International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 3 (2011), pp. 812–819

Recovering Color and Details of Clipped Image Regions

Elhanan Elboher[1] **and Michael Werman**[2]

[1]The Hebrew University of Jerusalem, School of Computer Science,
Givat Ram Campus, Jerusalem 90914, Israel
elhanane@cs.huji.ac.il

[2]The Hebrew University of Jerusalem, School of Computer Science,
Givat Ram Campus, Jerusalem 90914, Israel
werman@cs.huji.ac.il

Abstract: **In a picture of a bright scene, camera sensor readings may be saturated to a maximal value. This results in loss of variation and color distortion of bright regions in the image. We present an algorithm that exploits data from uncorrupted channels to recover image details and color information. The correction is based on the *Color Lines* model for image representation in RGB color space. Given an image we identify regions of color clipping, and expand each of them to a color cluster called a Color Line. Then we restore the clipped pixel values of each Color Line according to its unique properties. Our method is suitable for both raw and processed color images, including those with large clipped regions. We also present results of correcting color clipping in video sequences.**

Keywords: color correction, color clipping, sensor saturation, computational photography.

I. Introduction

Color clipping decreases the quality of pictures and movies. Pixel values are clipped at a maximal value, and the variation in bright regions is lost, as can be seen in Figure 1. The color of these image regions is also distorted, because of the change in the RGB pixel values. In the extreme case, when all of the color channels become are clipped, a bright colorful region becomes white.

There are two main reasons for color clipping. First, camera sensors have a finite dynamic range that may become saturated. In this case the sensor reading only constitutes a lower bound on the desired true value, which would have been measured if the sensor had a larger capacity.

Second, in the image processing pipeline, non-linear color enhancement functions reduce the differences between high color levels in comparison to middle color levels that remain more distinguishable. This effect is incorporated along with other processing steps that change the pixel values and make it harder to determine which image pixels are saturated. Actually, in many cases a range of the high levels in the output image (e.g. 240–255) should be treated as saturated. Figure 2 illustrates the effect of such distortions in RGB space.

In many cases, however, the clipped region itself contains useful information to help recover color and pixel variation. Figure 3 presents an example in which the details were eliminated from the Green channel, whereas the Blue channel preserves them completely. It is natural to make use of the information of the Blue channel in order to recover the pixel variation in the Green channel.

Different approaches have been suggested to recover the clipped information. Wang et al. [9] transfer details from an under-exposed region to a corresponding over-exposed one by user annotations. This method is appropriate for the completion of texture regions and is based on the assumption that the image contains patches similar to the clipped region. Didyk et al. [10] enhance the clipped region using clues from neighboring pixels. This method is specific for highlights and does not work with diffuse surfaces.

Regincos-Isern and Batlle [1] captured a series of images with different light intensities, and assigned the hue and saturation values of saturated (r,g,b) triplets according to the non-saturated ones. This method was useful for their goal, training a system for segmentation of a specific color from an image but it cannot be used as a general framework without knowing which colors have been distorted. Nayar et al. [3] also use varying exposures to construct a high dynamic range image that contains no clipped values. This method depends on a special hardware filter incorporated in the camera.

Zhang and Brainard [7] suggested a method for raw images that is based on the correlation of the RGB color channels. They model the joint distribution of the RGB values as a Gaussian and estimate its parameters (mean and covariance matrix) from the unclipped image pixels. The unclipped channels of a pixel are used as an observation in order to efficiently compute the expected value of its clipped channel. However, in the common case of different color regions in an image, a global model is not suitable. In practice, the real-world examples in their paper show only reconstructions of the variation in gray/white image regions.

Masood et al. [11] proposed a method to correct saturated images based on propagation of R:G:B ratio from unclipped pixels to their clipped neighbors. The authors show that their results are better than applying the colorization technique of

(a) Clipped (b) Clipped (c) Clipped

(d) Corrected (e) Corrected (f) Corrected

Figure. 1: Examples of clipped images and their correction. The most relevant regions are surrounded by ellipses. The first row contains clipped images, compared to those in the second row that are corrected by our algorithm. The original images were scaled linearly to the range of the corrected images. Note the appearance of details in (e,g) and the correction of Blue in (f). (d,h) illustrate the correction of Color Lines in the RGB space, by presenting the R:G histogram of (b,f) respectively.

Levin et al. [5] on the clipped pixels. They focused almost completely on face images and present results only on raw images (except a single non-raw example). In Section IV-B we compare our results with this work.

Recently, two papers proposed correcting processed (non-raw) images of general objects using a similar approach to [11]. Xu et al. [14] first correct the chroma of clipped pixels in YCbCr space, and then return to the RGB space to recover intensity. Their color correction is based on propagation of one of the Cb or Cr channels of the unclipped neighboring pixels. In the second step the image is converted back into RGB for the recovery of intensity. In another paper, Guo et al. [13] recover the intensity and the color information separately in La^*b^* space. Their color correction is based on probabilistic propagation of the a^*, b^* values of neighboring pixels. Both [14] and [13] presented only results of correcting color distortions in smooth and diffuse regions. They did not show results of recovering details and did not treat specularity.

A preliminary version of this paper appeared in [12]. It is based on a model suggested by Omer and Werman [6] for representing a color image in the RGB space, called *Color Lines*. This model makes it possible to handle each color region separately according to the physical properties of its representing Color Line. Although the authors proposed to use Color Lines for correction of clipped regions, they did not present a method that performs such correction. (Figure 14 in their paper shows a single example for concept demonstration. By personal communication, the input is a raw image that was manually corrected.)

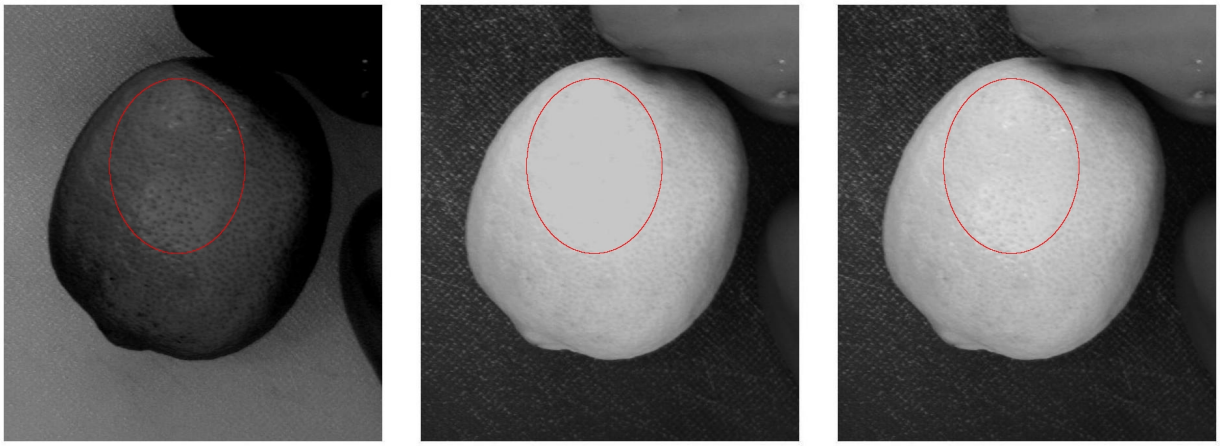
This paper presents a practical automatic algorithm which recovers color information and intensity variation of clipped image regions. The proposed method works with raw images, processed images and video.

We do not deal with the visualization of the recovered high dynamic range image. Our results are presented using simple linear scaling. For more sophisticated methods see e.g. [4].

The remainder of this paper is organized as follows. In Section II we review the Color Lines model and its advantages for color clipping correction. Section III describes our algorithm in detail. Section IV presents our results for raw image data, processed images and video sequences. Section V summarizes our work.

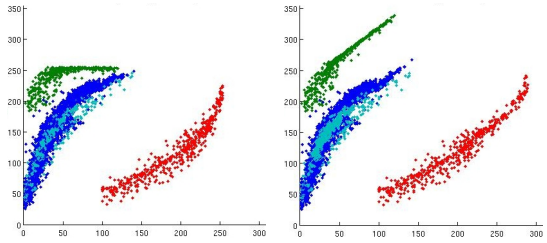
II. The Color Lines Model

It is common to separate color from intensity to decide if two image pixels share the same color. This separation can be linear as in YCrCb, YUV, YIQ color spaces, or non-linear as in HSV, HSI, CIE-LAB, CIE-LUV color spaces; see e.g. [8]. All these color models can be referred to as generic, i.e. they



(a) Blue (unclipped) (b) Green (clipped) (c) Green (corrected)

Figure. 3: An example of the RGB correlation. The unclipped Blue channel (a) preserves details that are eliminated in the clipped Green channel (b). This information was used for the correction shown in (c).



(a) R:G, clipped (b) R:G, corrected

Figure. 2: R:G histogram of the clipped image 1(b) and the corrected image 1(f).

(a) Input image (b) Saturated blobs (initial seeds) (c) Color Lines (segments)

Figure. 4: Illustration of the first two steps of the algorithm (Sections III-B, III-C). Given a clipped image (a), we identify *saturated blobs* (b) and expand them to *Color Lines* (c).

do not consider specific image properties.

In practice, however, most images undergo several types of color distortion caused by reflection, noise, sensor saturation, color enhancement functions etc. Thus, it is reasonable to develop image specific color models, since each image is changed differently.

Although distorted, the RGB histogram of real world images remains very sparse and more importantly, it is structured; i.e., most of its non-empty histogram bins have non-empty histogram neighbors.

Omer and Werman [6] combine this observation with another assumption on the histogram structure - that the norm of the RGB color coordinates of a given real world color increases with an increase of the illumination. Therefore they suggest representing the color distribution of a specific image by a set of elongated clusters called *Color Lines*. An example is shown in Figure 2(a).

In the context of the problem of color clipping, the Color Lines representation is powerful since it makes it possible to attribute saturated pixels to the appropriate color cluster. Moreover, since the increase in the RGB channels are correlated with the increase in illumination, the data from a non-saturated part of a color cluster provide useful information that can help to correct its saturated part. Considering a clipped channel as a monotonic increasing function of the unclipped channels, we can recover the pixel intensity and chromaticity.

III. Correction of Color Clipping

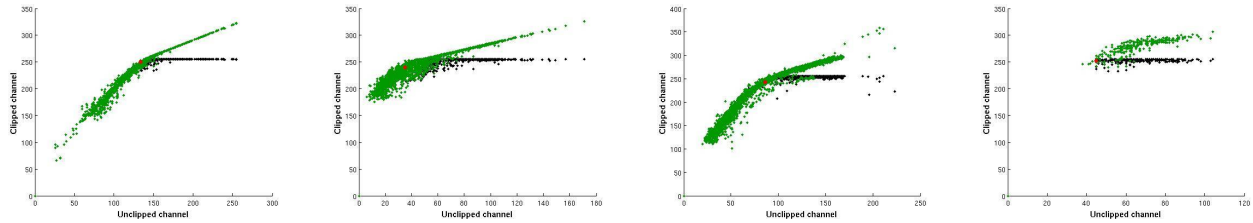
Our algorithm includes three main steps: (1) identification of color clipping; (2) expansion of clipped regions to Color Lines; (3) correction of the clipped channels of a Color Line (one or more), using information from its other channels. We add a post-processing step to fix artifacts that occur in white image regions. This step is optional and is especially suitable for specularities.

The process is carried out in RGB color space. This is natural since the camera sensors capture the visual information in RGB and the clipping occurs in the separate RGB channels.

A. *Identifying color clipping*

The first step identifies regions of color clipping in the image. The purpose of this step is to roughly localize clipped color regions that should be corrected, therefore missing part of the saturated pixels is allowed. It also separates clipped pixels from different color regions and forms saturated blobs that constitute a beginning point for the identification of the relevant image Color Lines.

Simple thresholding is not sufficient, however, because of the distortion in the high intensity levels. As described in Section I, it is difficult to determine an exact threshold. Therefore we operate as follows.



(a) 'Clean' correction (b) Noisy case (c) Noisy case (d) No unclipped part

Figure. 5: Correction of clipped Color Lines. After identifying the saturation point (marked in red), the original pixel values (black points) are replaced by new values (green points) according to 3D line equation, as described in Section III-C. For visualization we show a 2D plot of one unclipped channel against the corrected channel. Note that the green points between the clipped values are the same as the original points.

(a) Clipped image (b) Before white correction (c) After white correction

Figure. 6: Treating artifacts in white pixels (Section III-D). The clipped image (a) was corrected in (b). This recovered the blue color of the car (red ellipses) but colored white and metallic regions (black ellipses). The post processing step fixed this artifact, without damaging true correction of colorful regions.

Initially we only take into account pixels whose maximum RGB value is greater than t ($\sim$230). Then we apply several iterations ($\sim$5) of bilateral filtering [2] to those pixels in order to denoise them. After filtering, we compute the gradient and remove the pixels with the 10% highest gradient norms (edges). This results in connected components that we call saturated blobs, as can be seen in Figure 4(b). In order to be efficient we remove small blobs (smaller than 50 pixels). Typically, this leaves us with around 0–30 different blobs.

B. Color Lines expansion

After finding the saturated blobs, we find the Color Lines associated with them, as shown in Figure 4(c). We consider each blob from the previous step and iteratively expand it to neighboring image pixels. The color clusters are smooth, elongated, and positively correlated. We iteratively expand them by joining neighboring pixels in XY-RGB (i.e. spatial image coordinates and RGB values) until no new pixels are added.

We fuse the clusters with significant overlap (more than 50% of each cluster). A pixel can be in more than one cluster, in such case it is corrected to the average of the values derived from the correction of each cluster.

C. Color Lines correction

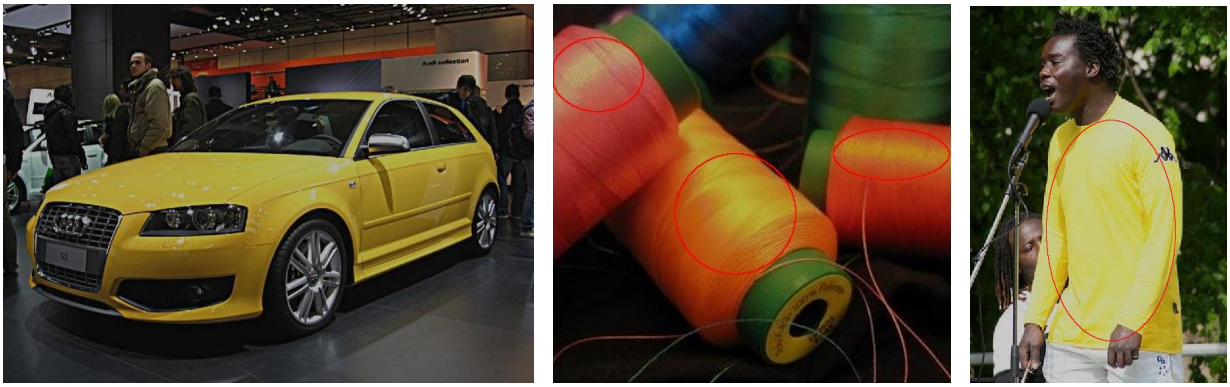
The last step in our algorithm corrects the clipped part of each color cluster according to its unclipped part. Actually, color clusters in processed images are noisy and nonlinear (see Figures 1(d,e) and 5); thus we consider only pixels that are close to the clipped region in the RGB histogram, and compute their R:G:B ratio to get a first-order approximation of the true values of the corrected channel.

As mentioned, the RGB channels of a color cluster are highly correlated. Thus we can treat each one of them roughly as a monotonic increasing function of any combination of the others. When a channel reaches the maximal value and is clipped, the increase in the other channels continues. Therefore, for each saturated channel there is a point that we call the *saturation point*, which separates the unclipped part of the color cluster from its clipped part, as shown in Figure 5. We find the saturation point as follows.

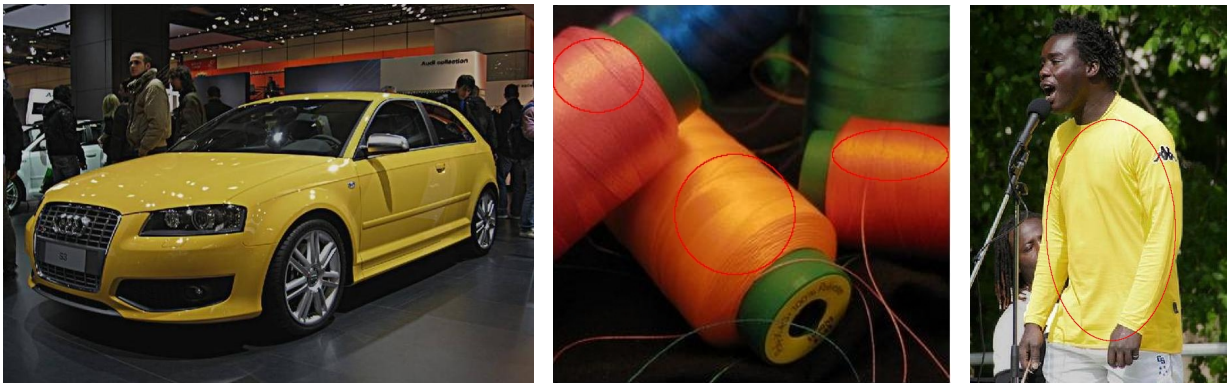
We threshold pixel values of a channel, for example R, by a high value ($\sim$250) and take only the pixels that are greater than it. Then we sort their values in the other channels (G,B) separately and take the values of a small percentile ($\sim$1%) in each of them as the (g, b) coordinates of the saturation point. The r coordinate is defined as the median R value of the pixels that are in a small radius of the (g, b) coordinates in RGB. Note that this value may be smaller than the initial threshold. In cases where the color cluster has more than one clipped channel, each clipped channel has a different breaking point and is corrected separately.

Pixels with a value greater than that of the saturation point are considered clipped in this channel and the others as non-clipped. This exempts us from determining an exact threshold, which may cause undesired discontinuity artifacts in the image.

In order to reconstruct the values of the clipped part of the Color Line, we need to estimate the slope of the unclipped



(a) Clipped (b) Clipped (c) Clipped



(d) Corrected (e) Corrected (f) Corrected

Figure. 7: More results of our method on processed images.

part in the RGB space near the saturation point. We approximate it using the first eigenvector of the scatter matrix of the unclipped pixels near the saturation point.

Combining the estimated slope a with the saturation point (s_r, s_g, s_b) we can compute the corrected r value of a pixel by its (g, b) values:

$$r_{corrected} = s_r + a \cdot \|g - s_g, b - s_b\| \quad (1)$$

However, in this way all of the pixels with the same (G,B) values are attributed the same R value unlike the situation in the original image (note that the original r may be smaller than the maximum value). Thus, we replace the above equation with

$$r_{corrected} = s_r + \rho(r) \cdot a \cdot \|g - s_g, b - s_b\| \quad (2)$$

where ρ is 1 when r is greater than an upper value, and decreases linearly to 0 when r is smaller than a lower value. In order to compute a robust solution we limit the angle of the line slope to be between an upper and lower value. Additionally, we limit the maximal intensity of the corrected pixel, to restrict the resulting dynamic range of the image.

Figure 5 shows typical examples of Color Lines correction. Figure 5(d) presents a case in which no unclipped part exists; namely, all of the pixels in the color cluster are clipped in some channel. In this case we cannot correct the color by restoring the original pixel values; however, we can still recover some of the variation between pixels by using the saturation point and a small arbitrary angle.

D. Treating pixels clipped in all RGB channels

There are two scenarios in which a pixel is clipped in all RGB channels. In case of a diffuse surface the true color is recovered by the above correction (Section III-C), even though the details cannot be restored. However, when the pixel is clipped due to specularity (e.g. as in Figures 1(a,c) and 7(a)), the correct color is often white, the typical ambient light. We add an optional post processing step to correct specular clipping. This step is relevant also for white diffuse regions, whose correction might be unbalanced, as color channels are corrected separately. This might result in a reddish or bluish hue. Figure 6(b) shows an example of coloring white regions incorrectly.

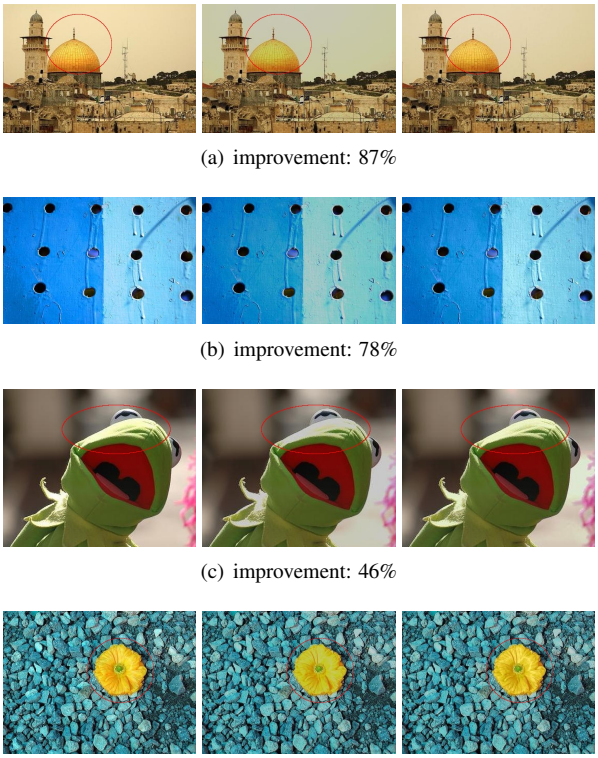
We solve this problem by modifying the additions to the RGB channels, as demonstrated in Figure 6. The output of the correction process is an image with a greater dynamic range than that of the input image. Thus, by subtracting the original pixel values from the corrected image, we can get an addition map and modify it in sensitive regions.

In order to handle white pixels we measure the L_2 distance from each pixel in the input image to the gray line $R = G = B$ to determine its whiteness $0 \leq w \leq 1$ (distance greater than 100 equals 0; zero distance equals 1). Then we project the addition toward the gray line, depending on relative whiteness:

$$\Delta_{rgb} = \frac{w}{3} \cdot I(\Delta_{rgb}) + (1 - w) \cdot \Delta_{rgb} \quad (3)$$

where Δ_{rgb} is the addition map and $I(x)$ the intensity of x.

As shown in Figure 6, the projection toward the gray line



(a) improvement: 87%

(b) improvement: 78%

(c) improvement: 46%

(d) improvement: 36%

Figure. 8: Examples of the experimental results (Section IV-A). The original images (left) were clipped to 200 (middle column) and restored by our method (right). We selected to show pictures with middle scores (from 87% to 36%), to demonstrate the improvement of such correction levels.

Table 1: Experimental results (Section IV-A).

saturation threshold (t)	#images	mean improvement	median improvement
180	551	44.89%	47.18%
200	849	43.72%	50.58%
230	357	46.33%	46.94%
245	301	54.09%	61.46%

returns white regions to their original color, without damaging the true correction of colorful regions. Since the intensity of the addition is not changed, white regions also become brighter, and some of their variation is emphasized.

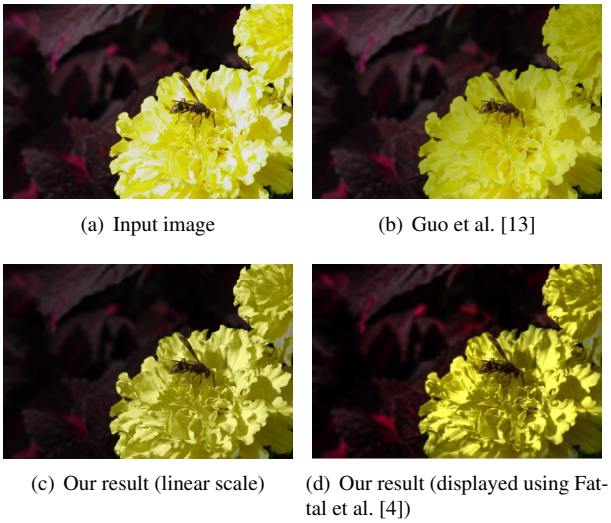
IV. Results

A. *Correction of processed images*

Due to the huge amount of media captured by cameras with low dynamic range, correcting processed (non-raw) images is crucial. Our results are demonstrated in Figures 1, 3, 6, 7 and 8. As can be seen, there is a significant improvement both in the intensity variation within the clipped regions, as well as recovering the correct color.

Note that the corrected images have an expanded dynamic range. The images were linearly scaled to [0..255] for display. This slightly darkens the images. However, one can choose other scaling functions depending on the importance of the high intensity levels, e.g. [4].

In order to provide a quantitative measure of this improvement, and demonstrate the generality of our method, we ran several experiments on a collection of images randomly picked from Flickr. The results of these experiments are presented in Table 1. In each experiment we clipped the images to a different threshold, and examined the correction of images meeting two conditions: (a) at least 15% of the pixel values in the image are clipped, and (b) at most 5% of the image pixels are clipped in all of the RGB channels. The first condition sifts out images without significant clipping, which are irrelevant for this experiment. The second eliminates cases where our method has no information to correct with.



(a) Input image (b) Guo et al. [13]

(c) Our result (linear scale) (d) Our result (displayed using Fattal et al. [4])

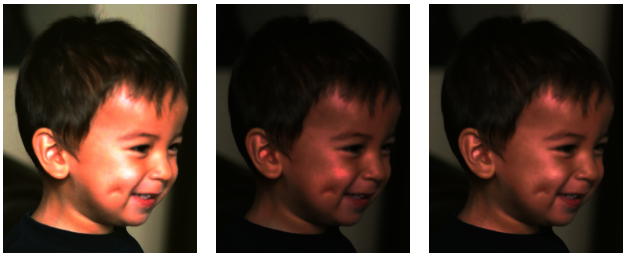
Figure. 9: Comparison with Guo et al. [13] (Section IV-A).

For each image we measure the improvement of the correction by $\frac{D_{01}-D_{02}}{D_{01}}$, where D_{01} is the sum of squared differences (SSD) between the original image and the clipped one, and D_{02} is the SSD between the original image and the corrected one. Thus, the improvement is positive if the corrected image is closer to the original image than the clipped one. A value of 1 is reached when the correction exactly restores all of the pixel values. In most cases, such an exact restoration is unattainable; however, as can be seen from the examples in Figure 8, values around 80% also reflect satisfactory correction, and even a value of 30% recovers at least some eliminated details.

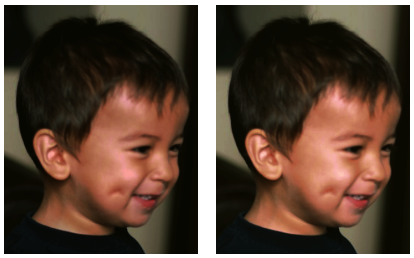
Figure 9 compares our method to Guo et al. [13] using an image from their paper. It can be seen that Guo et al. do not recover the clipped details, which results in low contrast. Figure 9(c) presents our result using linear scale. Different rescaling methods change the perception of color as demonstrated in Figure 9(d) using the method of Fattal et al. [4].

B. *Correction of raw images*

The main reason for color clipping is sensor saturation. Therefore, raw image data may also be clipped. The correction should be done at the beginning of the image processing pipeline, immediately after the first step of image demosaicing, which fills the RGB values of all of the image pixels. This avoids color artifacts that may be caused by later steps (e.g. white balancing). We apply our method as in processed images. Here the situation is even simpler than in processed



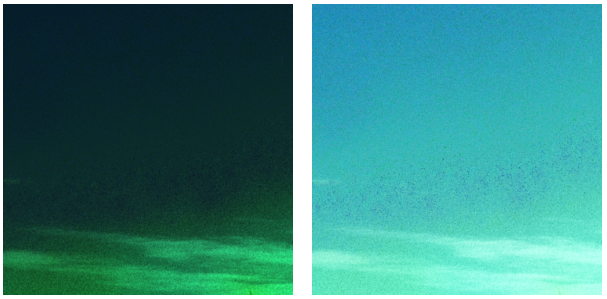
(a) Input image (b) Correction (Masood et al. [11]) (c) Correction (ours)



(d) Visualization (Masood et al. [11]) (e) Visualization (ours)

Figure. 10: Correction of raw face image. (b) shows the correction of Masood et al. [11], and (c) presents our correction. (b), (c) are both linearly scaled. (d), (e) show visual results accepted by non-linear scaling of (b), (c) respectively, as was done in [11].

(a) Input image (b) Correction (ours)



(c) Correction (Masood et al. [11]) (d) Visualization (Masood et al. [11])

Figure. 11: Correction of raw sky image. The clipped image (a) was corrected by our method (b) and by Masood et al. [11] (c). (d) was accepted by non-linear scaling of (c), as was done in [11].

images, since there are fewer steps that modify the pixel values and add noise.

Our results are presented in comparison with the method of Masood et al. [11] that was described in Section I. Figure 10 shows that ours performs as well as Masood et al. for face images, which are the focus of their work. (Our correction is slightly more accurate, e.g. in the region of the cheek). In other cases it seems that our method provides better results, as shown in Figure 11 for a clipped sky image. We made no comparison with Masood et al. on processed images, since they present only a single example of a linearized image.

C. Color clipping in video sequences

When treating video sequences, the main challenge is to achieve a stable correction. Here, a negative effect may be caused not only by a wrong correction but also from performing no action in some frames, while other frames are corrected. This results in flickering in the output sequence.

We applied our method to several video sequences, each one some tens of frames in length that contained moving bright colorful objects. Each frame was corrected independently. The results are available in the following link: `http://www.cs.huji.ac.il/~elhanane/clipping/index.html`.

The quality of the corrected sequences improved both in variation and color information. In general the correction was stable, except for a few cases of missing correction. However, in some movies two types of problems arose: (1) a missing correction because of erroneous segmentation (i.e., unsuccessful Color Lines expansion); (2) unstable correction of subsequent frames, i.e., adding different color levels to the clipped region (see Section III-C). The first difficulty requires the use of methods such as blob tracking and 3D segmentation. By contrast, the second problem is inherently caused by the Color Lines correction.

We overcome this difficulty by utilizing the information from a few sequential frames, so that the Color Lines representation becomes more stable. For each Color Line in the current frame we look to see which Color Lines from the few previous frames (5–10) have a significant overlap with it, and include their pixel values in the correction step. This has an influence on both the saturation point and the line slope that are used for the correction (see Section III-C). This makes the correction more stable both in its color and its intensity.

D. Failures

Wrong corrections can be caused be number of reasons, such as such as incorrect segmentation that fuse different color clusters, or misses clipped pixels. Figure 12 shows failure examples.

V. Conclusions

We presented a method that corrects the values of clipped pixels in bright image regions, and thus, restores both the variation and the color information that were distorted by color clipping. For the first time, our method automatically corrects not only raw data but also processed images, and is also applicable to video sequences.

(a) (b) (c)

Figure. 12: Failure examples. Fusing two different color clusters in (a) results in wrong correction (b). (c) Problem caused by the connectivity constraint (Section III-B).

References

[1] J. Regincós-Isern and J. Batlle. A system to reduce the effect of CCDs saturation. *Proceedings of International Conference on Image Processing*, volume 1, pages 1001–1004, 1996.

[2] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. *Proceedings of the Sixth International Conference on Computer Vision*, pages 839–846. Citeseer, 1998.

[3] S. Nayar and T. Mitsunaga. High dynamic range imaging: Spatially varying pixel exposures. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 472–479. IEEE, 2000.

[4] R. Fattal, D. Lischinski and M. Werman. Gradient domain high dynamic range compression. *ACM Transactions on Graphics*, 21(3):249–256, 2002.

[5] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23(3):689–694, 2004.

[6] I. Omer and M. Werman. Color lines: Image specific color representation. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 946–953. IEEE, 2004.

[7] X. Zhang and D. Brainard. Estimation of saturated pixel values in digital color imaging. *Journal of the Optical Society of America A*, 21(12):2301–2310, 2004.

[8] C. Poynton. Color faq. `http://www.poynton.com/notes/colour_and_gamma/ColorFAQ.html`.

[9] L. Wang, L. Wei, K. Zhou, B. Guo, and H. Shum. High dynamic range image hallucination. *Eurographics Symposium on Rendering*, pages 321–326. Citeseer, 2007.

[10] P. Didyk, R. Mantiuk, M. Hein, and H. Seidel. Enhancement of bright video features for HDR displays. *Computer Graphics Forum*, volume 27, pages 1265–1274. Blackwell Science Ltd, Osney Mead, Oxford, 2008.

[11] S. Masood, J. Zhu, and M. Tappen. Automatic Correction of Saturated Regions in Photographs using Cross-Channel Correlation. *Computer Graphics Forum*, volume 28, pages 1861–1869. John Wiley & Sons, 2009.

[12] E. Elboher and M. Werman. Recovering Color and Details of Clipped Image Regions. *IADIS Conference on Computer Graphics, Visualization, Computer Graphics and Image Processing*, pages 124–132, 2010.

[13] D. Guo, Y. Cheng, S. Zhuo and T. Sim. Correcting over-exposure in photographs. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 515–521, 2010.

[14] D. Xu, C. Doutre and P. Nasiopoulos. Correction of Clipped Pixels in Color Images. *IEEE Transactions on Visualization and Computer Graphics*, 17(3):333–344, 2010.

Author Biographies

Elhanan Elboher received his B.Sc. degree in computer science and life science and the M.Sc. degree in computer science from the Hebrew University of Jerusalem, Israel, in 2007 and 2009, respectively. He is currently a Ph.D. student in computer science at the Hebrew University of Jerusalem. His research interests include image processing and computer vision.

Michael Werman Ph.d. 1986, The Hebrew University, currently professor of computer science at the Hebrew University. His research has is mainly in designing computer algorithms and mathematical tools for analyzing, understanding and synthesizing pictures.