Color Lines: Image Specific Color Representation.

Ido Omer and Michael Werman
School of computer science and engineering
The Hebrew University of Jerusalem
Email: {idom,werman}@cs.huji.ac.il

Abstract

The problem of deciding whether two pixels in an image have the same real world color is a fundamental problem in computer vision. Many color spaces are used in different applications for discriminating color from intensity to create an informative representation of color. The major drawback of all of these representations is that they assume no color distortion. In practice the colors of real world images are distorted both in the scene itself and in the image capturing process. In this work we introduce ***Color Lines****, an image specific color representation that is robust to color distortion and provides a compact and useful representation of the colors in a scene.*

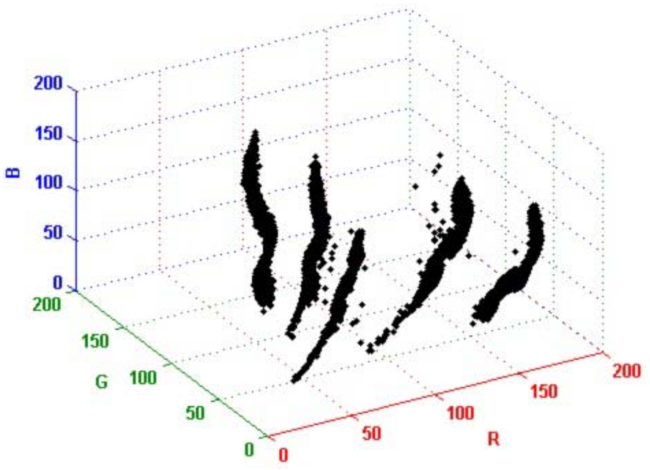

Figure 1: An *RGB* histogram.

1. Introduction

Color representation is a fundamental problem in computer vision. Many applications depend on a good color representation in order to perform well. Although most color images are captured in *RGB*, this color space is rarely used for computer vision tasks. The main reason is that *RGB* does not separate color from intensity which results in highly correlated channels.

Computer vision is mainly about retrieving information from images. Color is one of the basic information cues in images, but extracting information from the *RGB* values is not a trivial problem. We would like to separate the *RGB* information of each image's pixel into surface color, illumination color and intensity, noise etc. Unfortunately, this task is usually impossible to achieve. Another problem with the *RGB* coordinate system is that it does not provide an intuitive distance metric - the distance in *RGB* coordinates is not proportional to the color difference. A good color representation should also allow us to have better compression (similar pixels will have similar values), to reduce noise more efficiently and to achieve better image understanding.
Most color models assume that real world colors form linear color clusters that intersect the origin in the *RGB* histogram domain. This assumption is based on the fact that for lambertian objects, the emitted surface color is a multiplication of the surface albedo with the illumination. According to this assumption the *R/G/B* ratio of pixels having the same albedo color will be identical. Figure 1 shows the histogram of a real image and clearly illustrates that the color clusters do not form straight lines.
In this work we create an image-specific color model that is aimed at helping in providing a robust method for deciding whether two pixels share the same real world color or not and creating a better distance metric for the colors of an image.

Color Models Many color spaces have been suggested to separate color from intensity and create a method for deciding whether two pixels share the same color or not. These color spaces can be divided into two groups *Linear* and *Non-Linear* ones.
Among the linear ones, the most widely used are the *YCrCb*, *YUV* and *YIQ*. Among the non-linear ones, two groups are very popular [3]. The *HSV HSI*, *HSB*, *HSL*, ... color spaces separate the color into *Hue* - Color, *Saturation* - color saturation (purity) and *Value* - the intensity. The other group is the *CIE-LAB* and *CIE-LUV* color spaces that separate color into luminance and two color coordinates in an effort to create a color space which is perceptually uniform [2].

Previous works have suggested *application specific* color models, for example the $I_1 I_2 I_3$ color model proposed in [7] for image segmentation. These works do not try to exploit image specific attributes.
Throughout the paper, we refer to all these models as *Generic Models*, as they depend only on the *RGB* values of a single pixel.
A major drawback of all the generic color models is that they are all obtained through a fixed transformation from *RGB* without considering specific image properties.

Motivation To decide whether two pixels have the same real world color or not, we use the color coordinates of a generic color model. As explained above, these color models assume either there is no color distortion or there is an identical color distortion for all imaging conditions. In practice, when dealing with real world images of an unknown source, this assumption is rarely true as scene surface color is distorted differently in different images, depending on the scene and camera settings.
Figure 1 shows the histogram of 5 color patches. Although the color clusters are very obvious, it is clear they do not form straight lines that intersect the origin. As will be shown later on, there are many phenomena that occur in the scene itself as well as during the image capturing process, which distort color and cause pixels with the same real world color to have different color coordinates (or R/G/B ratios).
In spite of the color distortion, when looking at the *RGB* histogram of real world images, two important facts can clearly be observed; The histogram is very sparse, and it is structured. We used the Berkeley Segmentation Dataset [1] to provide statistics of histogram sparseness. The average number of non empty histogram bins for all 200 training images of the dataset is 0.22%. Not only the number of non empty histogram bins is very small, they are not homogeneously distributed in the *RGB* space and most pixels are contained in very small regions of the histogram space. 90% of the non-empty histogram bins have non-empty neighboring bins. This is because the colors of almost any given scene create very specific structures in the histogram.

Our ***Color Lines*** model exploit these two properties of color histograms by describing the elongated color clusters. By doing so, *Color Lines* create an image specific color representation that has two important properties: Robustness to color distortion and a compact description of colors in an image.

The following section discusses various reasons for color distortion in natural images. We then present our color lines model and provide an efficient algorithm for approximating the optimal *Color Lines* description of an image. Section 4 shows some experimental results. We suggest some possible uses of our model in Section 5. Section 6 summarizes our work.

2. Color Distortion

The colors of most natural images undergo a few types of distortion. In images taken using digital cameras, these phenomena can be divided into three groups (The color distortion of film-based cameras is similar):

1. Scene related color distortion.
2. Sensor related color distortion.
3. Other camera related color distortion.

Scene related color distortion Some of the deviation from the linear model is within the scene itself. Specularity, that has been described using the T-shape (or L-shape) model in [8], and inter reflection are two very important phenomena that result in non linear color clusters. Figure 2 shows the histogram of a specular object. Another example for a phenomenon that yields non linear color clusters, is the color of the sky. Figure 3 shows a typical sky color histogram.

Sensor related color distortion Sensor-related color distortion includes sensor saturation and cut off. Above a certain threshold the camera sensors reach saturation and produce a constant response, whereas below another threshold the sensor enters it's cut-off zone where it generates no response at all. Figure 4 shows the histogram of a saturated scene.

Other camera related color distortion Besides sensor-related distortion most digital cameras have other kinds of color distortion. These are usually referred to as *gamma correction* although in practice it is usually not a classic gamma correction, rather a more complicated color manipulation that attempts to extend the camera's dynamic range and enrich the color, providing images that are visually pleasing. Color manipulations performed by different cameras vary between different manufacturers and different models. An extensive survey on the subject can be found in [5]. Figure 5 shows two histograms of the same image. The image was taken with a camera that can save the image as raw data. Figure (a) shows the histogram of the raw image data, figure (b) shows the histogram of the processed image the user receives.
Another distortion from the linear color assumption is caused by blurring. Along edges and blurred regions each pixel might combine color information from different objects or regions resulting in non linear color clusters.

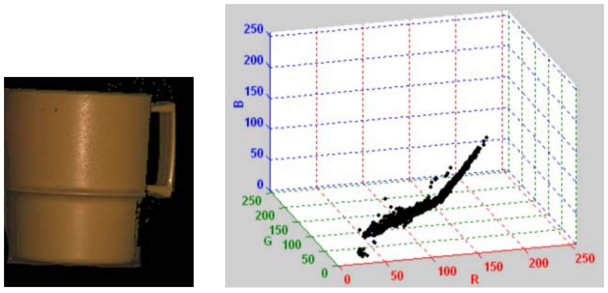

Figure 2: A specular object (a) and it's histogram (b).

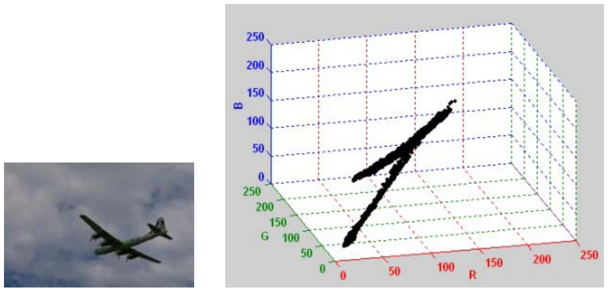

Figure 3: Sky color usually gradually varies gradually from blue to white/gray and does not form a linear color cluster in the *RGB* color space. in (a) we have an image of a plane in the sky (from the Berkeley segmentation database) it's histogram is in (b).

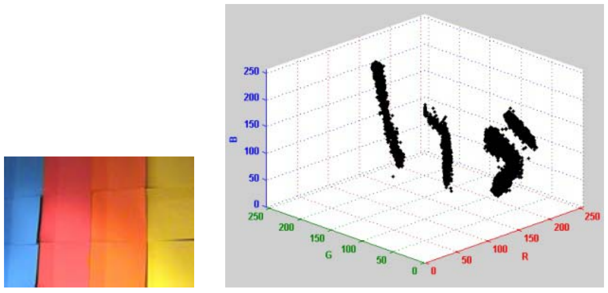

Figure 4: A saturated image (a) and it's histogram (b).

Due to the above reasons, color clusters in real world images can't be represented using merely two color coordinates (or a color line through the origin) as is usually the case when using a generic color space. Instead, a higher dimensional representation should be used. In this paper we suggest modelling color in an image-specific way that will be as immune as possible to color distortion, yet simple and easy to construct.

3. Color Lines

The previous section presented various color distortions that are found in natural images. Recovering a model that pro-

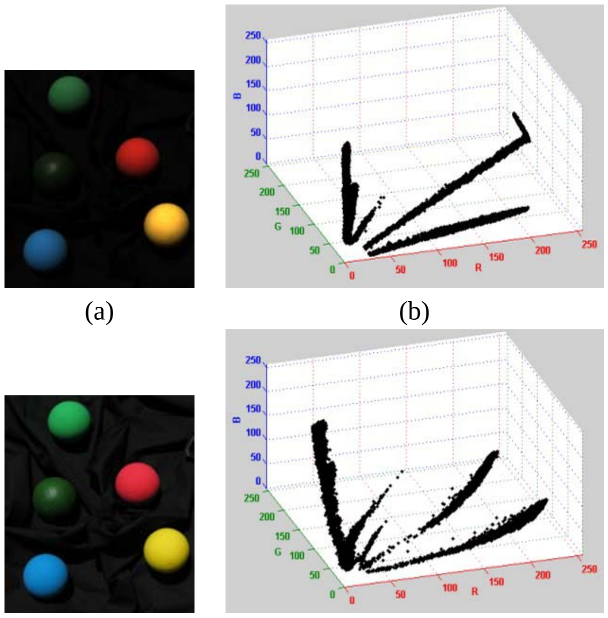

Figure 5: Figures (a) and (c) show the same image, in (a) we see the raw image data and in (c) the output image to the user after color manipulation done inside the camera. Figures (b) and (d) shows the histograms of the above images. In figure (a) part of the image pixels are saturated while in figure (b) they are not, this is due to the fact that the actual bit depth of the camera is larger than 8 therefore, without manipulating color, values larger than 256 are clamped.

vides a full explanation for the color of each pixel is a difficult (probably impossible) problem. Our goal is to produce a simple model of image color that will be robust to color distortion and will allow us to decide whether any 2 pixels share the same real world color or not.

To achieve this goal we use ***Color Lines***. The *Color Lines* model of an image is a list of lines representing the image's colors along with a metric for calculating the distance between every pixel and each *Color Line*. Each *Color Line* represent an elongated cluster in the *RGB* histogram. In our implementation the *Color Line* is defined by a set of *RGB* points along it's skeleton and a 2D gaussian neighborhood for each point in the plane that is perpendicular to the line (the cluster's are non-isotropic around their skeleton). Figure 6 shows two color lines. The green one is shown with 2D gaussians around each point.

The average number of color lines of an image is 40, on an average, each color line is defined by 6 color points (and their 2D gaussian neighborhoods), hence the size of an average model is about 1440 parameters (Our model uses about 240 color points. Each point is represented by 6 parameters: 3 for the location and 3 are the 2D gaussian parameters).

By using this very general model, the only assumptions we make about color behavior is that the norm of the *RGB* color

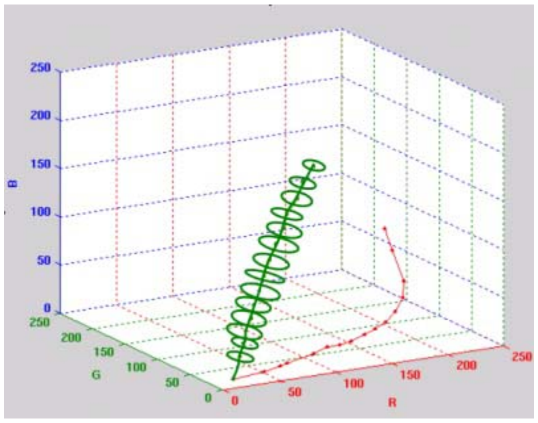

Figure 6: Two color lines, the green one is drawn with the 2D gaussians around each color point.

coordinates of a given real world color will increase with an increase of the illumination and that the change in the *R/G/B* ratio will be smooth.

Searching the *RGB* color space for the optimal elongated clusters is a difficult problem. Several phenomena that make this search so difficult are the noise and quantization in the histogram domain, edges and reflections that make the separation of color clusters difficult and the fact that many images tend to be very "grayish", which again makes the separation of the color clusters harder. In spite of the above problems, we are able to use natural image properties in order to make this search feasible. One property that can be used in order to simplify our search, is the fact that we are actually not looking for a general line in the histogram 3D space, rather we have strong knowledge of the line's orientation.

3.1. Computing Color Lines

We present a simple and efficient algorithm in order to recover the color clusters. The algorithm outline is shown in **Algorithm 1**.

Algorithm 1 Computing Color Lines

```
1. Construct histogram slices.
2. foreach slice
       - Locate local maxima points.
       - Define a region for each such maxima point.
3. Concatenate nearby maxima points from neighboring
   slices into color lines.
```

Constructing the *histogram slices* is done by slicing the *RGB* histogram using hemispheres of equal distance radii centered at the origin. Each *histogram slice* is the summation of all histogram points between two successive hemispheres. Slice i is therefore a 2D histogram of all image points with their *RGB* norm between N_i and N_{i+1}, where N is a vector of equal distanced norms. Figure 7 demonstrates the construction of the histogram slices. Assuming that the color clusters are roughly perpendicular to the hemispheres, we get local maxima in the *histogram slices* wherever an elongated color cluster intersects the slice. We smooth the histogram slices using anisotropic diffusion and then we find the local maxima points (we simply find points that are larger then their neighbors). We then merge nearby local maxima points according to their distance and a separation measurement (the depth of the valley connecting them). For each maxima point we fit a gaussian. These gaussians are used for pixel classification. Although the shape of the 2D color cluster in the histogram slice can be general, in practice it is usually ellipsoid and modelling it using a gaussian yields good results. Using gaussians also helps in introducing a probabilistic framework for color classification that can be later used for segmentation purposes. We finally combine nearby local maxima from neighboring slices to form color lines.

The algorithm is very robust and almost doesn't require any threshold tuning. Changing the threshold creates a tradeoff between the number of color lines found and the mean distance between actual pixel color and the color lines. The parameters we use are: histogram slices smoothing, a parameter that controls the merging of nearby local maxima points, and a parameter for deciding whether two maxima points from nearby slices belong to the same color line. Although there are algorithms in the literature that are aimed on finding general elongated clusters [4], we so far achieved the best results using our heuristic algorithm. We hope to incorporate an algorithm with a better theoretical background for the *Color Lines* computation in the future.

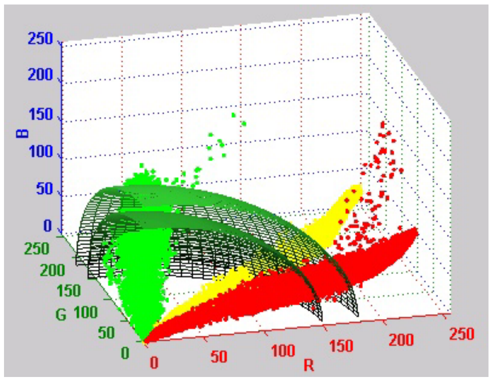

Figure 7: Histogram slicing.

While this approach does not utilize spatial information, it makes strong use of our prior knowledge of the world and in practice gives good results even for very complicated scenes. We would like to mention that in fact, considering the extreme case, when the entire *RGB* cube is considered as one slice, this approach would be identical to performing the segmentation in the 2D Nrgb color space. Nevertheless, slicing the histogram makes a much better usage of locality in the histogram domain.

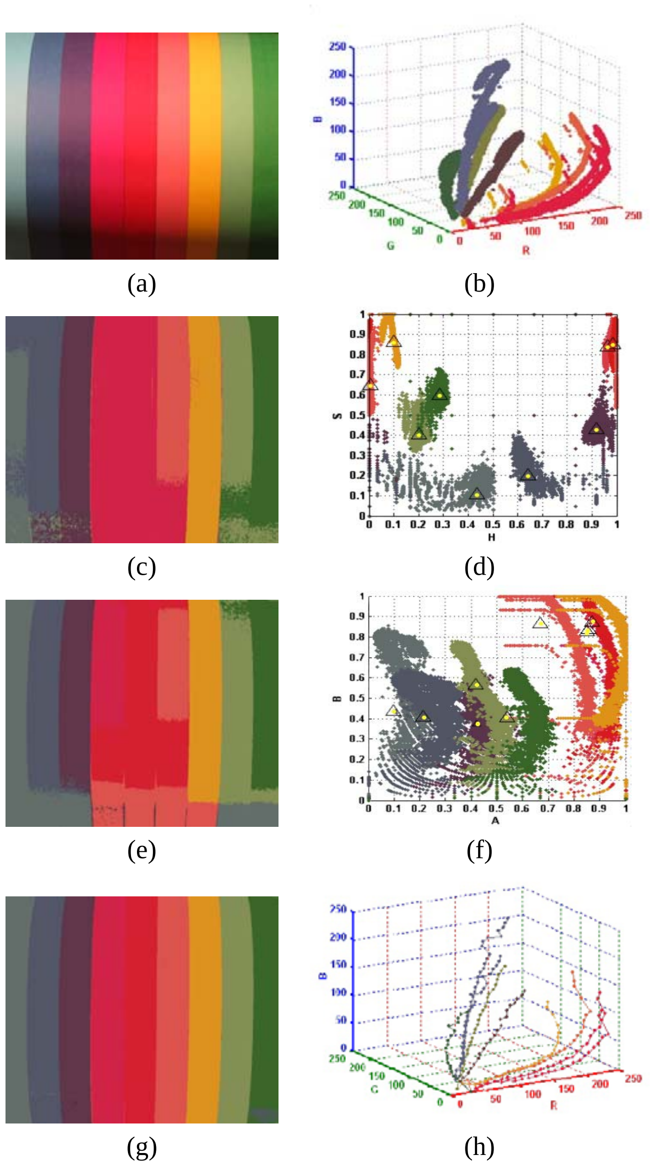

Figure 8: In figure (a) we see the original image, the histogram is provided in figure (b), figures (c) & (d) show the classification according to the HSV color model and the clusters in the HS plane (cluster centers are marked using triangles), figures (e) & (f) show the classification according to the CIE-LAB color model and the color clusters in the AB plane, figures (g) & (h) show the classification according to the Color Lines model and the color lines found.

4. Results

In order to demonstrate how well the image specific *Color Lines* model represents color compared to the HSV and CIE-LAB generic color models we begin by presenting a case study. We took an image of color stripes and manually segmented it into its color components. For each color component we built a color cluster representation in each of the color models: a color line in the case of our model and a 2D color point in the case of the HSV and CIE-LAB color models. We finally assigned each pixel of the original image to the nearest color cluster representation. For the HSV and CIE-LAB color spaces we used the 2D euclidean distance between the pixel's color and the cluster representation. For the *Color Lines* model, we used 3D euclidean distance between the pixel's *RGB* coordinates and the color line. This is in fact a slightly weaker version of our *Color Lines* model, since in our model we assume the color cluster is non-isotropic around the color line and use 2D gaussians around the color points for pixels classification. However, for fair comparison with the HSV and CIE-LAB color models we used the Euclidean distance metric. The results can be seen in figure 8

We used the Berkeley image segmentation dataset and the Corel image dataset to evaluate our color model. It is clear from our results that the histogram of a natural image can be described well using a small set of color lines. Figure 9 shows the distance histogram between the original pixels' colors and the color line best describing the pixel. The mean distance is 5 gray (or color) levels. We should mention that the color information in many of the images in the dataset is very poor due to strong compression. We therefore sometimes get artifacts that are clearly the result of JPEG compression as can be seen in figure 10. In spite of the above, for the large majority of images, our algorithm perform very well.

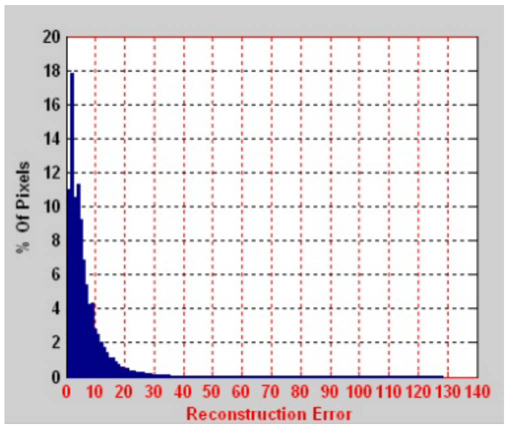

Figure 9: Distances histogram.

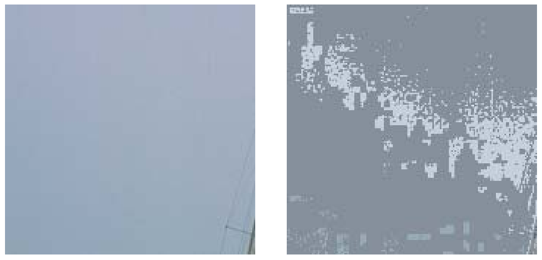
(a) (b)

Figure 10: (a) Part of a compressed image and (b) the JPEG artifacts it produces (the blocky pattern)

In the Corel dataset, images are divided into folders based on categories. In our version each category's folder contains 20 images. We recovered color lines for 30 different categories. The results are presented in a table and shown in Figure 1. For each category we provide the average number of lines found and the average reconstruction error. It is possible to see that both the average number of color lines and the mean reconstruction error is affected by the type of image. Landscape images can be usually described accurately using a small set of color lines, while highly detailed ones require a larger number of lines. Ironically, our algorithm did worst for the "color" folder, but this is not surprising, since most images there are either synthetic or contain a large amount of reflections. Figure 11 shows an example of a simple image and a complicated one, two more examples are shown in Figure 15. ***Many other examples are found in the CD***

5. Applications

Using the *Color Lines* representation has advantages over generic color models for several applications. In this section we outline the use of *Color Lines* for a few applications. We did not implement complicated, state of the art systems, and in all our examples we only use color information (and no spatial information, texture or edges).

Segmentation Integrating the color lines model into a segmentation application is very straight forward. The *Color Lines* provides us with a set of the dominant colors in the image and the probability of assigning every pixel to each of the color clusters. Assigning pixels to their closest color lines by itself can't provide good segmentation since it does not use any spatial information or other important features like texture. but in many cases, even this simple approach yields good results as shown in figure 12.

Compression By using *Color Lines* we create a compact representation of an image. For each pixel only two values are stored, an index to it's color line and a parameter along the line (or intensity). This representation can be easily compressed later on.

folder	number of lines	mean error
africa	34.8	2.23
agricltr	33.3	2.49
animals	32.1	2.61
architct	30.7	2.25
birds	36.0	2.34
boats	28.4	2.45
buisness	35.0	2.19
cars	42.5	1.97
children	50.6	2.37
cityscpe	30.0	2.36
coasts	28.8	2.45
color	83.3	2.63
design	34.0	2.40
flowers	52.3	2.40
food	59.2	2.54
industry	40.9	2.28
insects	51.5	2.00
landscpe	27.2	2.51
leisure	48.4	2.67
mideast	36.1	2.75
mountain	24.5	2.25
patterns	21.9	2.40
people	55.6	2.40
pets	39.6	2.16
portrait	39.9	1.98
seasons	42.5	2.75
sports	49.4	2.19
sunsets	35.5	2.06
textures	19.4	2.57
tourism	41.7	2.67
AVG.	39.5	2.38

Table 1: Corel Data Set Color Lines statistics.

Color editing Using *Color Lines* enables us to manipulate color very efficiently and in a very intuitive way, as shown figure 13. We can increase or decrease the color saturation of an object, or even completely replace colors by applying simple transformation upon the color line of the object.

Saturated color correction Another *Color Lines* application is correcting the color of saturated image pixels. The dynamic range of a typical natural or artificial scene is usually larger than the dynamic range that the camera's sensors can capture. As a result, in many pictures some of the pixels have at least one saturated color component (people are often not sensitive to this). In the histogram domain, this phenomenon appears in the form of a knee in the color cluster's line, the line looks as if it has been projected upon the *RGB* bounding box. We correct the saturated component by substituting one of the non saturated color components in the line equation of the non saturated line segment and retrieving the saturated component. (we can even use one non saturated component to correct the other two).

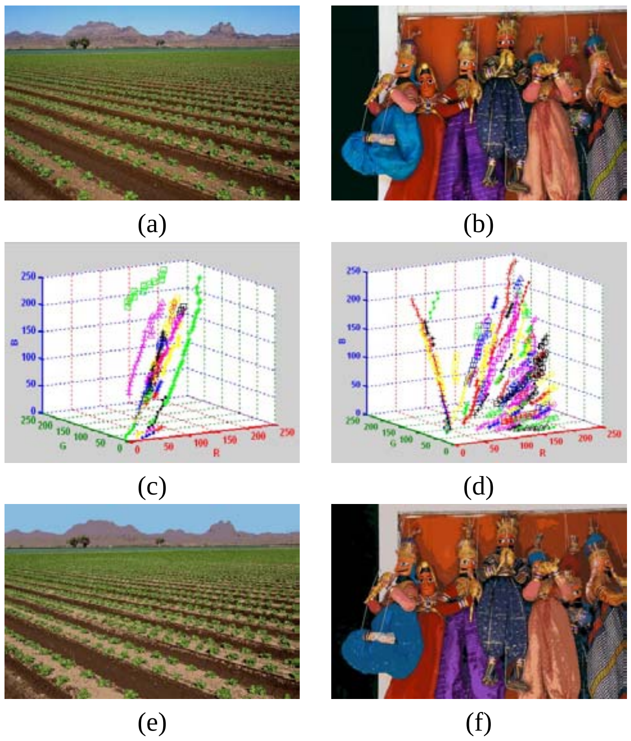

Figure 11: (a) is a simple image, it's color lines model (30 color lines) is shown in (c), and the pixels classification is shown in (e). (b) is a complicated scene with a lot of specularity and reflectance, it's color lines model is shown in (d) (92 lines found) and (f) shows the pixels classification

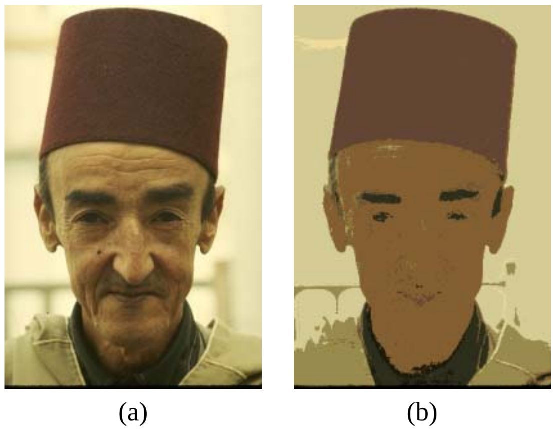

Figure 12: (a) Original image and (b) the image after assigning the color of each pixel to the mean color of its color line.

The color correction results are shown in figures 14. In order to readjust the dynamic range we use gamma correction or other methods for high dynamic range compression [6]. Simply rescaling the color will usually create an image that is significantly darker than the original image and therefore yields poor results.

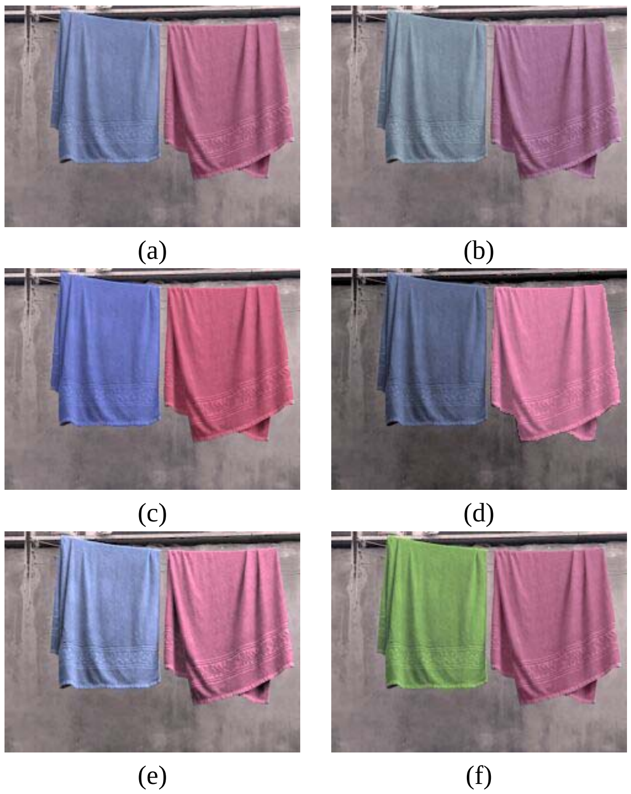

Figure 13: (a) Original image (segmented into 3 color lines). (b) Decreasing the color saturation. (c) Increasing the color saturation. (d) Shifting the intensities making one object brighter and the others darker. (e) Stretching the lines. (f) Changing colors.

6. Summary

We presented a new approach of modeling color in an image specific way. We have shown the advantages of this approach over existing color modelling methods that are generic and do not take into account color manipulation and distortion in the scene and in the image capturing process. Our approach doesn't assume any particular color distortion and only assumes that a region in the scene having homogeneous color will form a homogeneous elongated color cluster in the *RGB* histogram with the brighter pixels having larger *RGB* norms. This work is different from previous works like the T-shape model since it doesn't model a single phenomena but the entire color of an image and by the fact that it does not assume the shape of the resulting color clusters. By using *Color Lines* we describe the elongated clusters in the *RGB* histogram and create a precise and useful color representation.

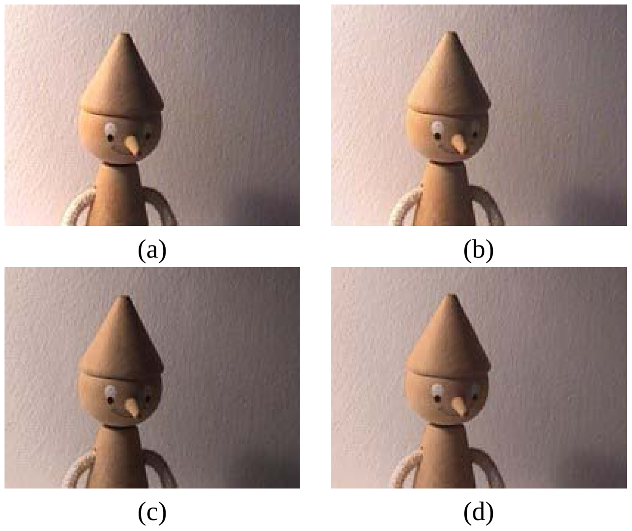

Figure 14: (a) Saturated image. (b) Using gamma correction. (c) Correcting saturated pixels and rescaling the colors. (d) Correcting saturated pixels and using gamma correction. It is possible to see that in figures (c) and (d) the saturated (yellowish) pixels in the left part of Pinocchio are corrected but the intensity range has increased from 255 to 305 and the image in (c) is too dark. The intensity in image (d) has been corrected using gamma correction.

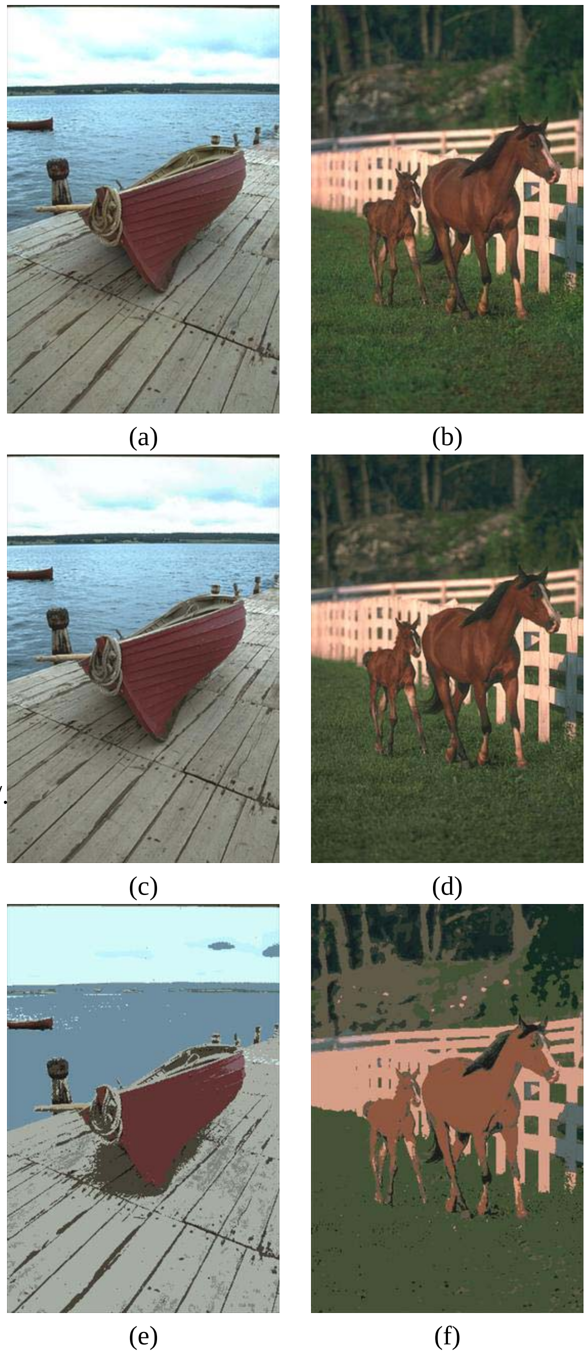

Figure 15: The original images are shown in figures (a) and (b). Figures (c) and (d) show the images after projecting the color of each pixel upon its color line. In Figures (e) and (f) the pixels are assigned to the mean color of their color lines. The number of Color Lines found for each of the images is 23.

References

[1] Berkeley segmentation dataset. http://www.cs.berkeley.edu/projects/vision/grouping/segbench/.

[2] Cie web site. http://www.cie.co.at/cie/index.html.

[3] Color faq. http://www.inforamp.net/∼poynton/ColorFAQ.html.

[4] J. Gomes and A. Mojsilovic. A variational approach to recovering a manifold from sample points. In *ECCV (2)*, pages 3–17, 2002.

[5] M. D. Grossberg and S. K. Nayar. What is the space of camera responses? In *Computer Vision and Pattern Recognition*.

[6] R. F. D. Lischinski and M. Werman. Gradient domain high dynamic range compression. In *Proceedings of ACM SIGGRAPH*, pages 249–256, July 2002.

[7] Y. Ohta, T. Kanade, and T. Sakai. Color information for region segmentation. *Computer Graphics and Image Processing*, 13(3):222–241, July 1980.

[8] G. K. S. Shafer and T. Kanade. A physical approach to color image understanding. *International Journal of Computer Vision*, 1990.