Color Source Separation for Enhanced Pixel Manipulations
MSR-TR-2011-98

C. Lawrence Zitnick
Microsoft Research
larryz@microsoft.com

Devi Parikh
Toyota Technological Institute, Chicago (TTIC)
dparikh@ttic.edu

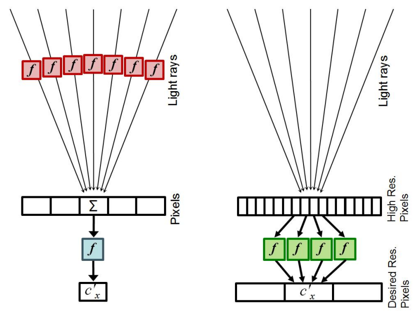
Abstract

A pixel's color value is the integration over a set of light rays. These lights rays may vary in color if they originate from different objects or surfaces that vary in albedo. To achieve photorealism, image manipulations such as contrast and gamma adjustment, histogram equalization or color replacement should ideally be applied to each light ray instead of the aggregate of the rays. In this paper, we attempt to separate the set of colors contributing to a pixel. We assume the set can be approximated using a finite set of weighted colors. Experimental results shown that two colors are sufficient. By applying image manipulations to each of the two colors independently, improved results can be obtained. We demonstrate our approach on a variety of image manipulations and compare both quantitatively and qualitatively against ground truth and baseline methods, as well as via human studies.

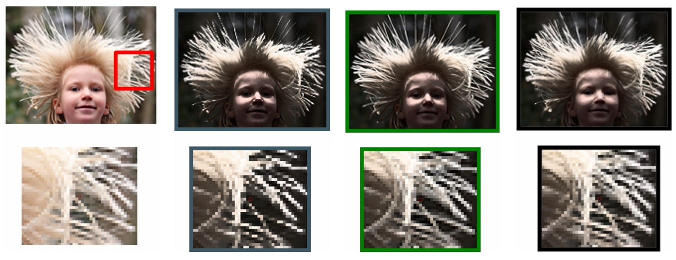
1. Introduction

Each pixel in a camera integrates light over a finite area of a lightfield [12, 7]. The set of light rays received by a pixel may vary dramatically in color and intensity. For instance, at object boundaries a pixel may receive contribution from both a foreground object and background object. Within a single object, a pixel may straddle changes in albedo. The problem of color source separation is to determine the amount and set of colors that contributed to a pixel's final color value.

The related problem of image matting attempts to separate a foreground object's color contributions along its boundary from that of the background [15, 4, 16]. Boundary matting information is useful for a variety of tasks such as image compositing and background substitution. Recently, the use of matting has also been applied to improving depth estimation [17, 2]. Other applications use image priors to estimate and restrict the number of colors contributing to a pixel, such as demosaicing [3] and image deconvolution [9].



(a) (b)



(c) (d) (e) (f)

Figure 1. Color manipulations such as increasing contrast, would ideally be applied to each light ray (a) contributing to a pixel (red boxes). Instead, manipulations are typically applied directly to a pixel's color (blue box). A high resolution image (b) can be used to approximate the full set of light rays. Applying the color manipulations to the high resolution image (green boxes) and then downsampling can create dramatically different results (e) than applying the manipulation directly to a pixel's color (d) in the low resolution image (c). (f) shows the results of our proposed approach.

In this paper, we propose a color source separation method for local pixel manipulations such as changes in contrast and gamma, histogram equalization and color replacement. Each of these manipulations transform a pixel's color based on some specified nonlinear function. To achieve photorealism, these effects should appear as if they were applied directly to the scene itself, before being captured by the camera. This is equivalent to independently applying the effect to each light ray sampled by a pixel, Figure 1(a). Professional photographers approximate this process by applying a transformation at the highest image resolution possible. The image is then downsampled to the final desired resolution, Figure 1(b). This technique is very effective as shown in Figure 1(e), which shows the dramatically different result of applying the effects before (e) or after downsampling (d). Artifacts such as over sharpening are common when applying these effects after downsampling, Figure 1(d).

We propose a technique that generates photo-realistic results for local color transformations based on estimating the distribution of light ray colors that contribute to a pixel. The common technique of applying the color transformation to a higher resolution image followed by downsampling does this explicitly. Instead, we estimate from a lower resolution image a discrete set of colors contributing to each pixel. The transformation is individually applied to each color and blended together. This provides two benefits: First, an image of higher resolution than the final desired resolution may not be available. Second by performing the pixel operations at the final desired resolution, improvements in computational efficiency may be achieved. Once the color source separation is performed off-line, the photographer can quickly experiment with various image transformations. Our results show that as little as two colors per pixel are sufficient to closely approximate the results using high resolution information.

The paper is organized a follows: In the next section we describe previous work, followed by our approach in Section 3. Implementation details are given in Section 4. We provide experimental results on a variety of tasks and compare against ground truth data and baseline approaches in Section 5. Finally, we conclude with a discussion of our approach in Section 6.

2. Previous work

Boundary matting has received significant attention over the past decade. Various methods have been proposed for separating foregrounds objects from the background using sampling techniques [15, 4] that use explicitly labeled foreground and background pixels and propagation techniques [16, 11] that assume foreground and background colors are locally smooth. The Bayesian matting approach of Chuang *et al.* [4] optimizes a likelihood function to find the optimal colors and alpha values. The Poisson matting approach of Sun *et al.* [16] assumes foreground and background colors are locally smooth and that the gradient of the matte is similar to that of the image. Levin *et al.* [11] extends this approach by fitting a linear model to both the foreground and background colors. Wang and Cohen [19] proposed a robust approach for handling complex foreground and background color distributions.

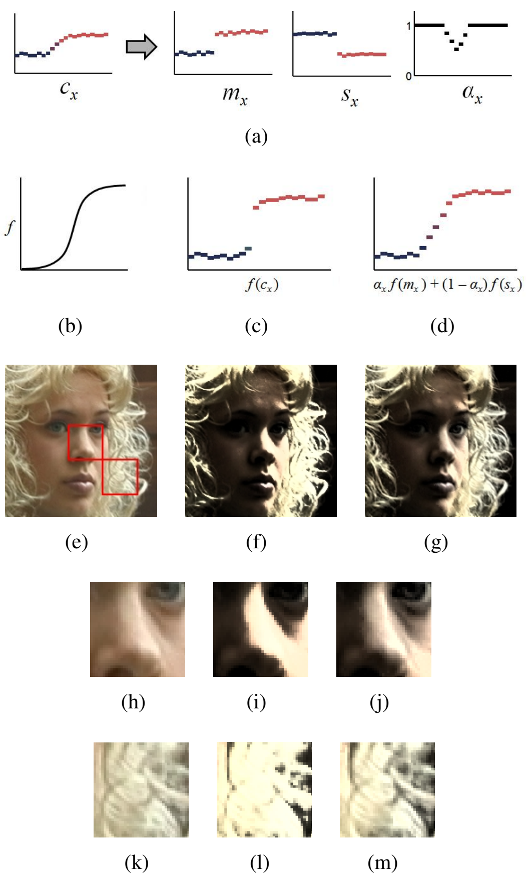


Figure 2. A 1D representation (a) of the image demonstrates the primary color m_x, the secondary color s_x and alpha value α_x computed for a set of pixels. A function f (b) may be applied to either the pixel values c_x (c) or the computed primary and secondary colors (d). The effect of applying the function f directly to c_x results in over sharpening and loss of edges (c, f, i, l). Applying f to m_x and s_x results in more photo-realistic results (d, g, j, m). Close-ups (h-m) are labeled in (e).

Recently, matting has been applied to a wider variety of applications. Bennett *et al.* [3] used matting information to improve the estimation of colors for demosiacing. Several works have combined matting and depth estimation for improved results [8, 17, 2]. Finally, matting has been used in the development of image priors for deconvolution and denoising [9].

Recently, work has also been done on image recoloring using gradient information [10] and edge preserving tonemapping [14]. Bae *et al.* [1] proposed an image manipulation technique for achieving a pictorial look using two scales based on bilateral smoothing.

3. Approach

In this section, we describe our approach to color source separation for use in non-linear pixel transformations such as contrast adjustment, histogram equalization, gamma correction or recoloring.

3.1. Image Formation

Each pixel in a camera receives a set of light rays over a finite area of the observed lightfield [12, 7]. The color value c_x of the pixel x is found by integrating over the set of observed light rays L_x, i.e. $c_x = \int_{l \in L_x} l$. To maintain photorealism, a color transformation f should ideally be applied independently to all light rays contributing to a pixel x to obtain our new pixel value $\acute{c}_x$,

$$\acute{c}_x = \int_{l \in L_x} f(l). \tag{1}$$

As shown in Figure 2, this is critical for maintaining edge profiles. If the transform f is applied directly to the pixel's value, i.e. $\acute{c}_x = f(c_x)$, over sharpening and loss of edges may occur, Figure 2(c,f,i,l). If a higher resolution image is available, L_x can be approximated using the pixels in the high resolution image contributing to the pixel c_x in the downsampled version, Figure 1(e). However, a higher resolution image may not always be available, and due to computational costs it might be advantageous to only manipulate the image at the resolution of the final desired image.

In this paper, we make the following assumption: Each pixel receives contribution from a finite set of color regions. That is, the set of light rays L_x only contains a small number of colors resulting from different objects, albedo, or lighting conditions. Formally our model is as follows: A pixel x's color c_x, is a linear combination of k colors $\{c_x^1, \ldots, c_x^k\}$ with a corresponding set of weights w_x, such that $\sum_i w_x^i = 1$ and:

$$c_x = \sum_{i \in k} w_x^i c_x^i \tag{2}$$

Note we assume the camera's response function is linear [13], or at least locally linear in color space. Given some function of a pixel's color f, we compute our transformed color using:

$$\acute{c}_x \approx \sum_{i \in k} w_x^i f(c_x^i) \tag{3}$$

If f is a nonlinear function such as contrast adjustment, the value of $\acute{c}_x$ will vary from $f(c_x)$. As stated above, using equation (3) we can achieve greater photorealism by better approximating the effect of applying the color transformation directly to L_x, Figure 2(d, g, j, m).

3.1.1 Estimating the number of color regions

We assume each pixel's color is a combination of a finite number of color regions. In this section we experimentally determine how many color regions are needed to achieve an accurate estimation of $\acute{c}_x$. We estimate the set of colors a pixel receives by sampling 10×10 patches from high resolution images. For each patch k-means clustering is performed on its colors for varying values of k. Finally, we compute the error between estimating the value of $\acute{c}_x$ using the original sampled colors from the 10×10 patch weighted uniformly and using the weighted k color clusters. Equation (3) is used in both cases. Results are shown in Figure 3 for various adjustments to contrast and gamma as defined in Section 5. The error decreases as k increases. However, the rate of improvement is significantly reduced for k greater than 2. As a result, we use $k = 2$ through out this paper.

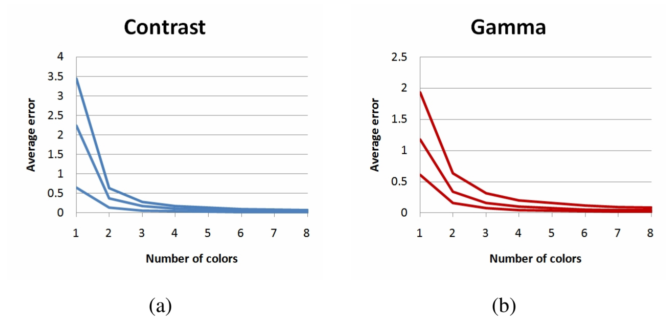


Figure 3. Graph of average intensity errors for contrast and gamma adjustment. Pixel color models with one to eight clusters from a ground truth set of 100 colors are used. Three settings for contrast and gamma adjustment are shown.

3.2. Color Model

We assume each pixel's color contributions can be modeled using two colors. For ease of notation, we call these two colors the primary color m_x and the secondary color s_x, with corresponding weights α_x and $(1 - \alpha_x)$. We assume the contribution of the primary color is always greater, i.e. $\alpha_x \geq 0.5$. Using this notation, equation (3) can be rewritten as:

$$\acute{c}_x \approx \alpha_x f(m_x) + (1 - \alpha_x) f(s_x). \tag{4}$$

To find the transformed color value $\acute{c}_x$ for each pixel x, the values of m_x, s_x and α_x need to be computed. We accomplish this by first estimating a small set of colors $H_x = h_{x1}, \ldots, h_{xk}$ that may contribute to c_x. The set of colors H_x are computed from the set of pixel colors C_x that exist in a small spatial neighborhood of x. The exact method for finding the set of colors H_x is described in the following section. H_x typically contains one to four colors, depending on the amount of local texture surrounding x. Using H_x, we model c_x as:

$$c_x = \alpha_x h_{xi} + (1 - \alpha_x) h_{xj} + n_x \tag{5}$$

where $h_{xi}, h_{xj} \in H_x$ and n_x is the residual or difference between the linear combination of h_{xi} and h_{xj}, and the observed color c_x. To ensure that equation (2) holds, we add the residual n_x onto h_{xi} and h_{xj} based on α_x to find our final colors m_x and s_x,

$$m_x = h_{xi} + \alpha_x n_x, s_x = h_{xj} + (1 - \alpha_x) n_x. \tag{6}$$

We choose the colors h_{xi} and h_{xj} from H_x, and estimate our alpha value α_x by maximizing the following likelihood function:

$$\begin{aligned} p(h_{xi}, h_{xj}, \alpha_x, n_x | c_x, C_x) = \\ p(h_{xi}|C_x) p(h_{xj}|C_x) p(\alpha_x) p(n_x | c_x, h_{xi}, h_{xj}, \alpha_x) \end{aligned} \tag{7}$$

where C_x is the set of pixel colors within a small spatial neighborhood of x.

The process of maximizing equation (7) is accomplished by evaluating it for every possible combination of colors h_{xi} and h_{xj} from H_x. For each set of colors h_{xi} and h_{xj} we determine the alpha value that maximizes $p(\alpha_x) p(n_x | c_x, m_x, s_x, \alpha_x)$. The set of values for h_{xi}, h_{xj} and α_x that result in the maximum value for equation (7) is then used to compute m_x and s_x in equation (6).

We now define the four terms of equation (7). The values of $p(h_{xi}|C_x)$ and $p(h_{xj}|C_x)$ are defined in the next section. The prior $p(\alpha_x)$ on α_x is used to bias pixels to receive contribution from only one color. Mixed pixels only occur at color boundaries and are therefore less likely to occur. As a consequence, we penalize alpha values that are not equal to 1:

$$p(\alpha_x) \propto \left\{ \begin{array}{cc} 1 & \alpha_x = 1 \\ \kappa & otherwise \end{array} \right\} \tag{8}$$

κ has a value less than one.

The final expression $p(n_x | c_x, h_{xi}, h_{xj}, \alpha_x)$ of (7) is the probability of n_x given the camera's noise model. The value of n_x is computed directly from the difference between the observed color c_x and the linearly blended color:

$$n_x = c_x - (\alpha_x h_{xi} + (1 - \alpha_x) h_{xj}). \tag{9}$$

In many digital images, the amount of noise is highest for mid-range values and decreases as the intensity becomes higher or lower. Within this paper, we assume the noise model for the camera is known, by either using a calibration grid, or from automatic methods [6, 18]. If $\sigma(c_x)^2$ is the variance predicted by the noise model for a color c_x:

$$p(n_x | c_x, m_x, s_x, \alpha_x) = \mathcal{N}(n_x; 0, \sigma(c_x)^2) \tag{10}$$

where $\mathcal{N}$ is a standard normal distribution. The color variance $\sigma(c_x)^2$ may be a full covariance matrix or a single scalar, depending on the complexity of the image noise model used.

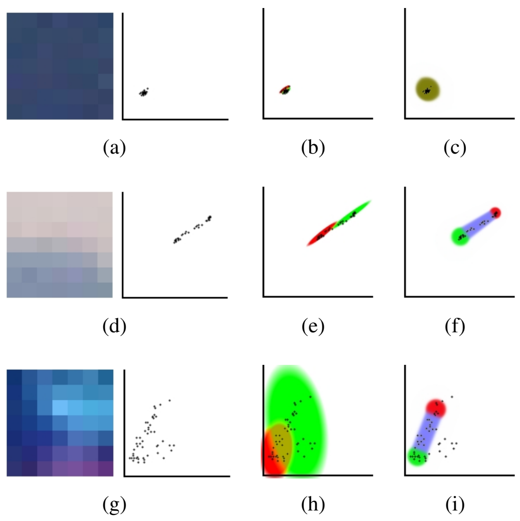


Figure 4. Illustration of different techniques for finding the potential colors within C_x. (a,d,g) are color patches from real images with 2D plots of red and green color values. (b,e,h) show the results using a two component GMM, with the green and red ellipses showing the extent of the components Φ_0 and Φ_1. (c,f,i) show the results with the mixed and outlier components added. Blue shows the extent of the mixed color component Ψ_{ij}, and yellow is the combination of both Φ_0 and Φ_1.

We need to determine the value for α_x that maximizes $p(\alpha_x) p(n_x | c_x, m_x, s_x, \alpha_x)$. Since it is not continuous, we cannot solve the expression directly. However, we can evaluate it twice, once for $\alpha_x = 1$ and once for $\alpha_x \neq 1$, and find the maximum. For $\alpha_x = 1$, we can solve for n_x using (9), and directly compute the result.

When $\alpha_x \neq 1$, the value of $p(\alpha_x) = \kappa$, and we can solve for the $\hat{\alpha}_x$ that maximizes $\kappa p(n_x | c_x, m_x, s_x, \alpha_x)$. The value of $\hat{\alpha}_x$ is determined by finding the point the minimizes the distance between c_x and the line segment from h_{xi} to h_{xj}:

$$\hat{\alpha}_x = \frac{\mathbf{u} \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}} \tag{11}$$

where $\mathbf{u} = c_x - h_{xj}$ and $\mathbf{v} = h_{xi} - h_{xj}$. To be considered as a possible alpha value, $\hat{\alpha}_x$ must lie between 0.5 and 1.

3.3. Color Estimation

In this section, we explore a technique for finding a set of colors H_x, given the set of pixel color values C_x in a spatial neighborhood of x. In practice the spatial neighborhood used to create C_x may vary from 3×3 to 11×11 and the number of colors l in H_x from two to four, depending on the amount of noise and texture in an image. Except in Section 3.3.1, we assume the number of colors l and the window size are given to the algorithm. If we assume the world consists of small regions with constant color, pixels receiving contribution from a single region will form clusters of points within C_x. The pixels that receive contribution from multiple color regions will be scattered between the clusters. The distribution of pixel colors in C_x can vary greatly across the image. For instance, a single mode may be found in constant colored areas (Figure 4a), two main modes with samples in between for border areas (Figure 4d), or several modes in highly textured areas (Figure 4g).

Our goal is to find the set of colors H_x that contribute to the set of pixel colors C_x. We model H_x as a set of clusters using a Gaussian mixture model (GMM). Each component $\Phi_i = \{h_i, \Sigma_i\}$ in the model consists of a mean h_i in color space and corresponding covariance matrix Σ_i. Thus, the likelihood of $c \in C_x$ given Φ_i is:

$$p(c|\Phi_i) \propto \mathcal{N}(c; h_i, \Sigma_i). \tag{12}$$

A standard method for learning a GMM is to use Expectation Maximization (EM) [5]. The results of using EM for a two component GMM, using only color information, are shown in Figures 4(b,e,h). There are several problems with this approach. First, even if one mode exists, it is split into multiple clusters (Figure 4b). Second, mixed pixels that lie between the modes contribute to the mean of the two components (Figure 4e). Finally, color outliers also contribute to the means, skewing the results (Figure 4h).

To address these problems we make several additions to the basic algorithm described above as first proposed in [9] for the case of just two colors. First, we assume that any variance of the colors within a cluster is due to image noise and not texture from the world. Thus, we fix the color variance Σ_i equal to that of the image noise $\sigma(h_i)^2$. As a result, if a single mode exists within C_x, all components of the GMM will merge to have the same mean and extent (Figure 4c).

To handle mixed colors and outliers, two components are added to our model. Our first component models mixed pixels that lie between two color clusters Φ_i and Φ_j. For $c \in C_x$, the component $\Psi_{ij} = \{\kappa, h_i, h_j\}$ representing mixed pixels between h_i and h_j is modeled as:

$$p(c|\Psi_{ij}) \propto \kappa \mathcal{N}(c; \tilde{c}, \Sigma(\tilde{c})) \tag{13}$$

where $\tilde{c}$ corresponds to the point closest to c on the 3D line segment between h_i and h_j. κ is set to the same value as in equation (8). In estimating the mixed pixels, all combinations of h_i and h_j are considered.

Our second additional component is a uniform distribution Υ used to model outliers within C_x. Thus, Υ is equal to some small constant and in our experiments $p(c_i|\Upsilon) = 10^{-6}$.

Combining all the above components we model the likelihood of a color c as:

$$p(c|\Phi, \Psi, \Upsilon) \propto \sum_i p(c|\Phi_i) + \sum_{i,j} p(c|\Psi_{ij}) + p(c|\Upsilon) \tag{14}$$

The results of adding the new components and solving using EM, can be seen in Figures 4f and 4i. In both cases, the means better model the peaks of their corresponding modes, since the mixed pixels and outliers are modeled by Ψ and Υ respectively.

Since EM is computationally expensive and must be computed for each pixel, we use the approximate method k-means to initialize the color values of H_x, [3]. After k-means, four iterations of EM are used to refine the values.

3.3.1 Using Adaptive Color Models

In the previous section we assumed the number of colors l in H_x was given. By placing a prior on the number of colors in H_x we can automatically determine the number of colors required to represent the set of pixel colors in C_x. If $\Phi(l)$ and $\Psi(l)$ are the mixture components computed using a model with l colors, the optimal number of colors can be found by:

$$\arg\max_l \Gamma(l) \prod_c p(c|\Phi(l), \Psi(l), \Upsilon) \tag{15}$$

where $\Gamma(l)$ is the prior probability of observing l colors in an image patch.

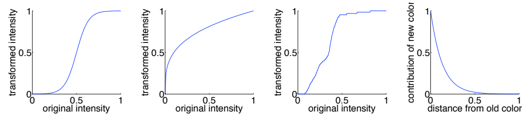
4. Implementation Details

In this section we describe some of the implementation details of our algorithm. The use of k-means to initialize the EM algorithm proposed in Section 3.3 helps improve computational efficiency. Performance gains may also be achieved by only computing the color model in textured areas. If the standard deviation in a 5×5 window is below the noise level (5 in our experiments), the color model is not computed and the pixel is assigned a single color.

5. Experimental Results

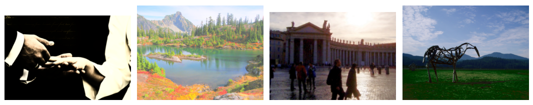
We demonstrate our proposed approach on four image transformations: contrast change, gamma change, histogram equalization and re-coloring. To quantify our results, we collected a dataset of 55 high resolution images, downsampled by a factor of 5 - 10 to $\sim 320 \times 240$. We compute an estimate of the ground truth for these images by transforming the original high resolution image, and then downsampling the transformed image to 320×240. Since the images are of sufficiently high resolution, we believe this accurately samples the distribution of light rays for each pixel.

The lower resolution original images are transformed using our approach of transforming the found colors m_x and s_x before blending, as well as the naive approach of transforming the original pixel's color, $f(c_x)$. We also compare our results to a baseline approach of upsampling the low



(a)

(b)



(c)

Figure 5. The nonlinear curve (a) used for each transformation (left to right) contrast, gamma, histogram equalization and re-coloring ; (b) shows an example image and the transformed image (c).

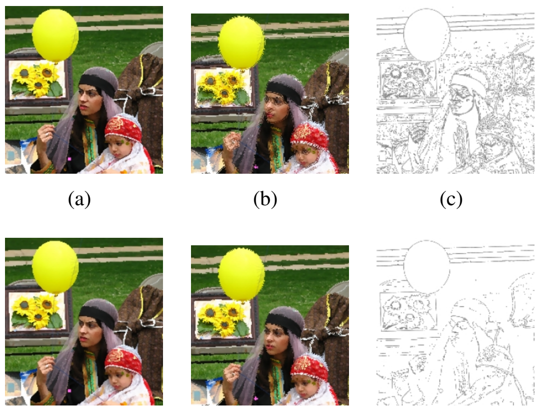


Figure 6. Example of image decomposition using $\kappa = 0.1$ (a,b,c) and $\kappa = 0.0001$ (d,e,f): (a,d) main color layer, (b,e) secondary layer, (c,f) alpha layer.

resolution original image using bicubic interpolation, transforming each pixel color in the resultant high resolution image, and then downsampling. We use the average root mean squared error and PSNR metrics. We now describe the specifics of the four image transformations we used.

After m_x, s_x and α_x are computed for each pixel, image manipulations can be performed in realtime. The additional cost is proportional to the number of mixed pixels M found in the image, which is typically less than 10%. As a result, performing operations directly on the low resolution $N \times N$ image is significantly faster than performing the same operation on a high resolution $HN \times HN$ image and downsampling, i.e. $H^2 \gg M$.

Contrast change: We apply the transformation to each of the RGB color components of the color c using $f(c) = c\frac{i'}{i}$, where i is the intensity of the color c computed as the average of the color components (ranging from 0 and 1), and i' is the contrast transformed intensity computed as

$$i' = \frac{1}{1 + \exp(-t(i - 0.5))}. \tag{16}$$

In our experiments, we set the parameter t to 15. The resultant curve is shown in Figure 5, along with an example contrast changed color image.

Gamma change: Similar to contrast change, we transform each color component using $f(c) = c\frac{i'}{i}$, where i' is now the gamma changed intensity computed as $i' = i^{\gamma}$. We set γ to 0.3. The resultant curve and an example gamma changed image are shown in Figure 5.

Histogram equalization: We transform each color component $f(c) = c\frac{i'}{i}$, where i' is the mapping of i using histogram equalization. We apply standard histogram equalization to the original grey scale low resolution image to obtain the non-parametric mapping $i' = h(i)$. This mapping is used for all methods being compared. An example of this mapping, and the associated image transformation is shown in Figure 5.

Re-coloring: This operation replaces a color c_s in an image with another color c_t as specified by the user. The amount a color is shifted towards c_t is based on its proximity to c_s. Our transformation is achieved using $f(c) = c + \beta\,(c_t - c_s)$ with $\beta = \exp(-d/0.13)$, where d is the squared euclidian distance between c and c_s. This fall off in the contribution of c_t to a pixel as a function of the distance of c to c_s, can be seen in Figure 5 along with an example re-colored image.

For our experiments to avoid having a user in the loop, c_s was set to the dominant color in the image (computed using k-means clustering on all the pixels in the image to obtain 5 clusters), and c_t was randomly selected from a uniform distribution over the RGB color space.

5.1. Results

We show example decompositions of an image using our approach (3×3 window, $l = 2$ colors) in Figures 6 for values of $\kappa = 0.1$ and $\kappa = 10^{-4}$. As would be expected, a higher value of κ results in more pixels having fractional alpha values.

Examples demonstrating the improved image quality (in terms of being closer to ground truth) using our proposed approach for each of the four above transformations, as

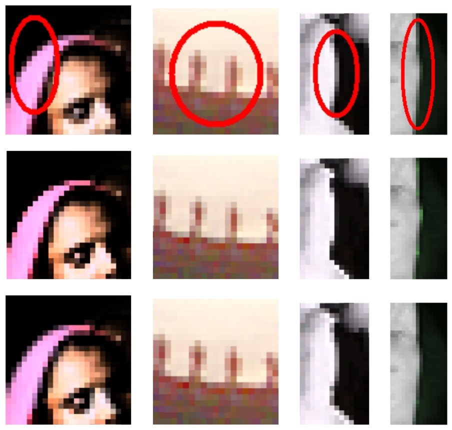


Figure 7. Comparison of our method as compared to the naive method. Columns: contrast change, gamma change, histogram equalization and re-coloring. Rows: ground truth, naive method and proposed approach. Notice over-sharpening and color leaks in the naive approach.

compared to the naive method can be seen in Figure 7.

Figures 10 and 11 compare our approach to the naive and baseline approaches across the four color transformations. We plot mean squared error and PSNR results using between two and four colors for l, as well as the adaptive-model approach of Section 3.3.1. In addition, various window sizes between 3×3 and 11×11 for creating C_x are tested. All examples in this test case use $\kappa = 10^{-4}$. For several transformations, the two-color model performs better for smaller window sizes, while the three-color and four-color models perform better for bigger window sizes. The adaptive-color model that automatically determines the number of colors in a given window captures a balanced trade-off between these, and often performs the best for a given window size (especially for histogram equalization). In most cases, the errors are lower than the naive and baseline methods, and significantly so in several cases.

Since RMSE and PSNR are known to not capture perceptual differences in images especially near edges, we provide results from a human study using Amazon's Mechanical Turk. We asked 20 subjects to rate 14 images using the 2-color and naive approaches. Figure 8 shows that 61% thought the 2-color model was better, while 11% thought it was worse, and 27% saw no difference. Sample images from the study using varying intensity curves are shown in Figure 9(a, b) and for re-coloring in Figure 9(c). Other qualitative results are shown in Figures 1(f) and 2(g, j, m). Notice the removal of over-sharpening and color leaking artifacts using the proposed approach.

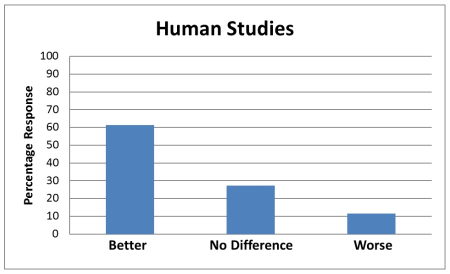


Figure 8. Results of human studies comparing the 2-color model to the naive model. The first bar shows the percentage that thought the 2-color model was better, followed by the same and worse.

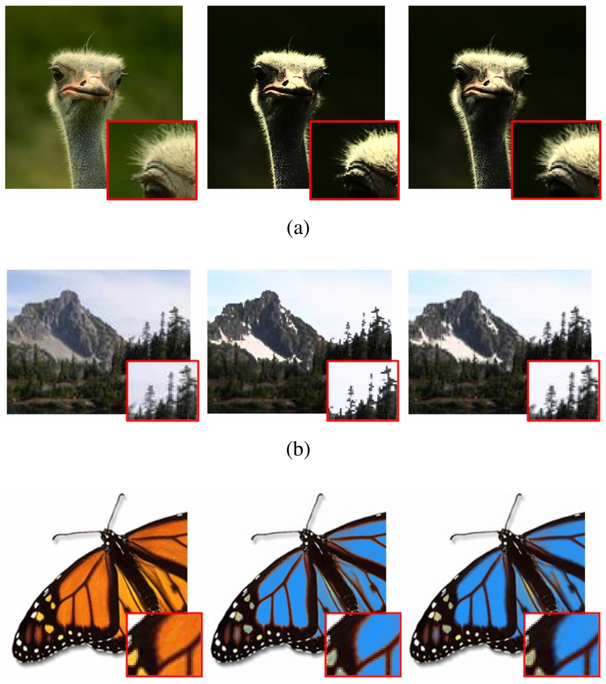


Figure 9. Qualitative results for arbitrary intensity curves (a, b) and re-coloring (c). The images from left to right are the original, the naive approach, and the proposed approach.

6. Discussion

In our paper we assume the distribution of light rays can be modeled using just two colors. Clearly, pixels may receive contribution from a large set of colors, especially when portions of the scene are blurred. Using more colors could increase the accuracy of the approach. Unfortunately, determining the contribution of each color is in many cases ambiguous, such as the case when using three colors and one color is a linear combination of the other two.

We studied four different possible image manipulations. Many other applications exist that could take advantage of

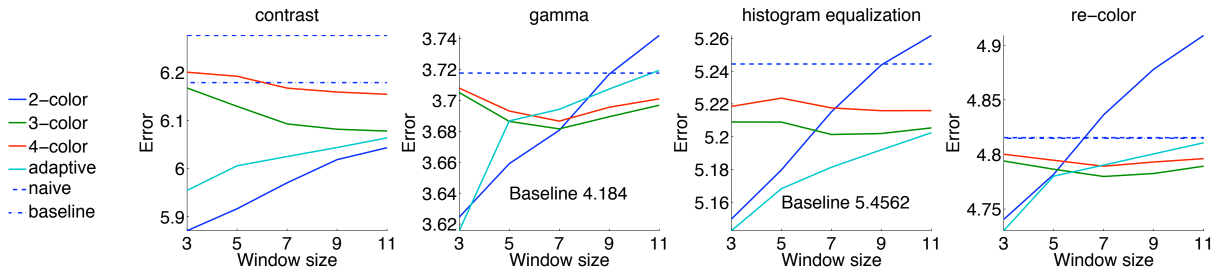


Figure 10. The root mean squared error of our method with different parameter settings, as compared to the naive and baseline methods. For the cases where the baseline error is too high, we report it on the plot in text.

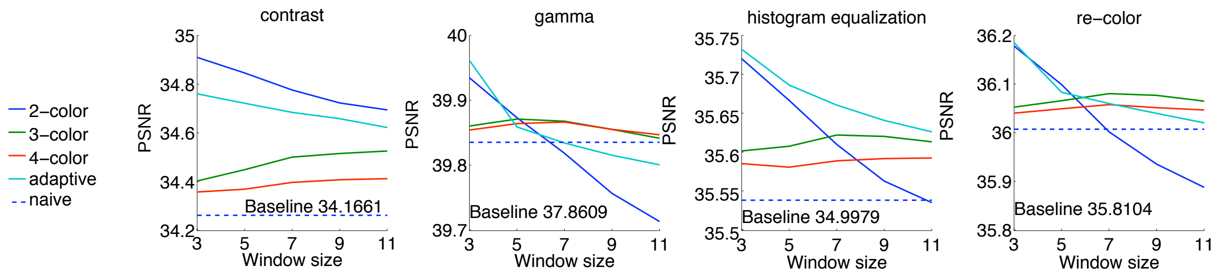


Figure 11. The PSNR of our method with different parameter settings, as compared to the naive and baseline methods. The PSNR of the baseline method is reported on the plots in text.

our approach. For instance, soft brushes could use the color contribution information to properly handle object boundaries at interactive speeds. Similarly, the information could be used for image selection tasks with soft boundaries.

If computational efficiency is a primary concern and high resolution information is available, the color model could be built directly from the high resolution information. The image manipulations could then be applied to a lower resolution image without loss of quality.

In conclusion, we present a color source separation method for estimating the two main colors that contribute to each pixel in an image. By applying color transformations directly to these computed colors and blending the result, improved photorealism can be achieved without the need for higher resolution information.

References

[1] S. Bae, S. Paris, and F. Durand. Two-scale tone management for photographic look. In *Proc. of SIGGRAPH 2006*, pages 637–645, 2006. 2

[2] Y. Bando, B. Chen, and T. Nishita. Extracting depth and matte using a color-filtered aperture. In *Proc. of SIGGRAPH Asia*, 2008. 1, 2

[3] E. Bennett, M. Uyttendaele, C. L. Zitnick, R. Szeliski, and S. B. Kang. Video and image bayesian demosaicing with a two color image prior. In *IEEE Proc. of ECCV*, volume 1, pages 508–521, 2006. 1, 2, 5

[4] Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski. A Bayesian approach to digital matting. In *IEEE Proc. of CVPR*, volume 2, pages 264–271, 2001. 1, 2

[5] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977. 5

[6] D. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995. 4

[7] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *Proc. of SIGGRAPH 1996*, pages 43–52, 1996. 1, 3

[8] N. Joshi, W. Matusik, and S. Avidan. Natural video matting using camera arrays. In *Proc. of SIGGRAPH 2006*, 2006. 2

[9] N. Joshi, C. L. Zitnick, R. Szeliski, , and D. Kriegman. Image deblurring and denoising using color priors. In *IEEE Proc. of CVPR*, 2009. 1, 2, 5

[10] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. In *Proc. of SIGGRAPH 2004*, 2004. 2

[11] A. Levin, D. Lischinski, and Y. Weiss. A closed form solution to natural image matting. *IEEE Trans. on PAMI*, 30(2), 2008. 2

[12] M. Levoy and P. Hanrahan. Light field rendering. *Proc. of SIGGRAPH 1996*, pages 31–42, 1996. 1, 3

[13] S. Lin, J. Gu, S. Yamazaki, and H. Shum. Radiometric calibration from a single image. In *IEEE Proc. of CVPR*, 2004. 3

[14] D. Lischinski, Z. Farbman, M. Uyttendaele, and R. Szeliski. Interactive local manipulation of tonal values. In *Proc. of SIGGRAPH 2006*, pages 646–653, 2006. 2

[15] M. A. Ruzon and C. Tomasi. Alpha estimation in natural images. In *IEEE Proc. of CVPR*, volume 1, pages 18–25, 2000. 1, 2

[16] J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum. Poisson matting. In *Proc. of SIGGRAPH 2004*, pages 315–321, 2004. 1, 2

[17] Y. Taguchi, B. Wilburn, and C. L. Zitnick. Stereo reconstruction with mixed pixels using adaptive over-segmentation. In *IEEE Proc. of CVPR*, 2008. 1, 2

[18] Y. Tsin, V. Ramesh, and T. Kanade. Statistical calibration of CCD imaging process. In *IEEE Proc. of ICCV*, volume 1, pages 480–488, 2001. 4

[19] J. Wang and M. Cohen. Optimized color sampling for robust matting. In *IEEE Proc. of CVPR*, 2007. 2