Recovering Intrinsic Images from a Single Image

Marshall F. Tappen, William T. Freeman, *Member*, *IEEE*, and Edward H. Adelson, *Member*, *IEEE*

Abstract—Interpreting real-world images requires the ability distinguish the different characteristics of the scene that lead to its final appearance. Two of the most important of these characteristics are the shading and reflectance of each point in the scene. We present an algorithm that uses multiple cues to recover shading and reflectance intrinsic images from a single image. Using both color information and a classifier trained to recognize gray-scale patterns, given the lighting direction, each image derivative is classified as being caused by shading or a change in the surface's reflectance. The classifiers gather local evidence about the surface's form and color, which is then propagated using the Generalized Belief Propagation algorithm. The propagation step disambiguates areas of the image where the correct classification is not clear from local evidence. We use real-world images to demonstrate results and show how each component of the system affects the results.

Index Terms—Computer vision, machine learning, reflectance, shading, boosting, belief propagation.

1 Introduction

The appearance of a scene depends on many characteristics, such as the illumination of the scene, the shape of the surfaces in the scene, and the reflectance of each surface. Each of these characteristics contains useful information about the objects in the scene. Barrow and Tenenbaum [1] proposed using *intrinsic images* to represent these characteristics. For a given intrinsic characteristic of the scene, the pixel of the corresponding intrinsic image would represent the value of that characteristic at that point. The intrinsic image representation may be useful as a stepping-stone to higher level analysis.

In this paper, we describe a system for decomposing a single image into two intrinsic images—a shading image (the illumination at each point) and a reflectance image (the albedo at each point). In setting up the problem this way, we make the implicit assumption that the surfaces are Lambertian, so that the observed image is the product of the shading and reflectance image. Fig.1 shows how the image in Fig.1a would be decomposed into an image representing the shading of each point and an image representing each point's reflectance. For many real-world surfaces, this model does not capture everything about the image; for example, there is no correct way to describe a specular highlight as belonging to a reflectance or shading image. Nonetheless, the Lambertian assumption offers a tractable starting point from which we can develop techniques for image decomposition. Future work with more sophisticated models should offer improved performance.

The shading and reflectance intrinsic images are found by classifying each derivative in the image as either being caused by shading or a reflectance change. The derivative classifiers are found by training on a set of example shading and reflectance images. Our system uses classifiers that take advantage of both color image information and gray-scale patterns. In addition, the classification of each derivative is propagated to neighboring derivatives. Once the derivatives have been correctly classified, the intrinsic images can be recovered.

In Section 2, we relate this work to previous work, particularly in the area of lightness recovery. Our basic method for decomposing an image into intrinsic images is described in Section 3. Section 4 describes the classifiers separate from the effects of shading from reflectance changes. Section 5 describes how these classifications can be improved by using a Markov Random Field to incorporate information about the classifications of neighboring derivatives.

2 Prior Work

Much of the early work on decomposing an image into shading and reflectance images was motivated by the study of human lightness perception. When viewing a scene, the human visual system may attempt to remove the effects of illumination in order to accurately judge the reflectance of a surface.

An early algorithm which attempted to recover the reflectance and illumination of a scene is the Retinex algorithm [14]. Retinex originally described lightness perception for "Mondrian" images. A Mondrian image consists of a collage of patches, each with a different reflectance. The patches are lit with a slowly varying illumination. The true reflectance of each patch, within a constant scale factor, can be found by examining the derivatives of the log of the observed image. Since the illumination varies slowly, a large derivative likely indicates a reflectance change at the border of two patches. On the other hand, a derivative with a small

- *M.F. Tappen and W.T. Freeman are with the MIT Computer Science and Artificial Intelligence Laboratory, The Stata Center, Building 32, 32 Vassar Street, Cambridge, MA 02139.*
  E-mail: mtappen@csail.mit.edu, billf@mit.edu.
- *E.H. Adelson is with the MIT Computer Science and Artificial Intelligence Laboratory, The Stata Center, Building 32, 32 Vassar Street, Cambridge, MA 02139 and the Department of Brain Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139.*
  E-mail: adelson@csail.mit.edu.

Manuscript received 18 Mar. 2004; revised 8 Oct. 2004; accepted 23 Nov. 2004; published online 14 July 2005.
Recommended for acceptance by D. Forsyth.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0137-0304.

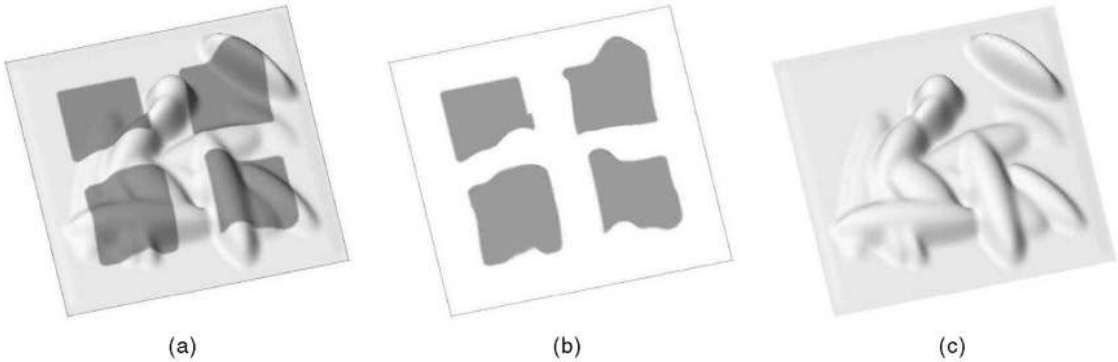

Fig. 1. Example of shading and reflectance intrinsic images. (a) Image of a scene. (b) The reflectance intrinsic image. This image contains only the reflectance of each point. (c) Shading intrinsic image. This image results from the interaction of the illumination of the scene and the shape of the surface.

magnitude is probably caused by the illumination. The reflectance of each patch is recovered by reintegrating the reflectance derivatives. Retinex was originally proposed to work on lines in the image; however, Horn [12] proposed a method for implementing this on 2D images. A similar strategy was also used by Finlayson et al. [4] to remove shadows from images. Color cues were used to find derivatives that were caused by shadows. After setting these derivatives to zero, the shadow-free image was reconstructed using the remaining derivatives.

Retinex and related algorithms rely on the assumption that the changes in the reflectance of a surface will lead to large derivatives, while illumination, which varies slowly, causes small derivatives. However, this assumption may not hold in real images. Freeman and Viola [6] used a smoothness prior on the inferred shape in an image to classify the image as either entirely created by shading or all due to reflectance changes. Instead of relying on just the magnitudes of the derivatives, Bell and Freeman [2] trained a classifier to use the magnitude of the output of a set of linear features. The classifier was trained to label the coefficients of a steerable pyramid [20] as caused by either shading or a reflectance change. Using steerable pyramid coefficients allowed the algorithm to classify edges at multiple orientations and scales. However, the steerable pyramid decomposition has a low-frequency residual component that cannot be classified. Without classifying the low-frequency residual, only band-pass filtered copies of the shading and reflectance images can be recovered. In addition, low-frequency coefficients may not have a natural classification as either shading or reflectance.

A second heuristic, which is related to Retinex, is that the shading and reflectance images can be found by filtering the log of the input image [17]. This approach assumes that the shading component is concentrated in the low spatial frequency bands of the log input, while the reflectance image can be found from the high spatial frequencies. Like the assumption underlying Retinex, this assumption also tends not to be true in real images. Fig. 2 shows a high and low-pass filtered version of the log of the image shown in Fig. 1a. Shading and reflectance changes appear in both images because neither are band-limited. This makes it impossible for band-pass filtering to isolate either shading or reflectance changes.

An alternative to these discriminative approaches, which attempt to distinguish the effects of shading and reflectance, are generative approaches, which create possible surfaces and reflectance patterns that explain the image, then use a model to choose the most likely surface. Previous generative approaches include modeling worlds of painted polyhedra [21] or constructing surfaces from patches taken out of a training set [5].

In a different direction, Weiss [23] proposed using multiple images where the reflectance is constant, but the illumination changes. This approach was able to create full frequency images, but required multiple input images of a fixed scene. Images with varying illumination are also used in [16] to eliminate shadows from surveillance images.

Our system takes a discriminative approach by classifying the derivatives of the image using both classifiers based on color information in the image and classifiers trained to recognize local image patterns to distinguish derivatives caused by reflectance changes from derivatives caused by shading. We also address the problem of ambiguous local evidence by using a Markov Random Field to propagate the classifications of those areas where the evidence is clear into areas of ambiguous classification.

3 Separating Shading and Reflectance

The input image at each spatial position x and y, $\mathcal{I}(x,y)$, is modeled as the product of the shading image, $\mathcal{S}(x,y)$, and the reflectance image, $\mathcal{R}(x,y)$:

$$\mathcal{I}(x,y) = \mathcal{S}(x,y) \times \mathcal{R}(x,y). \quad (1)$$

Our goal is to recover $\mathcal{S}(x,y)$ and $\mathcal{R}(x,y)$ from $\mathcal{I}(x,y)$. Instead of estimating $\mathcal{S}(x,y)$ and $\mathcal{R}(x,y)$ directly, we attempt to estimate their derivatives, which localizes the estimation. Consider the point marked with a white "X" in Fig. 3a. There

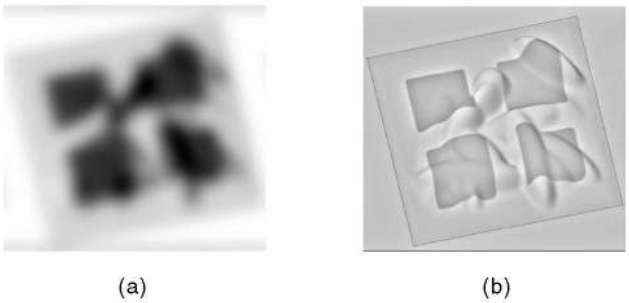

Fig. 2. These high and low-pass filtered versions of the log of Fig. 1a show the infeasibility of solving this problem with simple filtering. Neither the shading nor the reflectance changes in this image are band-limited, so band-pass filtering cannot isolate one or the other.

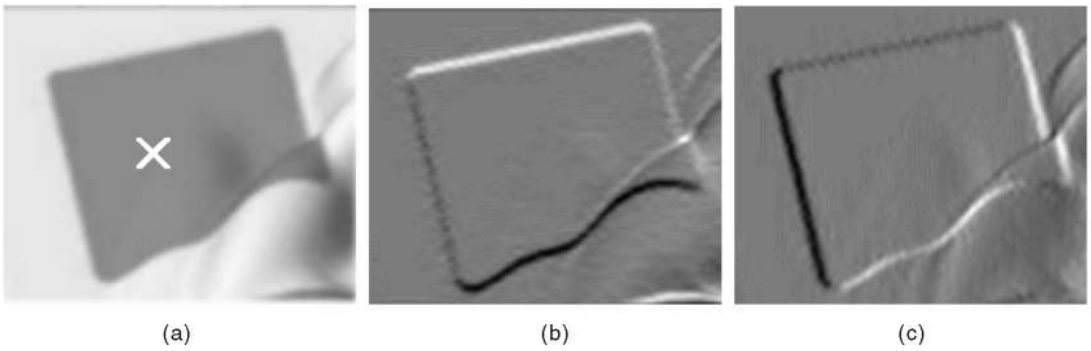

Fig. 3. It is easier to first estimate the derivatives of shading and reflectance images than directly estimate the actual pixel values. To classify the point marked with an "X" in (a), the estimator must use the information from the borders of the patch. If the derivatives are used instead, the derivatives on the borders of the patch can be estimated instead, likely using information in a local area around each derivative. (a) Original image. (b) Vertical derivatives. (c) Horizontal derivatives.

are clearly no changes in the reflectance pattern of the surface at this point, but correctly recovering the reflectance at this point requires understanding that the point is contained in a dark patch that reflects less light. This information is contained at the edges of the patch, which are relatively distant from the point being estimated. In the derivative images, shown in Figs. 3b and 3c, the points inside and outside the patch have the same intensity. Correctly recovering the shading and reflectance components of the whole patch only requires correctly decomposing the derivatives along the border of the patch. Correctly decomposing the derivatives at the edge of the patch is a more localized problem because the information in the image needed to properly decompose those derivatives is likely located along the edge.

Instead of trying to estimate the shading and reflectance component of each image derivative, we assume that it is unlikely that significant shading boundaries and reflectance edges occur at the same point. This allows us to treat every image derivative as either caused by shading or reflectance which reduces the problem of specifying the shading and reflectance derivatives to binary classification of the image's x and y derivatives. The derivatives of the shading and reflectance images are estimated by labeling each derivative as either shading or a reflectance change.

Once the derivatives of the shading and reflectance images are estimated, they can be used to recover the actual images. Each derivative represents a set of linear constraints on the image and using both derivative images results in an overconstrained system. We recover each intrinsic image from its derivatives with the same method used by Weiss in [23] to find the pseudoinverse of the overconstrained system of derivatives. If f_x and f_y are the filters used to compute the x and y derivatives and $\mathcal{F}_x$ and $\mathcal{F}_y$ are the estimated derivatives of shading image, then the solution for the shading image, $\mathcal{S}(x, y)$ is:

$$\mathcal{S}(x,y) = g * \left[(f_x(-x,-y) * \mathcal{F}_x) + (f_y(-x,-y) * \mathcal{F}_y)\right], \quad (2)$$

where $*$ is convolution, $f(-x,-y)$ is a reversed copy of $f(x,y)$, and g is the solution of

$$g * \left[(f_x(-x,-y) * f_x(x,y)) + (f_y(-x,-y) * f_x(x,y))\right] = \delta. \quad (3)$$

In this work, f_x and f_y are [-1 1] filters.

The reflectance image is found in the same fashion. One nice property of this technique is that the computation can be done using the efficiently FFT.

This model assumes that the final image is created by adding the shading and reflectance images. In reality, the images combine multiplicatively, so our system would ideally operate on the log of the input images. However, for the real-world images examples shown, the results are computed without first computing the log of the input image. This is because ordinary photographic tone-scale is very similar to a log transformation. Errors from not taking log of the input image first would cause one intrinsic image to modulate the local brightness of the other. We found that these brightness artifacts do not occur in the results when processing typical images without the log transformation.

4 Classifying Derivatives

Our system for recovering shading and reflectance images from a single image has three basic steps:

1. Compute derivatives of the input image.
2. Classify each derivative as being caused by either shading or a reflectance change.
3. Invert derivatives classified as shading to find shading images. The reflectance image is found in a similar fashion.

The most difficult of these steps is correctly classifying the derivatives. Classification is accomplished using two forms of local evidence to classify the image derivatives: color and intensity patterns. Section 4.1 describes how color information is used to classify image derivatives. Next, Section 4.2 describes how the statistical regularities of surfaces can be used to classify derivatives from gray-scale patterns. Section 4.3 shows how these two cues can be combined.

4.1 Using Color Information

The color classifier takes advantage of the property that a change in color between pixels often indicates a reflectance change [19]. When surfaces are diffuse and all of the illumination has the same color, any changes in a color image due to shading should affect all three color channels proportionally. Denote two adjacent pixels in the image as $\mathbf{c}_1$ and $\mathbf{c}_2$, where $\mathbf{c}_1$ and $\mathbf{c}_2$ are RGB triplets. If the change between the two pixels is caused by shading, then only the intensity of the pixels' color changes and $\mathbf{c}_2 = \alpha\mathbf{c}_1$ for some scalar α. If $\mathbf{c}_2 \neq \alpha\mathbf{c}_1$, the chromaticity of the colors has changed and the color change must have been caused by a

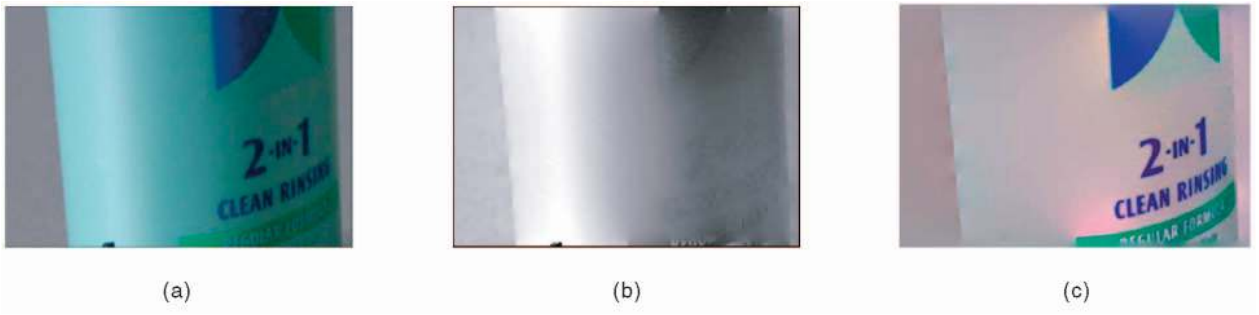

Fig. 4. The shading and reflectance images computed using (a) as input. Only color information is used to classify derivatives. To facilitate printing, the intrinsic images have been computed from a gray-scale version of the image. The color information is used solely for classifying derivatives in the gray-scale copy of the image. (a) Original image. (b) Shading image. (c) Reflectance image.

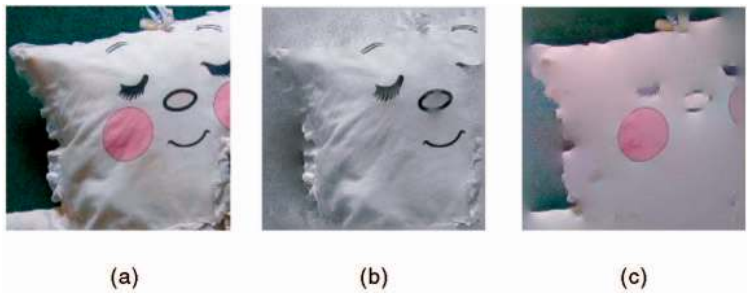

Fig. 5. Intrinsic images created using only color image information to classify the derivatives. The pink cheek patches are correctly placed in the reflectance image. However, the black face is incorrectly placed in the shading image because areas where only the intensity of the color changes, not the chromaticity, are ambiguous based on color alone. (a) Input image. (b) Shading image. (c) Reflectance image.

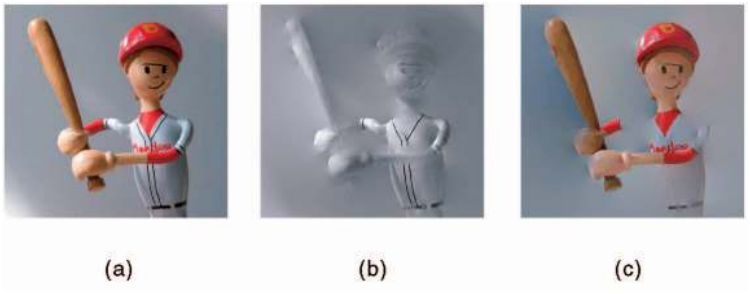

Fig. 6. Intrinsic images created using only color image information to classify the derivatives. This image demonstrates how non-Lambertian surfaces are treated by the color classifier. The specularities on the hat of the figure are placed in the reflectance image. (a) Input image. (b) Shading image. (c) Reflectance image.

reflectance change. Thus, a chromaticity change in the image indicates that the reflectance must have changed at that point.

To find chromaticity changes, we treat each RGB triplet as a vector and normalize them to create $\hat{\mathbf{c}}_1$ and $\hat{\mathbf{c}}_2$. We then use the angle between $\hat{\mathbf{c}}_1$ and $\hat{\mathbf{c}}_2$ to find local evidence for reflectance changes. Ideally, when the change is caused by shading, $(\hat{\mathbf{c}}_1 \cdot \hat{\mathbf{c}}_2)$ should equal 1. In practice, if $(\hat{\mathbf{c}}_1 \cdot \hat{\mathbf{c}}_2)$ is below a threshold, then the derivative associated with the two colors is classified as a reflectance derivative. In our implementation, this threshold, which was set to cos(0.01), was chosen manually. Using only the color information, this approach is similar to that used by Funt et al. [9]. The primary difference is that our system classifies the vertical and horizontal derivatives independently.

Images previously compressed with the JPEG standard frequently contain noticeable chromaticity artifacts. If the image has been processed with the JPEG standard, we alleviate these artifacts by first smoothing the images with a 5×5 Gaussian filter.

Fig. 4 shows an example of the results produced by the algorithm on an image of a shampoo bottle. The classifier marked all of the reflectance areas correctly and the text is cleanly removed from the bottle. This example also demonstrates the high quality reconstructions that can be obtained by binary classification of image derivatives.

Fig. 5 shows an example where color information alone is insufficient. The mouth and eyes of the pillow are formed by a black-to-white intensity change. When only the color classifier is used, intensity changes must be classified as shading since shading can only cause intensity changes. However, an intensity change, such as the eyes and mouth on the pillow, could also be a change in reflectance. This causes the eyes in Fig. 5 to be incorrectly classified and appear in the shading image shown in Fig. 5b.

The toy shown in Fig. 6 also demonstrates the same ambiguity. The black line is incorrectly placed in the shading image. This image also shows the effect of a specular, non-Lambertian surface on the color classifier. The specularities on the hat of the toy figure are placed in the reflectance image.

4.2 Using Gray-Scale Information

The ambiguity inherent in color data can be reduced by examining the structure of patterns in the image. The regular properties of surfaces and illumination in the real world give shading patterns a unique appearance that can be discriminated from most common reflectance patterns. The ripples on the surface of the pillow in Fig. 5b have a much different appearance than the face pattern that has been painted on the pillow. This regular appearance of shading patterns should allow us to use the local gray-scale image pattern surrounding a derivative to classify that derivative.

The basic feature of the gray-scale classifier is the absolute value of the response of a linear filter. We refer to a feature computed in this manner as a *nonlinear filter*. The output of a nonlinear filter, F, given an input patch I_p is

$$F(I_p) = |I_p * w|, \tag{4}$$

where $*$ is convolution and w is a linear filter. The filter, w is the same size as the image patch, I, and we only consider the response at the center of I_p. This feature could also be viewed as the absolute value of the dot product of I_p and w. We use the responses of linear filters as the basis for our feature, in part, because they have been used successfully for characterizing [15] and synthesizing [11] images of textured surfaces.

Before choosing to use these nonlinear filters as features, we also evaluated the features used by Bell and Freeman to classify wavelet coefficients [2]. Bell and Freeman used cascaded, nonlinear filters as the basic feature for their classifier, similar to the cascaded filters used by Tieu and Viola [22]. A cascaded, nonlinear filter consists of a nonlinear filter, similar to (4), with a second filtering and absolute value operation added. In addition, the output of the first filtering stage is down-sampled before it is filtered again. After evaluating both classifiers, we found that using a cascaded, nonlinear filter, rather than just a nonlinear filter, did not improve the performance of the classifier. In addition, taking the absolute value between filtering steps makes it more difficult to interpret the features chosen; leading us to only use a single filtering step followed by an absolute value operation. We also found that removing the down-sampling step did not affect the classifier's performance on the training set.

4.2.1 Learning a Classifier

Training the classifier actually involves two tasks: 1) choosing the set of nonlinear filters to use as features for the classifier and 2) training a classifier based on those features.

Similar to [22], we accomplish these two tasks simultaneously using the AdaBoost algorithm [7]. Given a set of weak classifiers, the AdaBoost algorithm is used to combine those weak classifiers into a single strong classifier. The output of the strong classifier, given an input sample, is a weighted combination of the classifications produces by applying each of the weak classifiers to the input sample.

To train the system, all derivatives in the training set caused by shading are assigned the label 1 and derivatives caused by a reflectance change are labeled -1. A weak classifier, $h(I_p)$, where I_p is an image patch surrounding the derivative to be classified, consists of a single non-linear filter and a threshold. An example is classified by comparing the response of the filter to a threshold:

$$h(I_p) = \gamma \cdot h'(I_p), \tag{5}$$

where $\gamma \in \{-1, 1\}$ and

$$h'(I_p) = \begin{cases} -1 & \text{if } F(I_p) < T \\ 1 & \text{Otherwise,} \end{cases} \tag{6}$$

where T is some threshold. As described above, $F(\cdot)$ is a nonlinear filter defined by some linear filter w.

The AdaBoost algorithm is used to choose both the weak classifiers and the weight that should be assigned to each of them. The first step in the algorithm is to choose a weak classifier. For the classifier described in (5), this involves choosing a linear filter w, γ, and a threshold T. The filter w is chosen from a set of 17×17 pixel derivative filters. To maximize the number of filters evaluated as possible weak classifiers, each candidate filter was formed from the combination of two smaller filters taken from a set of 9×9 filters. This set consists of nine derivative of Gaussian filters and nine second-derivative of Gaussian filters oriented every 22.5 degrees between 0 and 180 degrees. In addition, the set included an impulse filter and four Gaussian filters with different widths. The actual candidate set of filters that the classifiers are chosen from consists of every possible combination of the smaller filters, with the requirement that each combination contain at least one derivative filter.

The candidate weak classifiers are evaluated against a weighted training set. Throughout the training process, the AdaBoost algorithm maintains a distribution, D_t, on the training samples. $D_t(i)$ can also be thought of as the weight of training example i. These weights are updated depending on the performance of the weak classifier. When selecting a classifier, AdaBoost only requires that the selected weak classifier classify the reweighted training set with better than 50 percent accuracy. For iteration t, the goal is to find a classifier h_t such that

$$\Pr_{i \sim D_t}[h_t(x_i) = y_i] > 0.5, \tag{7}$$

where x_i is training example i and y_i is its true label. The notation $i \sim D_t$ denotes that i is chosen according to the probability distribution D_t. Similar to [22], we use this step to perform feature selection. At each iteration, the next classifier chosen is the classifier that has the highest probability of being correct on the weighted training set. Because each weak classifier consists of a single nonlinear filter and a threshold, greedily choosing the weak classifier is equivalent to greedily choosing the best filters.

Once the weak classifier has been chosen, the next step is to calculate the weight to assign to the weak classifier. This weight, α_t, is

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right), \tag{8}$$

where ϵ_t is the probability of error, $\Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$, when the training examples are drawn with probability distribution D_t. This essentially weights the errors according to D_t.

Once α_t has been computed, the weight for each training example in the next iteration, D_{t+1} is computed as

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}, \tag{9}$$

where Z_t is a normalization factor needed to make D_t a valid distribution. In this step, training examples that were incorrectly classified will receive more weight, while those correctly classified by the weak classifier will receive less weight.

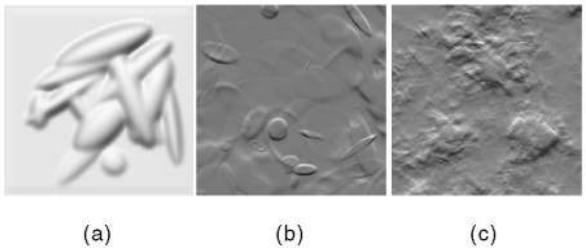

Fig. 7. Examples of shading from the training set. (a) Random ellipses rendered with Lambertian shading. (b) Random ellipses rendered with linear shading. (c) Fractal surface rendered with linear shading.

Once the desired number of iterations has run, the output of the strong classifier on a new sample is computed by having each of the weak classifiers independently classify the sample. Given a sample, x, each weak classifier casts a vote to classify the sample as -1 or 1. The final classification output of the combined classifier, $H(x)$ is the weighted average of the votes

$$H(x) = \text{sign}\left(\sum_{t=1} N\alpha_t h_t(x)\right), \tag{10}$$

where N is the number of weak classifiers used.

4.2.2 Generating a Training Set

The training set for learning the gray-scale classifier is taken from synthetic images. In order to generate a training set that captures the statistics of shading, the examples of shading are created with three methods. Approximately 63 percent of the set is generated by creating random surfaces with ellipsoids, then rendering them with Lambertian shading. The surfaces are created by randomly placing ellipsoids throughout the image. At points where ellipsoids overlap, the maximum of the height of every ellipsoid at that point is used. Before being rendered, each surface is smoothed with a Gaussian filter. An example image is shown in Fig. 7a. The remaining 37 percent of the examples of shading was taken from the set created by Bell and Freeman for their work on producing intrinsic images [2]. Half of the Bell and Freeman training set was also created with random surfaces created with ellipses, except the surfaces are rendered using linear shading, an approximation to the true Lambertian shading when the angle of the illumination is oblique to the surface [18]. An example of this part of the training set is shown in Fig. 7b. The rest of the Bell and Freeman training set comes from rendered fractal surfaces, such as Fig. 7c. Each example image is created by using the midpoint displacement method to create a surface, then rendering it using linear shading.

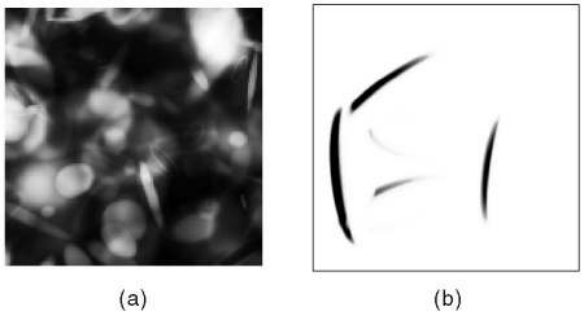

Fig. 8. Examples of reflectance changes from the training set. (a) Random ellipses rendered as reflectance changes. (b) Random lines.

Every shading image was lit from the same direction. In the training set, the illumination comes from the top of the image. We assume that for every input image, the direction of illumination is roughly known. Before classifying an input image, the image is rotated so that the dominant illumination in the image also comes from the top side of the image. The requirement that the illumination come from the top could be removed using a strategy similar to that used in [2]. This strategy finds the dominant direction of the illumination by evaluating multiple rotated copies of the image, then choosing the rotation with the largest number of steerable pyramid coefficients classified as shading.

The examples of reflectance changes were generated in two fashions. Approximately 37 percent of the reflectance examples were generated by randomly placing ellipses in the image, then rendering them as reflectance changes. Fig. 8a shows an example of an image produced in this fashion. The remaining 63 percent of the training set is generated by randomly rendering lines. In order to create images with corners and anti-aliased lines, we use ellipsoid images similar to those used as shading examples. To create the lines, we mask the darkest portions of the ellipsoid image, then set the rest of the image to white. The intensity of each line is then set randomly. An examples of an image of rendered lines is shown in Fig. 8b.

The training set had 2,200 total samples, evenly divided between examples of shading and examples of reflectance changes. While these images are only a small subset of the possible shading or reflectance images, we find that the classifiers trained from these images generalize well in classifying the image derivatives from real images.

4.2.3 Examining the Filters Learned

For the results shown, our system used 10 weak classifiers. The columns of Fig. 9 shows the filters associated with the weak classifiers chosen by the AdaBoost algorithm. Each column

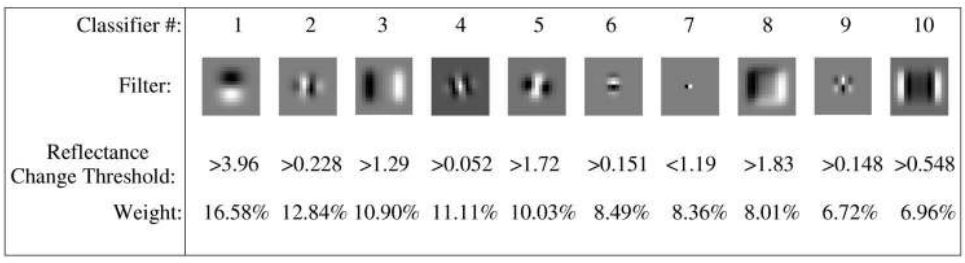

Fig. 9. An example of the filters selected by AdaBoost for classifying vertical derivatives when the illumination is from the top of the image. Each column contains one nonlinear filter. The reflectance change threshold row describes the threshold test for labeling a derivative as a reflectance change.

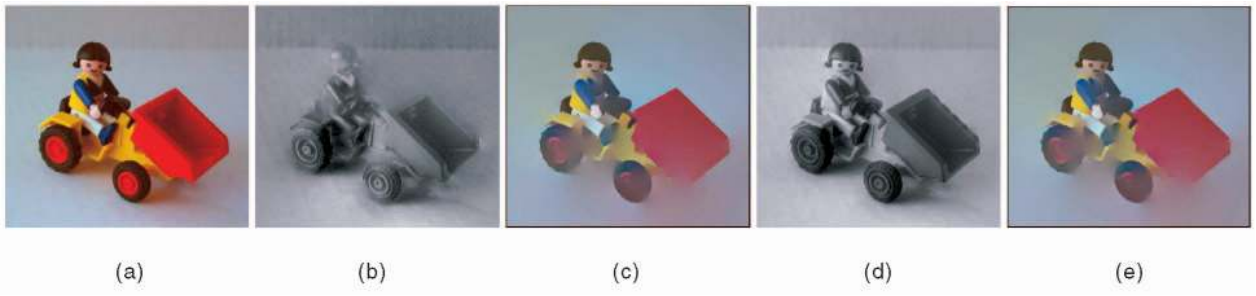

Fig. 10. Intrinsic images created from (a) using only gray-scale image information to classify the derivatives. (b) and (c) show the shading and reflectance images created using our method. The shading image created by our method is shown in (b). The 3D structure of the truck is properly placed in the shading image, which should appear as if the entire object is made of one matte material. The reflectance image is shown in (c). (d) and (e) show the intrinsic images created using the Retinex classifier for comparison. The Retinex classifier produces a good reflectance image, but many reflectance changes that should have been removed from the shading image still remain in (d).

contains the filter for one weak classifier. The filters shown in Fig. 9 are classifiers trained for vertical derivatives when the illumination is from the top of the image. The reflectance change threshold row shows the threshold used when testing whether to label a derivative as a reflectance change. For all but the filter number 7 in Fig. 9, the AdaBoost algorithm has chosen to label responses with a magnitude above the threshold as a reflectance change. These values assume the input intensity image is normalized between 0 and 1.

The percentage of weight allocated to each classifier is shown beneath the corresponding filter in Fig. 9. The votes from weak classifiers with larger percentages influence the final classification more strongly. It is interesting to note that the filter with the greatest weight distinguishes shading and reflectance changes in fashion very similar to Retinex. In Retinex and its derivatives, derivatives with a large magnitude are assumed to be reflectance changes, while small derivatives indicate shading. The filter with the most weight effectively smoothes the image, then classifies derivatives with a large magnitude as reflectance changes. This validates that Retinex is a good heuristic for accomplishing this task.

Another interesting aspect of the AdaBoost classifier is that seven of the 10 filters chosen to classify vertical derivatives are horizontally oriented. This is related to the fact that in the training set, each image is illuminated from the top of the image. The choice of these filters indicates that it is unlikely that edges caused by shading will be oriented perpendicular to the illumination.

4.2.4 Evaluating Classifier Performance

As a baseline for comparison, we also created a classifer based on the basic assumption of Retinex. This assumption is that image derivatives with a large magnitude correspond to reflectance changes, while small derivatives are most likely caused by shading. The Retinex classifer was implemented as a simple threshold test. If ΔI is an image derivative, then ΔI will be classified as a reflectance change if $k_1 \cdot |\Delta I| + k_0 > 0$. Otherwise, ΔI will be classified as shading. The constants k_0 and k_1 were found using logistic regression. This classifier was trained on the same training set as our gray-scale classifier.

To compare the two methods, we used the method described in Section 4.2.2 to draw a new set of 1,000 samples. Our gray-scale classifier correctly classified 89 percent of the test samples correctly, while the Retinex classifier correctly classified 60 percent of the samples.

Because these results were found using synthetic data, it is difficult to use them to predict each classifier's performance on real images. However, Figs. 10, 11, 12, and 13 show empirical evidence that given real images, our classifier will still perform better than a simple Retinex classifier. One reason for this superior performance is that our classifier is able to both take advantage of the same heuristic as the Retinex classifier. As mentioned in Section 4.2.3, the weak classifier with the most weight uses a smoothed derivative to distinguish between shading and reflectance changes. In addition, our gray-scale classifier is able to use information from the additional weak classifiers.

Figs. 10b and 10c show the shading and reflectance images generated by our gray-scale classifier, using Fig. 10a as input. For comparison, the shading and reflectance images created using the simple Retinex classifier are shown in Figs. 10d and 10e. The results can be evaluated by thinking of the shading image as how the scene should appear if it were made entirely of gray plastic. The reflectance image should appear very flat, with the three-dimensional depth cues placed in the shading image. In this example, the system performs well. The shading image, shown in Fig. 10b, has a very uniform appearance. In addition, almost all of the effects of the reflectance changes are correctly placed in the reflectance image, shown in Fig. 10c. The Retinex classifier is able to produce a good reflectance image, but the shading image still contains many reflectance changes that should have been removed.

When applied to the graffiti image in Fig. 11a, the gray-scale classifier successfully separates the paint from the shading of the wall, shown in Fig. 11b. Some of the ridges of the surface are misclassified and appear in the reflectance image shown in Fig. 11c. The most significant cause of these misclassifications is that ridges such as these did not appear in the training set. In addition, a majority of weak classifiers, shown in Fig. 9, have learned to classify edges with a strong contrast as reflectance changes. In this example most of the misclassified edges have a strong contrast. The results from the simple Retinex classifier, shown in Figs. 11d and 11e, again show that it fails to prevent many reflectance edges from appearing in the shading image.

In Fig. 13, our classifier places most of the lines on the jersey in the reflectance image, while some of the lines are incorrectly shown in the shading image. These lines are caused by ambiguities in the local evidence used by

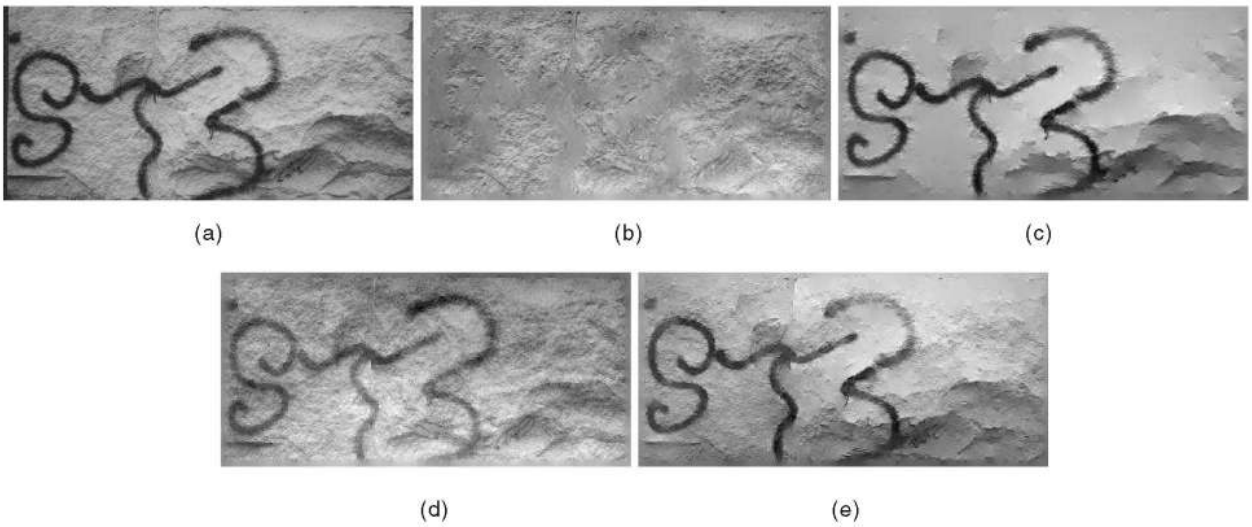

Fig. 11. Intrinsic images created from (a) using only gray-scale image information to classify the derivatives. (b) and (c) show the shading and reflectance images created using our method. The shading image, (b), is missing some edges that have been misclassified as reflectance changes because edges such as these did not appear in our training set. The paint is correctly removed from the surface and placed in the reflectance image, (c). (d) and (e) show the intrinsic images created using the Retinex classifier for comparison. Again, many reflectance changes are incorrectly placed in (d).

Fig. 12. Intrinsic images created from (a) using only gray-scale image information to classify the derivatives. (b) and (c) show the shading and reflectance images created using our method. (d) and (e) show the intrinsic images created using the Retinex classifier for comparison. Again, many reflectance changes are incorrectly placed in (d).

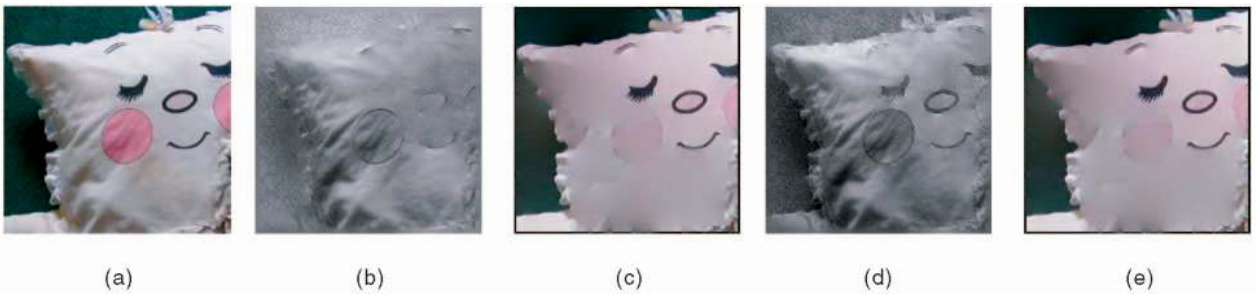

Fig. 13. Intrinsic images created from the pillow image from Fig. 5 using only gray-scale form image information to classify the derivatives. (b) and (c) show the shading and reflectance images created using our method. The weak gradient of the cheek patches leads them to be incorrectly placed in the shading image, (b). The face is correctly placed in the reflectance image, (c). (d) and (e) show the intrinsic images created using the Retinex classifier for comparison. Again, many reflectance changes are incorrectly placed in (d).

classifiers. The reason for this type of error and our method for fixing these types of errors are discussed in Section 5.

The intensity classifiers also fix many of the errors in the decomposition of the pillow example generated using the color-only algorithm of Section 4.1. In Fig. 13, our form-based classifier correctly identifies the eyes and most of the mouth as reflectance changes. It also correctly identifies the right and left edges of the cheek patch. The most significant error made by our classifier is that the top and bottom edges of the check patches are incorrectly classified as shading. This is partly due to the fact that the top and bottom edges of the cheek are locally ambiguous. They are edges that could have been produced by either shading or a reflectance change.

4.2.5 *Classifier Limitations*

The pillow shown in Fig. 13 shows one of the most significant limitations of both our gray-scale classifier and the Retinex classifier. Besides local ambiguity, the other main cause of the misclassification of the derivatives around the cheek patches is low contrast edges. The borders of the cheeks are created by low contrast edges. Both our classifier and the Retinex classifier base their predictions on the response of a linear

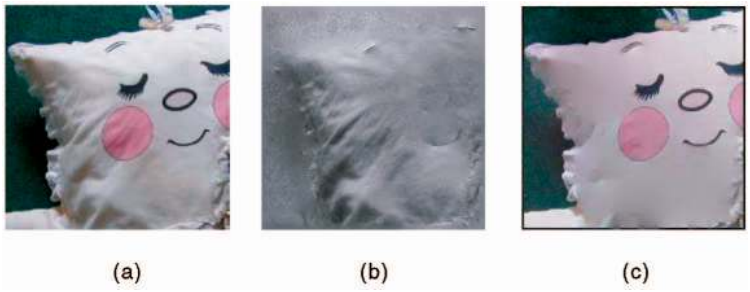

Fig. 14. Intrinsic images created from the pillow image using both color and gray-scale image information to classify the derivatives. Almost the entire image is classified correctly. Portions of the mouth remain in the shading image because those areas are locally ambiguous. (a) Input image. (b) Shading image. (c) Reflectance image.

filter. This biases the classifier toward classifying weak edges as shading.

The gray-scale classifier is also trained to only distinguish between reflectance changes and shading. Its behavior on other common visual events, such as occluding edges, shadows, and specularities is undefined. Typically, derivatives caused by these events are classified as reflectance changes by the classifier.

4.3 Combining Color and Gray-Scale Information

Neither the color nor the gray-scale classifier is able to classify the entire pillow image correctly. However, by combining the results of the two classifiers, the quality of the classifications can be enhanced significantly.

Color and gray-scale information can be used simultaneously by requiring the classifiers to assign probabilities to the classification of each derivative. To combine the results of the color and gray-scale classifiers, let D be the classification of some derivative. We denote D_s as the event that the derivative should be labeled shading and denote D_r as the event that the derivative should be labeled a reflectance change. From the color classifier, we obtain $\Pr[D_s|C]$ and $\Pr[D_r|C]$, the probabilities of the derivative being caused by shading or a reflectance change, given the local color information C. The gray-scale classifier returns $\Pr[D_s|G]$ and $\Pr[D_r|G]$, the probabilities of the derivative's classification, given the local gray-scale information G.

We assume that the outputs of the color and gray-scale classifiers are statistically independent variables. Using Bayes' rule, this enables the probability of the derivative being caused by shading, $\Pr[D_s|G,C]$, to be expressed as

$$\Pr[D_s|G,C] = \frac{\Pr[D_s|G]\Pr[D_s|C]}{(\Pr[D_s|G]\Pr[D_s|C] + \Pr[D_r|G]\Pr[D_r|C])}. \tag{11}$$

The probability that the derivative is caused by a reflectance change is found by

$$\Pr[D_r|G,C] = \frac{\Pr[D_r|G]\Pr[D_r|C]}{(\Pr[D_s|G]\Pr[D_s|C] + \Pr[D_r|G]\Pr[D_r|C])}. \tag{12}$$

To obtain $\Pr[D_s|G]$, the probability of a derivative being caused by shading from the gray-scale classifier, we used the method suggested by Friedman et al. to transform the output of an AdaBoost classifier into the probability of each label [8]. Given a sample, x, the AdaBoost classifier, $H(x)$, returns a value between -1 and 1. The probability that the true label of x, denoted as y, has the value 1 is well approximated by

$$\Pr[y=1] \approx \frac{e^{H(x)}}{e^{H(x)} + e^{-H(x)}}. \tag{13}$$

In our system, the AdaBoost classifier uses $G(i,j)$, a patch of local image information around location (i,j), to classify the derivative at location (i,j). Using this approximation, the probability that the derivative is cause by shading is

$$\Pr[D_s|G(i,j)] = \frac{e^{H(G(i,j))}}{e^{H(G(i,j))} + e^{-H(G(i,j))}}. \tag{14}$$

The probabilities of each label, given the color information, are obtained in a different fashion because the color-based classifier is not an AdaBoost classifier. The output of the color classifier is transformed into a probability by setting $\Pr[D_s|C(i,j)]$, the probability that a derivative is caused by shading, given the color information, to be some constant probability wherever the color classifier finds a reflectance change. For the results shown, we use the value 0.1 for the constant. The probabilities associated with chromaticity information are assigned this way because without calibrated color images, the measurements of chromaticity changes may be flawed. Using a threshold test for the classifier and setting the probabilities of the classifications in this manner make our algorithm robust to situations where our assumptions about chromaticity changes may not hold.

As explained in Section 4.2, intensity changes are ambiguous so the probability that derivative should be labeled shading is set to 0.5 at locations where the color classifier does not indicate a reflectance change.

Fig. 14 shows the results on the pillow image when both color and gray-scale information are used. Using the two cues, the cheeks and eyes that are painted on the pillow are now correctly removed. However, there are some portions of the mouth that still appear in the shading image. In these areas, the local evidence available to the classifiers is not sufficient. These errors can be resolved by propagating the local evidence spatially.

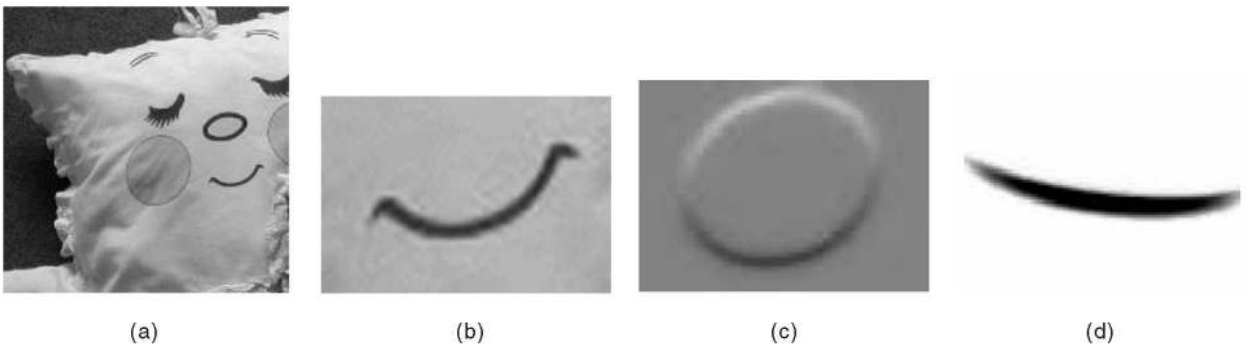

Fig. 15. An example of where propagation is needed. The smile from the pillow image in (a) has been enlarged in (b). (c) and (d) contain an example of shading and a reflectance change, respectively. Locally, the center of the mouth in (b) is as similar to the shading example in (c) as it is to the example reflectance change in (d).

5 Propagating Evidence

While the combined color and gray-scale classifier works well, there are still areas in the image where the local information is ambiguous. An example of this is shown in Fig. 15. When compared to the example shading and reflectance change in Figs. 15c and 15d, the center of the mouth in Fig. 15b is equally well classified with either label. However, the corners of the mouth can be classified as being caused by a reflectance change with little ambiguity. Since the derivatives in the corner of the mouth and the center all lie on the same image contour, it seems natural they should have the same classification. A mechanism is needed to propagate information from the corners of the mouth, where the classification is clear, into areas where the local evidence is ambiguous. This will allow areas where the classification is clear to disambiguate those areas where it is not clear. The propagation step is based on the heuristic that derivatives corresponding to the same image contour should have the same labels.

In order to propagate evidence, we treat each derivative as a node in a Markov Random Field, or MRF, with two possible states, indicating whether the derivative is caused by shading or caused by a reflectance change. Setting the compatibility functions between nodes correctly will force nodes along the same contour to have the same classification. Separate MRF's are used for the horizontal and vertical derivatives.

5.1 Model for the Potential Functions

Each node in the MRF corresponds to the classification of a derivative. The compatibility functions for two neighboring nodes in the MRF, x_i and x_j, have the form

$$\psi(x_i, x_j) = \begin{bmatrix} \beta & 1-\beta \\ 1-\beta & \beta \end{bmatrix} \tag{15}$$

with $0 \leq \beta \leq 1$.

The term β controls how much the two nodes should influence each other. An MRF such as this, where the compatibility functions depend on the observed image information, is often referred to as a Conditional Random Field [13]. The derivatives along an image contour should have the same classification, so β should be close to 1 when two neighboring derivatives are along a contour and should be 0.5 when no contour is present.

Since β depends on the image at each point, we denote it as $\beta(\mathcal{I}_{xy})$, where $\mathcal{I}_{xy}$ is the image information at some point. To ensure $\beta(\mathcal{I}_{xy})$ between 0 and 1, it is modeled as $\beta(\mathcal{I}_{xy}) = g(z(\mathcal{I}_{xy}))$, where $g(\cdot)$ is the logistic function and $z(\mathcal{I}_{xy})$ is a function that should have a large response when two adjacent nodes should have the classification.

5.2 Learning the Potential Functions

The function $z(\mathcal{I}_{xy})$ is based on two local image features, the magnitude of the image and the difference in orientation between the gradient and the orientation of the graph edge. These features reflect our heuristic that derivatives along an image contour should have the same classification. For simplicity, we constrain $z(\cdot)$ to be a linear function of the form:

$$z(\hat{\phi}, |\nabla\mathcal{I}|) = a \cdot \hat{\phi} + b \cdot |\nabla\mathcal{I}| + c, \tag{16}$$

where $|\nabla\mathcal{I}|$ is the magnitude of the image gradient and the constants a, b, and c are to be found by minimizing (17). Both $\hat{\phi}$ and $|\nabla\mathcal{I}|$ are normalized to be between 0 and 1.

The difference in orientation between a horizontal graph edge and image contour, $\hat{\phi}$, is found by constraining the image gradient, ϕ, to be in the range $-\pi/2 \leq \phi \leq \pi/2$, making $\hat{\phi} = |\phi|$. To assign the compatibilities for vertical graph edges, $\hat{\phi} = |\phi| - \pi/2$.

To find the values of $z(\cdot)$, we maximize the probability of a set of the training examples over the parameters of $z(\cdot)$. The examples are taken from the same set used to train the gray-scale classifiers. The probability of training samples is

$$P = \frac{1}{Z}\prod_{(i,j)} \psi(x_i, x_j), \tag{17}$$

where all (i, j) are the indices of neighboring nodes in the MRF and Z is a normalization constant. Note that each $\psi(\cdot)$ is a function of $z(\mathcal{I}_{xy})$.

The constants, a, b, and c, are found by maximizing (17) over a set of training images similar to those used to train the local classifier. In order to simplify the training process, we train using a pseudolikelihood to approximate the true probability of each MRF [3]. The pseudolikelihood is formed

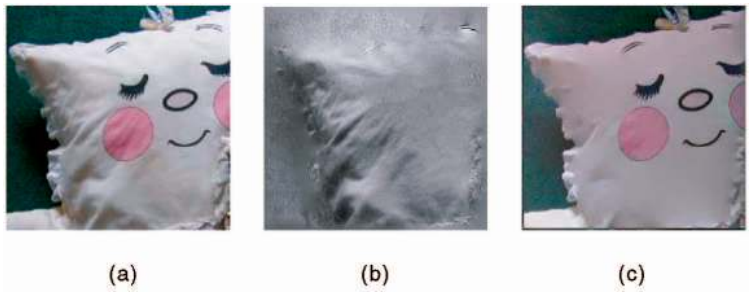

Fig. 16. The pillow from Fig. 15. This is found by combining the local evidence from the color and gray-scale classifiers, then using Generalized Belief Propagation to propagate local evidence. The propagation step has placed the mouth totally in the reflectance image. (a) Original image. (b) Shading image. (c) Reflectance image.

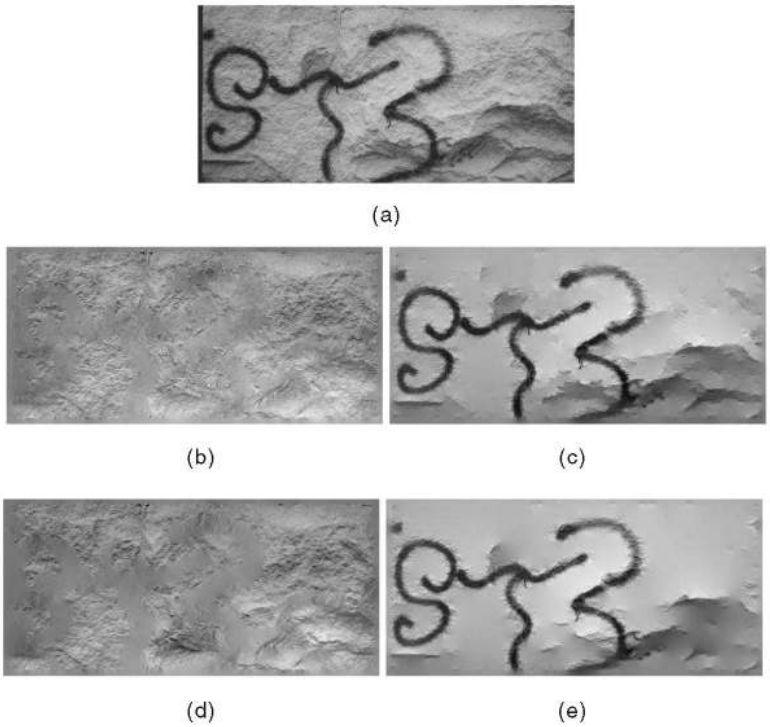

Fig. 17. Intrinsic images created with the addition of the propagation step. Adding the propagation step reduces the number of edges in the surface that are misclassified as reflectance changes. (a) Original image. (b) Shading image without the propagation step. (c) Reflectance image without the propagation step. (d) Shading image with the propagation step. (e) Reflectance image with the propagation step.

by assuming that Z is constant. Doing so, leads to the following values for a, b, and c: $a = -1.2$, $b = 1.62$, $c = 2.3$. These measures break down in areas with a weak gradient, so we set $z(\cdot)$ to 0 for regions of the image with a gradient magnitude less than 0.05. Combined with the values learned for $z(\cdot)$, this effectively limits β to the range $0.5 \leq \beta \leq 1$.

Larger values of $z(\cdot)$ correspond to a belief that the derivatives connected by the edge should have the same value, while negative values signify that the derivatives should have a different value. The values found in (16) lead to our desired result; two derivatives are constrained to have the same value when they are along an edge in the image that has a similar orientation to the edge in the MRF connecting the two nodes.

5.3 Inferring the Correct Labeling

We used the Generalized Belief Propagation algorithm [24] to infer the best label of each node in the MRF because ordinary Belief Propagation performed poorly in areas with both weak local evidence and strong compatibility constraints.

The addition of the propagation step improves the results on the pillow image, shown in Fig. 16 and removes the stray mouth markings seen in Fig. 14, where no propagation was used. In Fig. 16, the ripples on the pillow are correctly identified as being caused by shading, while the face is correctly identified as having been painted on.

The propagation step also cleans up many of edges in the graffiti image, shown in Fig. 17, which have been misclassified as reflectance changes by the gray-scale classifier.

Fig. 18 shows how the propagation step can improve image quality for ambiguous contours in an image. The local ambiguity of the stripes on the chest of the toy lead to both blotchy image artifacts in Fig. 18d and a stripe incorrectly being placed in the shading image. After propagation, the lines are removed from the shading image, shown in Fig. 18c.

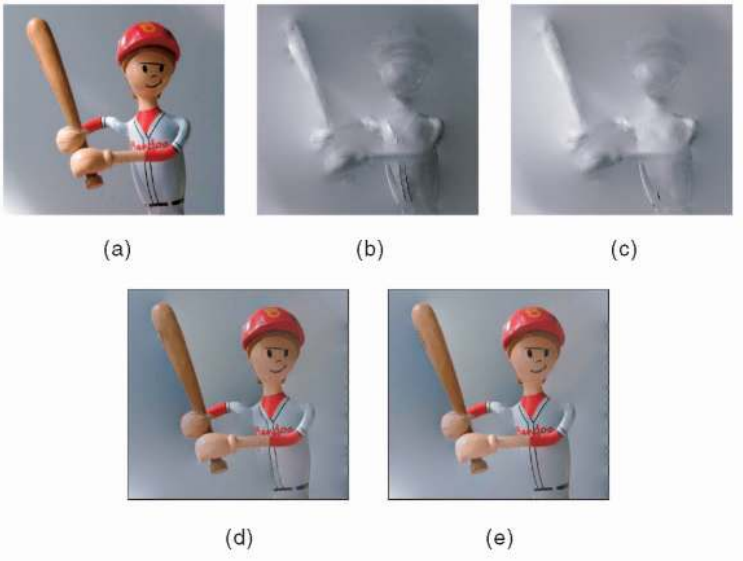

Fig. 18. An example of how propagation can improve image quality. Before propagation, the lines on the jersey appear in both the shading and reflectance images. After propagation, the lines are solely in the reflectance image. (a) Input image. (b) Shading image without propagation. (c) Shading image with propagation. (d) Reflectance image without propagation. (e) Reflectance image with propagation.

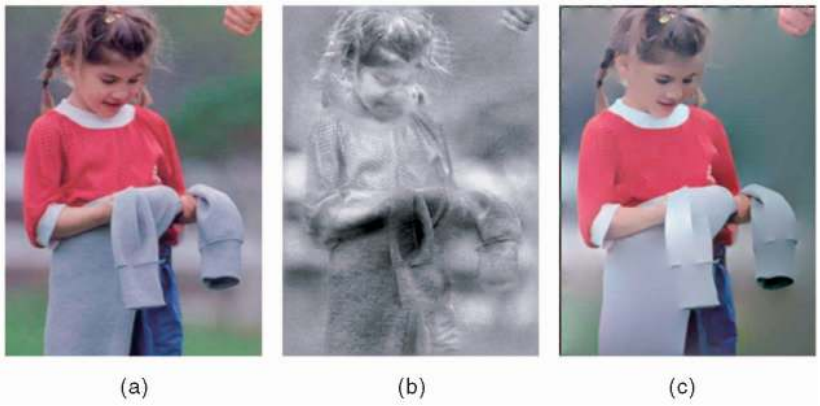

Fig. 19. Example generated by combining color and gray-scale information, along with using propagation. (a) Original image. (b) Shading image. (c) Reflectance image.

The improvement provided by the propagation step depends on the performance of the local color and gray-scale classifiers. If the classifiers are overconfident in a derivative's label, then the propagation step will not improve the results.

6 Additional Examples

In this section, we present additional examples of how our system performs on a variety of images.

In another real-world example, shown in Fig. 19, the algorithm correctly identifies the change in reflectance between the sweatshirt and the jersey and correctly identifies the folds in the clothing as being caused by shading. There are some small shading artifacts in the reflectance image, especially around the sleeves of the sweatshirt, caused by inadequate generalization to shapes not present in the training set.

In Fig. 20a, we have recreated a demonstration in Gibson's classic 1966 vision book [10] that describes how both shading and reflectance changes can create structure in the image. The results of applying our algorithm to this image is shown in Figs. 20b and 20c.

In the next example, shown in Fig. 21, we applied our system to the image used in Section 1 to introduce the concept of intrinsic images. Overall, the image is decomposed correctly. The most noticeable errors in the image are around the occluding contours of the lumps in the surface. It is reasonable that our system misclassifies these lumps because our training set does not include occluding contours.

In Fig. 22, we apply our system to a unique real-world image. In Fig. 22a, the embossing of the sign is rotated with respect to how the sign has been painted. The sign has a shiny, metallic surface that is not modeled well by the color classifier, so these results are produced using the gray-scale classifier and the propagation step. The system performs well, in this case, the embossing is correctly placed in the shading image, while the painting is correctly placed in the reflectance image.

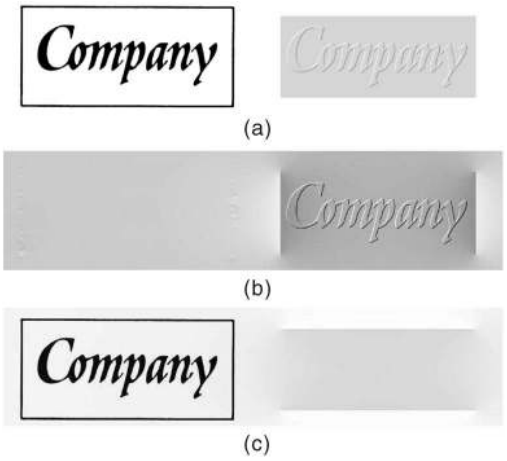

Fig. 20. The results from decomposing a recreation of a demonstration created by Gibson to show shading and reflectance changes [10] create structure in images. (a) Original image. (b) Shading image. (c) Reflectance image.

7 Conclusion

We have presented a system that is able to estimate shading and reflectance intrinsic images from a single real image, given the direction of the dominant illumination of the scene. Although some properties of real-world scenes are not modeled directly, such as occlusion edges, the system produces satisfying image decompositions. The basic strategy of our system is to gather local evidence from color and intensity patterns in the image. This evidence is then propagated to other areas of the image. This same strategy can be applied to other vision problems.

The most computationally intense steps for recovering the shading and reflectance images are computing the local evidence, which takes about six minutes on a 700MHz Pentium 3 for a 256×256 image, and running the Generalized Belief Propagation algorithm. Belief propagation was used on both the x and y derivative images and took around 6 minutes to run 200 iterations on each 256×256 image. The pseudoinverse process took under 5 seconds.

One of the primary limitations of this work was the use of synthetic training data. This limited both the performance of the system and the range of algorithms available for designing the classifiers. We expect that performance would be improved by training from a set of intrinsic images gathered from real data. In addition, a set of labeled training examples would enable different types of derivative classifiers to rigorously compared against each other.

Acknowledgments

Portions of this work were completed while William T. Freeman was a senior research scientist and Marshall F. Tappen was a summer intern at Mitsubishi Electric Research Labs. This work was supported by an NDSEG fellowship to Mr. Tappen, a NIH Grant EY11005-04 to Dr. Adelson, a grant from NTT to Dr. Adelson, an ONR/MURI contract N00014-01-0625 to Dr. Adelson, and by a contract with Unilever Research.

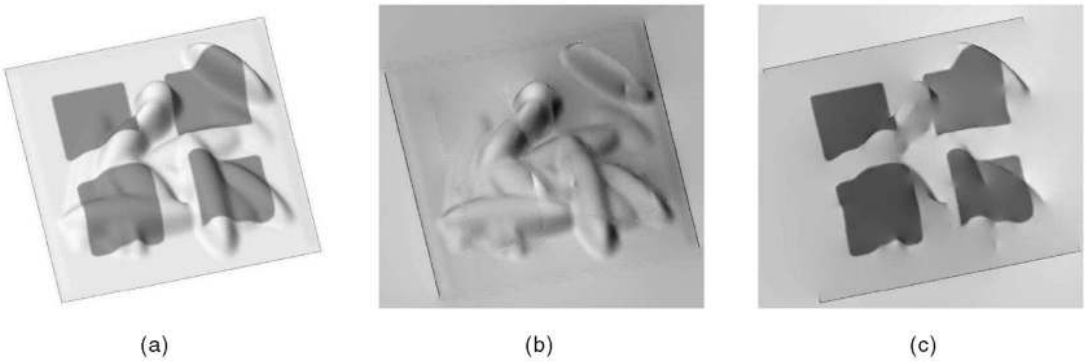

Fig. 21. The results of our system applied to the image used in Section 1 to describe intrinsic images. The most noticeable errors are along the occluding contours of the surface. This is understandable because no occluding contours are included in the training set. (a) Original image. (b) Shading image. (c) Reflectance image.

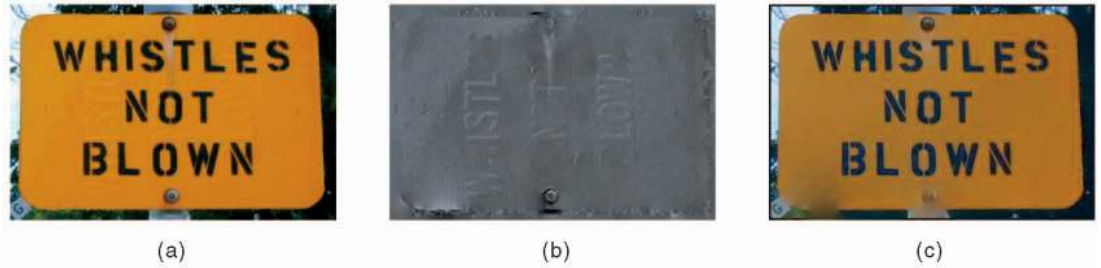

Fig. 22. Our system applied to a unique real-world image. In this image, the embossing of the sign does not correspond to how the sign has been painted. The shiny surface of the sign is not Lambertian, so we did not use the color classifier for this image. The embossing is correctly placed in the shading image, while the painting is correctly placed in the reflectance image. (a) Original image. (b) Shading image. (c) Reflectance image.

REFERENCES

[1] H.G. Barrow and J.M. Tenenbaum, "Recovering Intrinsic Scene Characteristics from Images," *Computer Vision Systems,* A. Hanson and E. Riseman, eds., pp. 3-26. Academic Press, 1978.

[2] M. Bell and W.T. Freeman, " Learning Local Evidence for Shading and Reflection," *Proc. Int'l Conf. Computer Vision,* 2001.

[3] J. Besag, "Statistical Analysis of Non-Lattice Data," *The Statistician,* vol. 24, no. 3, pp. 179-195, Sept. 1975.

[4] G.D. Finlayson, S.D. Hordley, and M.S. Drew, "Removing Shadows from Images," *Proc. European Conf. Computer Vision,* pp. 823-836. 2002.

[5] W.T. Freeman, E.C. Pasztor, and O.T. Carmichael, "Learning Low-Level Vision," *Int'l J. Computer Vision,* vol. 40, no. 1, pp. 25-47, 2000.

[6] W.T. Freeman and P.A. Viola, "Bayesian Model of Surface Perception," *Advances in Neural Information Processing Systems,* Apr. 1998.

[7] Y. Freund and R.E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," *J. Computer and System Sciences,* vol. 55, no. 1, pp. 119-139, 1997.

[8] J. Friedman, T. Hastie, and R. Tibshirami, "Additive Logistic Regression: A Statistical View of Boosting," *The Annals of Statistics,* vol. 38, no. 2, pp. 337-374, 2000.

[9] B.V. Funt, M.S. Drew, and M. Brockington, "Recovering shading from Color Images," *Proc. Second European Conf. Computer Vision,* G. Sandini, ed., pp. 124-132, May 1992.

[10] J.J. Gibson, *The Senses Considered as Perceptual Systems,* chapter 10. Houghton Mifflin, 1966.

[11] D. Heeger and J. Bergen, "Pyramid-Based Texture Analysis/Synthesis," *Proc. SIGGRAPH, Computer Graphics,* pp 229-238, Aug. 1995.

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

[13] J. Lafferty, F. Pereira, and A. McCallum, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," *Proc. Int'l Conf. Machine Learning (ICML),* 2001.

[14] E.H. Land and J.J. McCann, "Lightness and Retinex Theory," *J. Optical Soc. Am.,* vol. 61, pp. 1-11, 1971.

[15] T. Leung and J. Malik, "Recognizing Surfaces Using Three-Dimensional Textons," *Proc. IEEE Int'l Conf. Computer Vision,* 1999.

[16] Y. Matsushita, K. Nishino, K. Ikeuchi, and M. Sakauchi, "Illumination Normalization with Time-Dependent Intrinsic Images for Video Surveillance," *Proc. 2003 Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 3-10, 2003.

[17] A.V. Oppenheim, R.W. Shafer, and J. Thomas, G. Stockham, "Nonlinear Filtering of Multiplied and Convolved Signals," *IEEE Trans. Audio and Electroacoustics,* vol. 16, no. 3, pp. 437-466, Sept. 1968.

[18] A.P. Pentland, "Linear Shape from Shading," *Int'l J. Computer Vision,* vol. 1, no. 4, pp. 153-162, 1990.

[19] J.M. Rubin and W.A. Richards, "Color Vision and Image Intensities: When Are Changes Material," *Biological Cybernetics,* vol. 45, pp. 215-226, 1982.

[20] E.P. Simoncelli and W.T. Freeman, "The Steerable Pyramid: A Flexible Architecture for Multi-Scale Derivative Computation," *Proc. Int'l Conf. Image Processing,* vol. 3, pp. 444-447, Oct. 1995

[21] P. Sinha and E.H. Adelson, "Recovering Reflectance in a World of Painted Polyhedra," *Proc. Fourth Int'l Conf. Computer Vision,* pp. 156-163, 1993.

[22] K. Tieu and P. Viola, "Boosting Image Retrieval," *Proc. IEEE Computer Vision and Pattern Recognition,* vol. 1, pp. 228-235, 2000.

[23] Y. Weiss, "Deriving Intrinsic Images from Image Sequences," *Proc. Int'l Conf. Computer Vision,* 2001.

[24] J. Yedidia, W.T. Freeman, and Y. Weiss, "Generalized belief propagation," *Advances in Neural Information Processing Systems 13,* T. K. Leen, H.G. Diettrich, and V. Tresp, eds., pp. 689-695. 2001.

Marshall F. Tappen received the BS degree in computer science from Brigham Young University in 2000 and received the SM degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT) in 2002. He is a PhD candidate in the MIT Computer Science and Artificial Intelligence Laboratory. He is the recipient of the National Defense Science and Engineering Graduate (NDSEG) Fellowship. His interests are in computer vision and machine learning.

William T. Freeman received the BS degree in physics and the MS degree in electrical engineering from Stanford in 1979, and the MS degree in applied physics from Cornell in 1981. He studied computer vision for the PhD degree in 1992 from the Massachusetts Institute of Technology (MIT). He is an associate professor of electrical engineering and computer science at MIT, having joined the faculty in September, 2001. From 1992-2001, he worked at Mitsubishi Electric Research Labs (MERL), in Cambridge, Massachusetts, most recently as senior research scientist and associate director. His current research focus is machine learning applied to computer vision and computer graphics, in particular, object recognition, super-resolution, and spatial inference using belief propagation. Previous research topics include steerable filters and pyramids, the generic viewpoint assumption, analyzing style and content, and computer vision for computer games. He holds 24 patents. From 1981-1987, he worked at Polaroid, developing image processing algorithms for printers and cameras. In 1987-1988, he was a foreign expert at the Taiyuan University of Technology, People's Republic of China. Dr. Freeman is active in the program or organizing committees of computer vision, computer graphics, and machine learning conferences. He is a member of the IEEE.

Edward H. Adelson received the BA degree in physics and philosophy from Yale University, and the PhD degree in experimental psychology from the University of Michigan. After a postdoctoral at New York University (NYU), he worked on human and computer vision at RCA Sarnoff Labs in Princeton, New Jersey. He is a professor of vision science in the Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology (MIT) and is a member of the Computer Science and Artificial Intelligence Lab. He joined the MIT Media Lab in 1987, and moved to the Department of Brain and Cognitive Sciences in 1995. He has published numerous papers in the fields of human perception, computer vision, visual neuroscience, and image processing, and holds a dozen patents. His current interests include midlevel vision, motion perception, and the perception of materials and surfaces. His work has been honored with the Adolph Lomb Medal and the Rank Prize. He is member of the IEEE.