$ in the patch with index $i$. We use similar notation for the observations $o$.
11:         $d_k^0 \leftarrow \sum_{l=1}^i e^{-\sum_j (o_m^j - p_l^j)^2}$
12:     end for
13:     Set $f_i = \arg\min_f \sum_{k=1}^M \left(\frac{d_k^0 r_k^{i-1} + d_k (f_i^T o_k)}{d_k^0 + d_k}\right)^2$
14:     for $m = 1 \ldots M$ do
15:         Update $r_k^i$ to be the current estimate of $c_k$ from $o_k$.
16:     end for
17: end for
```

4.3. Extending to Alternate Criteria

The ExpertBoost algorithm is not limited to minimizing the squared error. The error function can be replaced with other cost functions, such as robust cost functions. This is done by replacing Step 13 with

$$f_i = \arg\min_f \sum_{k=1}^M g\left(\frac{d_k^0 r_k^{i-1} + d_k(f_i^T o_k)}{d_k^0 + d_k}\right) \tag{3}$$

where $g(x)$ is an arbitrary cost function. The Lorentzian [7] robust cost function is one alternative:

$$g(x) = \log(1 + \frac{1}{2}x^2) \tag{4}$$

It is not possible to solve Equation 4 in closed form, so we minimize it with the non-linear conjugate gradient routine from the NETLAB package.[16] In practice, this is much slower than the solving the least-squares problem.

The ExpertBoost algorithm can also be extended to a classification framework, allowing us to compare with the approach of [18]. In this case the classifier function takes the form:

$$r(\hat{p_O}) = \text{sign}\left(\frac{\sum_{i=1}^{N}\left(e^{-\sum_j (p_i^j - p_O^j)^2}\right) b_i}{\sum_{i=1}^{N} e^{-\sum_j (p_i^j - p_O^j)^2}}\right) \tag{5}$$

where $b_1 \dots b_N$ are scalars.

The the prototype patches, $p_1 \dots p_N$ and the associated $b_1 \dots b_N$ are chosen to minimize the exponential loss between the predicted labels and the ground truth labels. More information on the exponential loss can be found in [11]. After choosing patch p_i, b_i is found by solving a quadratic Taylor series expansion of the exponential loss at $b_i = 0$, similar to the GentleBoost algorithm [11].

4.4. Taking Advantage of Multiscale Information

Until this point, we have assumed that the image patches were taken directly from the image. Unfortunately, the dimensionality of the patches rises quickly as larger areas of the image are considered. To work with larger patches of the image, while avoiding the "curse of dimensionality" [5] and its problems, we use a multiscale representation of the image patches. To create a multiscale patch, we constructed a three level Laplacian Pyramid [1] using a five-tap filter, without downsampling between levels. The multiscale patch is then created by taking a 3×3 patch from each level of the pyramid. This multiscale representation allows each patch to effectively represent a larger area of the image with only a slighly larger number of dimensions. We also found that when estimating derivatives, the estimators performed better when we subtracted the mean of the 3×3 patch taken from low-frequency residual of the Laplacian Pyramid.

5. Reconstructing the Image

Each of the estimates of the linear functions, such as the derivatives used in the previous section, can be viewed as a constraint on the estimated image. The final stage in the algorithm is to find the image that best satisfies these constraints in the least-squares sense.

This can be expressed formally if $\hat{x}$ is defined to be the shading image predicted from the observation o. For concreteness, assume that there are two linear constraints: a horizontal discrete derivative and a vertical discrete derivative. Let C_{dx} and C_{dy} denote the matrices that express the 2-D image convolution with each filter as a matrix. We assume that the convolution operation produces only those pixels where the convolution kernel is fully contained inside the image, making each of the matrices have less rows than columns. We also denote the estimated value of each of these derivatives as $\hat{c}_{dx}$ and $\hat{c}_{dy}$. The estimated image $\hat{x}$ is recovered using the Moore-Penrose pseudo-inverse:

$$\hat{x} = \left(C^T C\right)^{-1} C^T \hat{c} \tag{6}$$

where

$$C = \begin{bmatrix} C_{dx} \\ C_{dy} \end{bmatrix}$$

and

$$\hat{c} = \begin{bmatrix} \hat{c}_{dx} \\ \hat{c}_{dy} \end{bmatrix}$$

It is important to note that the constraints are not limited to first-order derivatives. The matrix C can be any arbitrary linear combination. For some combination of functions, such as derivatives, C will be a singular matrix. In these cases, we include a constraint specifying the value of a small number of pixels. For all of the results shown, $\hat{x}$ is found using Gaussian Elimination in MATLAB (the "backslash" operator). If computation is an issue, C can also be solved with the conjugate gradient algorithm or FFT [19].

6. Learning to Assign Weights to Constraints

When constructing the estimate of the intrinsic component image, previous systems have assumed that all of constraints imposed by the estimated filter responses should be equally weighted. However, it is likely that some estimates of the linear functions are more reliable than others. As discussed in [18], the correct filtered image value is often ambiguous given only a patch of local image information. Estimates in ambiguous patches should receive less weight than estimates unambiguous image patches.

In this section, we demonstrate how to learn the proper weighting of different estimates by learning a function that weights a filter response according to appearance of the image patch from which it was estimated.

6.1. Forming the Weighting Matrix

Additional knowledge about the reliability of these estimates can be incorporated into the pseudo-inverse step by adding a diagonal weighting matrix, W:

$$\hat{x} = \left(C^T W C\right)^{-1} C^T W \hat{c} \tag{7}$$

Our goal is to construct W so as to minimize the squared error between the estimated intrinsic component images and the ground-truth images. We accomplish this by making W a function of the observed image, $\mathcal{I}$. We denote this matrix $W(\mathcal{I})$. The weight assigned to a particular estimated filter response is based on an image patch from the same location in the observed image. If the observed image patch is similar to patches where estimates are known to be poor, a low weight is assigned to the filter response. Responses in patches that are known to be reliable receive high weights.

We denote matrix $W_i(\mathcal{I})$ as the weight associated with estimator i. Each element along the diagonal is denoted as

w_i^j for the jth element along the diagonal of $W_i(\mathcal{I})$. For a system with I linear filters, the complete weighting matrix, W, is a block-diagonal matrix with $W_1(\mathcal{I}) \ldots W_I(\mathcal{I})$ along the diagonal.

Given this model for weighting estimates, it is natural to use prototype patches that parameterize the ExpertBoost estimators as the basis for judging the quality of an estimate. This allows some experts to be flagged as unreliable. We model the weight of an estimated filter response as a function of the normalized distance from an image patch surrounding corresponding location in the observed image to set of prototype patches, $p_1 \ldots p_N$ associated with the estimator:

$$w_i^j = \sum_{n=1}^{N} (\alpha_i^n)^2 \frac{e^{-\sum_s (o_j^s - p_n^s)^2}}{\sum_{n=1}^{N} e^{-\sum_s (o_j^s - p_n^s)^2}} \qquad (8)$$

Each α term is squared simply to ensure that it is not negative. The coefficient α^2 should be close to zero for an expert that produces poor quality estimates. Hence, filter responses from patches similar to the prototype defining that estimator will receive low weight.

6.2. Learning the weights

The weights, α, are found by minimizing the squared error between the ground truth images, x and the predicted images, $\hat{x}$. Using the formulation of $\hat{x}$ in Equation 7, the mean squared error for a training set of K images can be expressed as

$$\sum_{k=1}^{K} (A_k \hat{c_k} - x_k)^T (A_k \hat{c_k} - x_k) \qquad (9)$$

where

$$A_k = \left(C^T W_k C\right)^{-1} C^T W_k \qquad (10)$$

and $\hat{c_k}$ is a vector containing the estimates of the filtered component images for the ith image. The summation is over the K images in the training set. For brevity, we use W_k in place of $W(\mathcal{I}_k)$, where $\mathcal{I}_k$ is the kth image in the training set.

Letting z denote this error, the derivative of the error with respect to a weight coefficient α_i^n is

$$\frac{\partial z}{\partial \alpha_i^n} = \sum_{k=1}^{K} 2 (A_k \hat{c_k} - x_k) \left(\frac{\partial A_k}{\partial \alpha_i^n} \hat{c}_k\right)^T \qquad (11)$$

The derivative $\frac{\partial A_k}{\partial \alpha_i^n}$ can be derived using the product rule for matrices. The matrix $\frac{\partial W_k}{\partial \alpha_i^n}$ is derived using Eq. 8. The weights can be found using gradient descent to minimize Eq. 9.

7. Experiments

The results produced by our method depend on the choice of both cost function and the representation of the patches. In this section, we evaluate the performance of the algorithm using different settings for both of these choices, compare our method against an estimator that relies on the same heuristic as the Retinex algorithm, and evaluate the benefit of weighting the filter responses.

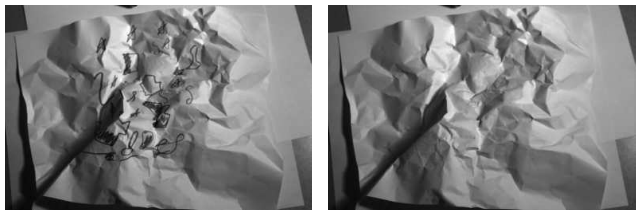

(a) Red Channel (b) Green Channel

Figure 3. These images are the red and green channel of a photograph of a piece of paper colored with a green maker. The coloring can be seen in (a) but not in (b). We use this property to construct a training set.

7.1. Creating Shading and Reflectance Data

Effectively comparing algorithmic decisions requires images of real surfaces, along with corresponding shading and albedo decompositions. Even in controlled settings, with multiple cameras and light sources, it is difficult to create shading and albedo images without significant artifacts in the images. However, we developed a simple method to create a set of ground-truth decompositions, using color to measure shading separately from albedo. Figure 3(a) shows the red channel of an image taken of a piece of paper colored with a Crayola "Electric Lime-Green" Washable Marker and illuminated with a standard incandescent light. Both the markings and the surface shading is visible. The green channel of that image, shown in Figure 3(b), does not contain any of these markings. We use this image as the shading image.

In order to collect data with different types of surface statistics, the training and test sets were collected using three types of paper: office paper, paper towels, and facial tissue. After coloring the paper with a green marker, we used a Canon 10D in RAW mode, which has an approximately linear response, to capture each image. The albedo image was extracted dividing the red and green channels of the image. After finding the shading and albedo images, the data set was created from the log of these images to make the problem additive. This dataset provides photgraphs of real-world surfaces with high-quality shading and albedo image decompositions. The database will be available for download at `http://www.csail.mit.edu/~mtappen`.

Testing on crumpled paper images is useful because these images violate the basic assumptions of Retinex. The shading images have strong edges that would be considered albedo changes under the Retinex model. To our knowledge, previous work has not attempted to recover the shading for images as difficult of these.

7.2. Evaluating performance

To evaluate the performance of our method, we used a training set of 22 paper images and a test set of 11 paper images. In addition, we added ten synthetic images, such as those used in [18], to the training set. We found that our method generalized to the examples used in [18] much better with the addition of these images. Four similar images were also added to the test set. The paper images were all created

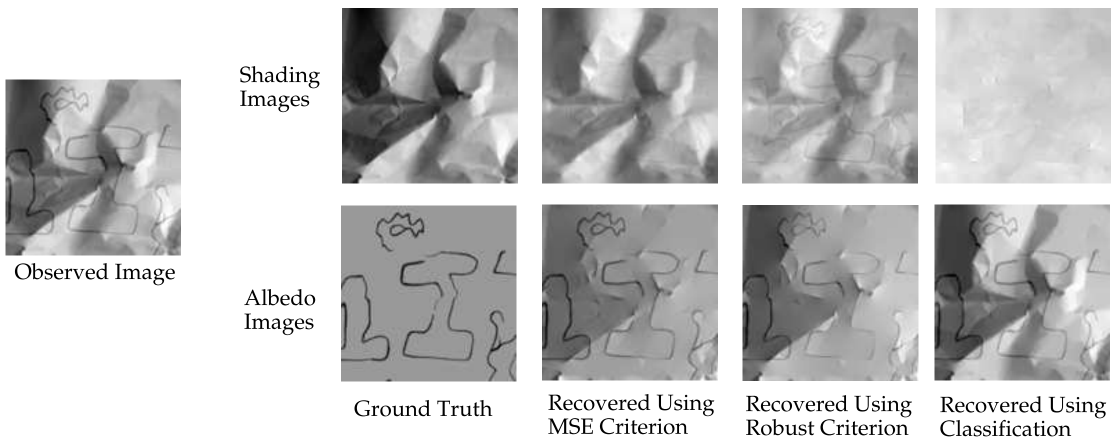

Shading Images
Albedo Images
Observed Image
Ground Truth
Recovered Using MSE Criterion
Recovered Using Robust Criterion
Recovered Using Classification

Figure 4. The shading and albedo images generated from the observed image on the left. The different images are created by using different cost functions, described in the text, to train the estimators of the horizontal and vertical derivatives. We found estimators trained using a squared-error criterion produced the most visually appealing images. Minus a constant, each of the image pairs add together to form the observed image. The mean squared errors for the three shading images, from left to right, are 8.4×10^6, 2.4×10^7, 4.8×10^7

using the method described in Section 7.1. The size of all the images was 127 $\times$ 127 and each image ranged roughly between 0 and 255.

7.2.1 Evaluating the choice of the cost function

We first investigated the effect of using either the Lorentzian error criterion or the squared-error criterion. For both criteria, we used the ExpertBoost algorithm to learn an estimator for the [-1 1] horizontal derivative of the shading image and a second estimator for the [-1 1] vertical derivative. We used the multiscale patches described in Section 4.4 as the representation of the input data. The final shading images were recovered using the method described in Section 5. The estimators trained using the least squares cost functions were each trained using 1000 prototype patches. When training with the Lorentzian cost function, there was little improvement in the training error after adding 400 prototype patches. This, coupled with the significant computation required to compute the regression coefficients, led us to use 400 prototype patches for the estimator trained using the Lorentzian cost function. The scale factor, h in Eq. 2, for these experiments was 384.

To compare our approach with the classification-based approach described in [18], we converted the estimation problem into a classification problem by assigning each derivative in the training set a label of +1 if the magnitude of the derivative was above a threshold of 2 gray levels, out of 255 possible gray levels. Otherwise, the derivative received the label -1.

Figure 4 shows an example of the shading and albedo images produced using the estimators trained under each of the cost functions. The albedo images were produced by subtracting the estimated shading image from the observed image. Generally, we found the results produced by the least-squares cost function to the most visually pleasing. The estimator trained using the robust cost function generally produced shading images with more artifactual albedo content.

Algorithm	Training Error ($\times10^5$)	Testing Error ($\times10^5$)
Retinex	36.9	32.7
Basic 5 $\times$ 5 System	17.3	20.0
Using Multiscale Patches	14.3	16.6
Weighting Estimates	5.61	9.78

Figure 5. The average squared error per image between the ground-truth shading images and the images produced using different types of estimators. The lowest error is achieved using multiscale patches and the weight adjustment method described in Section 6. For both the training and test set, the error is summed over the whole set.

For the rest of the experiments presented in this paper, the error criterion is the squared-error. In the shading and reflectance problem, there is a constant of integration that cannot be recovered using only derivatives. For this reason, we subtracted the mean error from each image before computing the error in the image.

7.2.2 Evaluating the Multiscale Patches

To evaluate the usefulness of the multiscale patches, we trained the estimators described above, but used 5 $\times$ 5 image patches as the observation vectors instead of the multiscale patches described in Section 4.4. As with the multiscale patches, we subtracted the mean from each the 5×5 patches. As shown in Figure 5, using the multiscale patches decreases the test error by 19%.

7.2.3 Comparing with Retinex

We also compared the estimator trained with ExpertBoost against the results produced using an estimator based on the assumption underlying the Retinex algorithm, which is that derivatives with small magnitudes are caused by shading and derivatives with large magnitude are caused by changes in the albedo of the scene. Given the value of a derivative in the

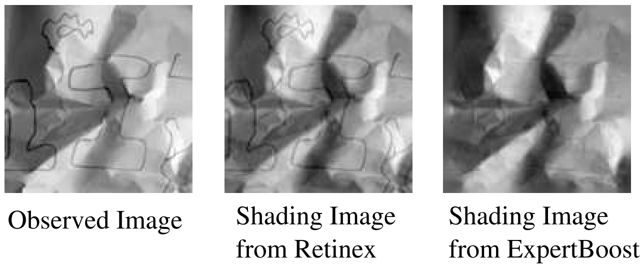


Figure 6. Shading images estimated using ExpertBoost estimators and Retinex estimators (see Section 7.2.2). The image produced using the ExpertBost estimators retains little of the scribbling, while the Retinex estimators are unable to eliminate the scribbling. [The contrast of each image is set individually to fill the whole range of image values.]

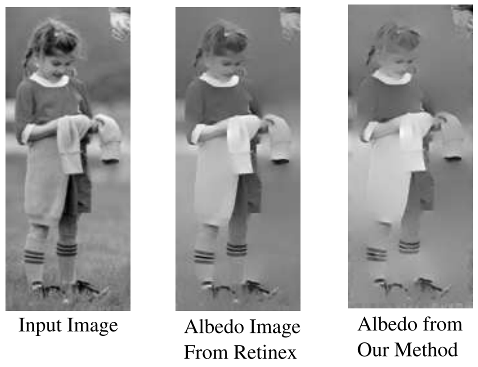


Figure 7. The ExpertBoost estimators are also able to perform as well as Retinex on images where Retinex does well, such as this example from [18].

observed image, δo, the estimate of the shading derivative, $\hat{c}_{dx}$ is

$$\hat{c}_{dx} = \frac{\delta o}{1 + e^{a|\delta o| + b}} \tag{12}$$

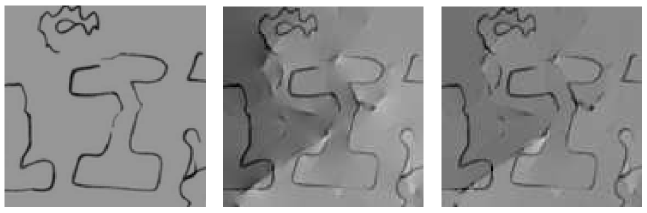
The sigmoid function in Equation 12 acts to set derivatives with large magnitudes to 0.

We found the results produced by our estimators to be much superior both visually and in terms of squared error. Figure 6 shows shading images returned by the ExpertBoost estimators and the Retinex estimators. Much more of the scribbling is left over in the images found using the Retinex estimators than in the images found with the ExpertBoost estimators. This is corroborated by the differences in error. As shown in Figure 5, the error in the test images produced using the Retinex estimators is over twice the error in the images produced using the ExpertBoost estimators.

The ExpertBoost estimators also performs well on images where Retinex does well, such as the image shown in Figure 7. In this image, the arm of the girl has less shading in the image produced by the ExpertBoost estimators than in the image produced with the Retinex estimators.

7.2.4 Evaluating Weighting the Estimates

Using the same training set and constraints described in Section 4.3, we used optimized Eq. 9 using a basic gradient descent algorithm. Similar to the "bold driver" method [6], the algorithm decreases the step size when the error rises instead of falls. The weights were all initialized to 1, which is equivalent to the set-up in Section 7.2.2 and the third row of Figure 5. As seen in the fourth row of Figure 5, optimizing the weights decreases the error rates achieved using just the multiscale patches by 37%. This can also be seen qualitatively in Figure 8. The albedo image found using weighted derivative estimates, shown in Figure 8(c) appears much flatter than the albedo image recovered without weighting the derivative estimates, shown in Figure 8(b).



(a) Ground Truth Albedo Image (b) Estimated Albedo without Adjusting Weights (c) Estimated Albedo after Adjusting Weights

Figure 8. These images demonstrate the benefit of adjusting the weights when reconstructing the image. After adjusting the weights, the reconstructed albedo image, shown in (c), appears much flatter. To enable better comparison, the contrast of these images is different than the contrast in Figure 4.

8. Application to Denoising

To demonstrate the ability of our method to generalize to other types of intrinsic components, we trained our system to estimate a scene image from an observation corrupted with Gaussian white noise. For this application, the intrinsic components are the clean image and the noise. For denoising, we obtained the best results by using six filters as the basis for the system: two first derivative filters, three second derivative filters, and a Gaussian filter with standard deviation 1 that was truncated to a 3×3 filter.

We trained the estimators using a set of 64 images taken from the Berkeley Segmention Database. The images were scaled to one-half their original size, cropped to size 128×128, and added to a Gaussian white noise image with a standard deviation of 10. The ExpertBoost algorithm chose 2400 prototype patches for each estimator. The estimators still performed well when fewer prototype patches were used.

For evaluation, we used a set of 40 test images from the Berkeley database. Each of these images was processed in the same manner described above. For comparison, we used the Field of Experts code provided by Roth et al. [17], which has achieved results close to the best-published results for denoising. As shown in Table 1, our method is competitive with the Field of Experts, although the PSNR, computed in the same manner as [17], of the results from the Field of Experts algorithm is about 0.2 dB higher. The primary difference between the results of the two algorithms appears to be in the low spatial-frequency bands of the images. The second column of Table 1 shows the PSNR of images created by high-pass filtering the results from the two algorithms.

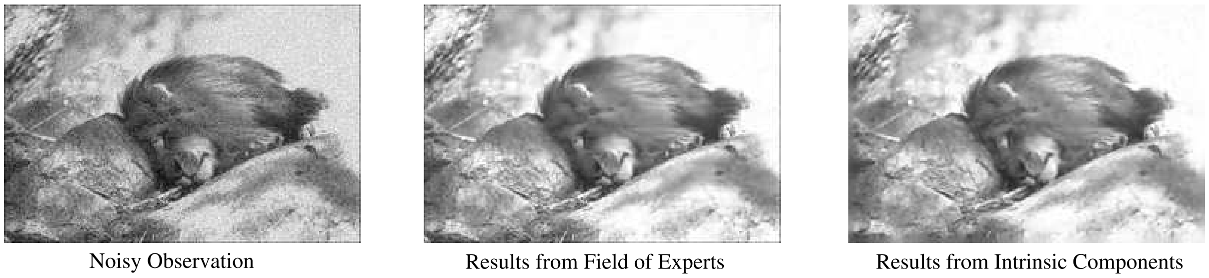

Noisy Observation
Results from Field of Experts
Results from Intrinsic Components

Figure 9. An example of using our method for denoising. The image on the left has been corrupted with white Gaussian noise ($\sigma = 10$). The results produced by our method are competitive with the recent Field of Experts Model.

Algorithm	PSNR	PSNR of High-Pass Filtered Image
Field of Experts	32.46	34.58
Intrinsic Components	32.26	34.51

Table 1. The PSNR in dB of denoised images produced by our method and the Field of Experts algorithm. The second column compares the PSNR of high-pass filtered versions of the results.

For these images, the PSNR is almost identical. As shown in Figure 9, the results are also visually similar. Using the weight adjustment technique described in Section 6 gave the system a small 0.06 dB increase in the PSNR. We expect that applying our algorithm at multiple scales would lead to further increases in the PSNR.

9. Conclusion

An intrinsic component image describes intrinsic characteristic of a scene, such as the shading or albedo, as an image. An image can be modelled as the composition of a scene's intrinsic component images. In this paper, we have described a method for estimating the intrinsic component images of a scene from a single image. We estimate an intrinsic component by first estimating a set of local linear constraints, such as derivatives, from patches of the observed image. The component image is then found by solving for the image that best satisfies these constraints. We have also introduced a novel method that enables accounts for the uncertainty in the estimates of the constraints when solving for the estimated intrinsic component.

The effectiveness and flexibility of our method enable it to be used for both denoising images and estimating shading and albedo images. To demonstrate our method's effectiveness we have introduced a new set of real-world albedo and shading images. This data set will be made publicly available. On this set, which is more difficult than those used previously, our method is found to outperform previous approaches, both the classification approach of [18] and an estimator based on the same assumptions as Retinex.

Our method for computing intrinsic components can also be applied to image denoising by splitting the noisy image into a clean image and a noise image. The intrinsic component method performs comparably to state of the art methods for denoising.

Acknowledgments

This work was supported by a grants from Shell Oil, the National Geospatial-Intelligence Agency, NGA-NEGI, and the Nippon Telegraph and Telephone Corporation as part of the NTT/MIT Collaboration Agreement.

References

[1] E. H. Adelson and P. Burt. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236, 1983.

[2] E. H. Adelson and A. P. Pentland. The perception of shading and reflectance. In D. Knill and W. Richards, editors, *Perception as Bayesian Inference*, pages 409–423. Cambridge University Press, New York, 1996.

[3] H. G. Barrow and J. M. Tenenbaum. Recovering intrinsic scene characteristics from images. In A. Hanson and E. Riseman, editors, *Computer Vision Systems*, pages 3–26. Academic Press, 1978.

[4] M. Bell and W. T. Freeman. Learning local evidence for shading and reflectance. In *Proceedings International Conference on Computer Vision*, 2001.

[5] R. E. Bellman. *Adaptive Control Processes*. Princeton University Press, 1961.

[6] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[7] M. J. Black and P. Anandan. Robust dynamic motion estimation over time. In *IEEE CVPR*, pages 296–302, 1991.

[8] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at *http://www.csie.ntu.edu.tw/ cjlin/libsvm*.

[9] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *International Journal of Computer Vision*, 40(1):25–47, 2000.

[10] J. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29:1180, 2001.

[11] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, 2000.

[12] B. K. P. Horn. *Robot Vision*, chapter 9. MIT Press, Cambridge, Massachusetts, 1986.

[13] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[14] R. Kimmel, M. Elad, D. Shaked, R. Keshet, and I. Sobel. A variational framework for retinex. *International Journal of Computer Vision*, 52(1):7–23, 2003.

[15] E. H. Land and J. J. McCann. Lightness and retinex theory. *Journal of the Optical Society of America*, 61:1–11, 1971.

[16] I. Nabney. Netlab neural network software. http://ncrg.aston.ac.uk/netlab/index.php.

[17] S. Roth and M. Black. Field of experts: A framework for learning image priors. In *IEEE CVPR*, volume 2, pages 860–867, 2005.

[18] M. F. Tappen, W. T. Freeman, and E. H. Adelson. Recovering intrinsic images from a single image. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

[19] Y. Weiss. Deriving intrinsic images from image sequences. In *Proceedings International Conference on Computer Vision*, Vancouver, Canada, 2001. IEEE.