Journal of Imaging Science and Technology® 55(4): 040503-1–040503-10, 2011.

Spatio-Temporal Retinex-Inspired Envelope with Stochastic Sampling: A Framework for Spatial Color Algorithms

Øyvind Kolås
Gjøvik University College, N-2802 Gjøvik, Norway and Intel Corporation, London EC3N 1DD United Kingdom

Ivar Farup
Gjøvik University College, N-2802 Gjøvik, Norway
E-mail: ivar.farup@hig.no

Alessandro Rizzi
Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano, Via Bramante, 65 26013 Crema (CR), Italy

Abstract. *The authors present a new framework for algorithms for a wide range of image enhancement and reproduction applications, named* STRESS*—Spatio-Temporal Retinex-inspired Envelope with Stochastic Sampling. The algorithms work by recalculating each pixel using envelopes for local upper and lower bounds in the image. The envelopes are obtained sampling neighbor pixels and can be interpreted as local reference maximum and minimum. This approach derives from a computational simplification of previous spatial color algorithms like Retinex or* ACE*. With the proposed method, various tasks such as local contrast stretching, automatic color correction, high dynamic range image rendering, spatial color gamut mapping, and color to grayscale conversion can be performed with good results. The algorithm exhibits behaviors in line with some aspects of the human visual system, e.g., simultaneous contrast.*

[DOI: 10.2352/J.ImagingSci.Technol.2011.55.4.040503]

INTRODUCTION

Our vision system collects the meaningful information to produce its final perception, not from the stimulus coming from each single point in the scene, but rather from the spatial relationships among various stimuli.[1] This is at the origin of several well-known visual effects including local contrast enhancement, simultaneous contrast, etc. As a direct consequence of this, with the same stimulus, properly arranged in the space, we can form nearly all possible color sensations.[2]

One of the earliest models able to deal with locality of perception is Retinex, as presented by Land and McCann.[3] The scientific community has continued to be interested in this model and its various applications, as reported in Ref. 4. In the basic implementation of Retinex by Land and McCann, locality is achieved by long paths scanning across images. Different implementations and analysis have followed after this first work. These can be divided into two major groups and differs in the way they achieve locality. The first group[5–10] explores the image using paths or extracting random pixels around the pixel in question. The second group[11–16] computes values over the image with convolution mask, distance weighting, or variational techniques. All the sampling implementations use a high number of samples in order to lower the amount of noise.

A recent implementation, in order to investigate the effects of different spatial samplings, replaces paths with random sprays, i.e., two-dimensional (2D) point distributions across the image, hence the name "Random Spray Retinex" (RSR).[17] The random sprays replaced the paths with some advantages, but still the required number of sampled points is very high. A high number of points mean long computational time, which has been always the weak point of this family of algorithms.[18]

An edge-preserving version of Retinex was proposed by Sobol[19] for high dynamic range images. The edge-preserving behavior was obtained by introducing a ratio modification operator. Shaked and Keshet have proposed to use envelope operators for Retinex.[20] In order to obtain a fast implementation, the envelopes were represented as 2D infinite impulse response filters.

In this article, we present an alternative technique implemented with an extremely small number of sample points, using two envelopes to characterize the local visual context. The envelopes are two signals: E^{max} and E^{min}, which are constructed such that the image signal is always between the two (see Figure 1 and "The STRESS Framework" section). The envelopes are calculated using stochastic sampling technique that is a simpler alternative to the approach in Ref. 20. By using a simple weighting of the sample values, an edge-preserving method is obtained without the need of introducing any ratio modification operator as in Ref. 19. The properties of the proposed approach are in line with other spatial color algorithms (SCA).[18] The algorithm

Received Sep. 9, 2010; accepted for publication Mar. 7, 2011; published online Jul. 28, 2011
1062-3701/2011/55(4)/040503/10/$20.00.

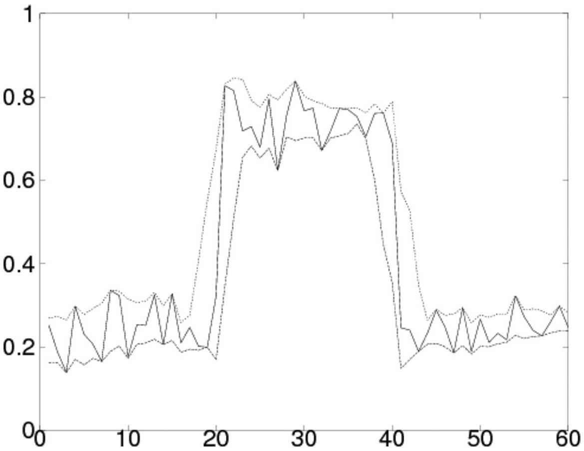

Figure 1. Illustration of the envelopes of one scan-line of an image. We see the scan-line (blue), $E^{\min}$ (green), and $E^{\max}$ (red). Notice that the exact shape of the envelopes will depend not only on the single scan-line but also on the content of the whole image due to the intrinsic 2D properties of the image.

framework is called STRESS: Spatio-Temporal Retinex-inspired Envelope with Stochastic Sampling. STRESS cannot be considered as one of the many Retinex implementations. It inherits from Retinex the idea of local white reference, but it implements a very different pixel value stretching. In contrast with ACE, it is based on linear averages and sampling only a few pixel values in each iteration.

The structure of the article is the following: First, the STRESS framework is presented in "The STRESS Framework" section. Then, its details and properties are demonstrated through a set of image processing algorithms derived from the STRESS framework. Finally, the STRESS parameters and its behavior are discussed.

THE STRESS FRAMEWORK

Basic Idea

By the human visual system, a relatively bright detail in a very bright part of an image can appear darker than a darker detail in a very dark part of the same image. The central part of the STRESS framework is to calculate, for each pixel, the local reference lightness and darkness points in each chromatic channel. This is done through calculating two envelope functions, the maximum and minimum envelopes, completely containing the image signal. The envelopes are slowly varying functions, such that the image signal is always in between the envelopes or equal to one of them.

In calculating the envelopes (that serve as the local reference maximum and minimum values), the most nearby parts of the image have the strongest influence on the envelopes at that point. In agreement with Rizzi et al.,[21] this dependency should be related to the distance. In order to avoid checking all image pixels in recomputing every pixel, resulting in an $O(N^2)$ algorithm, stochastic sampling will be used as in Ref. 17. All computations are assumed to be performed in a space that is close to perceptually uniform. In practice, this means, at least, that gamma corrected images are used. Since the gamma correction work more or less as a logarithm, this means that the pixel differences involved in the calculations are similar to lightness ratios. However, this is not mandatory. The algorithm shows goods results on linearly scaled images as well.

Formal Definition of the Envelope Computation

For each pixel, p_0, the values of the maximum and minimum envelopes, $E^{\max}$ and $E^{\min}$ at the corresponding positions, are computed in an iterative manner using N iterations. In every iteration, M pixels intensity values $p_j, i \in \{1, \dots, M\}$, are sampled at random with a probability proportional to $1/d$, d being the Euclidean distance in the image from the sampled pixel to the pixel in question. The intensity value of the center pixel, p_0, is not eligible for random sampling but is always included in the sampled set. The pixels are sampled only from a disk with radius R around the center pixel. When using such a random spray to sample the image, the strategy we have chosen when a sample outside the image is attempted is simply to try again until a sample within the image is found. From these samples, the maximum and minimum samples in the spray are found:

$$s_i^{\max} = \max_{j \in \{0,\dots,M\}} p_j, \tag{1}$$

$$s_i^{\min} = \min_{j \in \{0,\dots,M\}} p_j. \tag{2}$$

Since p_0 is always one of the sample points, $s_i^{\max} \le p_0 \le s_i^{\max}$ always. The range r_i of the samples and the relative value v_i of the center pixel are then given as

$$r_i = s_i^{\max} - s_i^{\min}, \tag{3}$$

$$v_i = \begin{cases} 1/2 & \text{if } r_i = 0, \\ (p_0 - s_i^{\min})/r_i & \text{else.} \end{cases} \tag{4}$$

Thus, $v_i \in [0, 1]$ always. These quantities are averaged over the N iterations in order to get a better estimate:

$$\bar{r} = \frac{1}{N} \sum_{i=1}^{N} r_i, \tag{5}$$

$$\bar{v} = \frac{1}{N} \sum_{i=1}^{N} v_i. \tag{6}$$

Averaging r_i and v_i instead of averaging $s_i^{\max}$ and $s_i^{\min}$ directly makes sure that the algorithm is edge-preserving and does not introduce haloing artifacts. If $s^{\max}$ and $s^{\min}$ had been averaged directly, too much weight would have been given to pixels with pixel values distant from the value of the center pixel. Averaging v_i instead, the inverse of the sample range $1/r_i$, acts as a weighting factor, giving more weight to collection of samples with a narrow range. Thus, pixel collections where all pixels come from the same side of some nearby edge are given higher weight. The envelopes are finally computed from the estimated average range, the average value, and the pixel value as follows:

$$E^{\min} = p_0 - \bar{v}\bar{r}, \tag{7}$$

$$E^{\max} = p_0 + (1-\bar{v})\bar{r} = E^{\min} + \bar{r}. \tag{8}$$

Since $v_i \in [0,1]$, also $\bar{v} \in [0,1]$ always. Thus, we are sure that $E^{\min} \leq p_0 \leq E^{\max}$. We also notice that with this definition $E^{\max} = p_0$ at the global maximum of the image and $E^{\min} = p_0$ at the global minimum. It is also possible but not necessary that the envelopes touch the local extrema. How closely the envelopes will follow the image will depend on the choice of R, N, and M. An illustration of the envelopes for one scan-line of an image is shown in Fig. 1. For color images, this is done three times separately for each chromatic channel.

Implementation Issues

There are different possible ways to perform the random sampling of the pixels needed for calculating the envelope. One could precalculate various "sprays"[17] or one could use the same spray for all pixel positions. In agreement with the discussion of Ref. 17, we use individual sprays.

The position of the pixel is chosen in polar coordinates as follows: First the distance from the center pixel to the sample pixel is chosen as a random number $d \in [0, R]$, R being the radius of the spray, using a uniform probability distribution. Then, the polar angle of the sample pixel is chosen from a uniform distribution, $\theta \in [-\pi, \pi]$. This results in a probability density that is inversely proportional to the distance to the center pixel of the spray.[17] In order to speed up the calculation of the spray, we use precomputed look-up tables for the conversion between polar and Cartesian coordinates.

The computational complexity of the algorithm is $O(NMn)$, where n is the number of pixels in the image. In other words, the algorithm is linear in the number of image pixels. However, for practical purposes, the radius of the spray, R, has to be increased as $O(\sqrt{n})$ when increasing the image size. Doing this, the density of sampling points gets lower, so N or M, or both, have to be increased somewhat in order to obtain results of similar perceptual quality, in practice, making the algorithm heavier than a simple linear one. As a reference, calculating the image of Figure 2 (512×779 pixels) took approximately 8 s using a C implementation running on a T7700 2.4 GHz Intel Core2 Duo CPU, under Linux.

We have implemented STRESS in CUDA 1.1 on a Quadro FX3700 graphic card, without any optimization (CUDA and Quadro are provided by NVIDIA, see http://www.nvidia.com/). For a 512×1024 pixel image, with 10 sampling points and 100 iterations, the computation takes about 2 s.

The complexity of the proposed method is comparable to other SCAs.[18] Like many of them, the efficiency can be greatly improved by techniques such as using subsampling and subsequent upscaling with a local linear lut.[22]

Figure 2. The STRESS algorithm applied to the grayscale image on the left. The image size is 512×779 pixels, and the parameters used were $R = 300$, $M = 3$, $N = 100$.

APPLICATIONS

In this section, we present STRESS details and properties, through the description of a set of possible applications.

Local Contrast Enhancement of Grayscale Images

A straightforward application of STRESS is local contrast enhancement of grayscale images. Since the envelopes can be interpreted as local reference maximum and minimum points, to obtain a local effect, the pixel value should be compared to these quantities. In this way, a bright pixel should have a low value if it is close to a local reference minimum, and a dark pixel should have a high intensity value if it is close to a local reference maximum.

This is implemented assigning the values 0 to the local reference minimum and 1 to the local reference maximum and performing a linear scaling between these extrema. Again, it is important to remember that all computations are performed in a gamma corrected or perceptually uniform space. This corresponds exactly to calculating the relative position within the envelope (see Fig. 1 for an illustration):

$$p_{\text{stress}} = \frac{p_0 - E_{\min}}{E_{\max} - E_{\min}} \tag{9}$$

An example contrast stretched grayscale image is shown in Fig. 2.

Local Color Correction of Color Images

The same approach can be used for color images. If the calculation is performed independently for each color channel, the three maximum envelopes together will define a local reference maximum and the three minimum envelopes will define a local reference minimum in the three chromatic channels, respectively. As in other algorithms of the same family,[18] in case of a global or local color cast, it will result in an automatic color adjustment. An example image is shown in Figure 3. This kind of automatic color

Figure 3. The STRESS algorithm applied to the color image on the left. The image size is 512×779 pixels, and the parameters used were $R = 300$, $M = 3$, $N = 100$.

correction will, as for Retinex and ACE, tend to remove color casts if present, where wanted or not.

The local reference minimum and maximum together define the extremes of a local color cube in the linear RGB color space containing the pixel to compute (for the same reasons as in the 1D case). One way to describe locality of the envelope in color, is considering it as a local color space. This means that for each point in the image, the RGB color space changes locally according to the spatial neighborhood. With local color spaces, we can deal with a classic set of problem in a local framework.

The algorithm will perform an automatic color adjustment along the lines of ACE[21] or RSR.[17] Consequently, if the change of global white point is not wanted, methods used in ACE, such as "keep original gray," to preserve the original mean values, or "keep original color cast," to preserve (if present) a color dominant, can be easily incorporated.[23]

HDR Image Rendering

Traditionally, high dynamic range (HDR) image rendering or tone mapping has been considered a specific field of research on its own. One particularly interesting property of STRESS is that it can be applied directly, without any modification, as a local tone-rendering operator for high dynamic range images. That is, HDR images are mapped according to Eq. (9).

Also this feature is in line with SCA algorithm family.[18] It rearranges spatially and locally the relative values according to the scene content, preserving edges and compressing gradients.

Two examples of rendered HDR images are shown in Figures 4 and 5. It should be noted that not only tone-rendering but also local color adjustment, contrast stretching, and luminosity normalization are applied. If the overall color adjustment is not wanted, techniques such as keep original color cast[24] can easily be added. In the renderings, we can see details in both very light and very dark regions

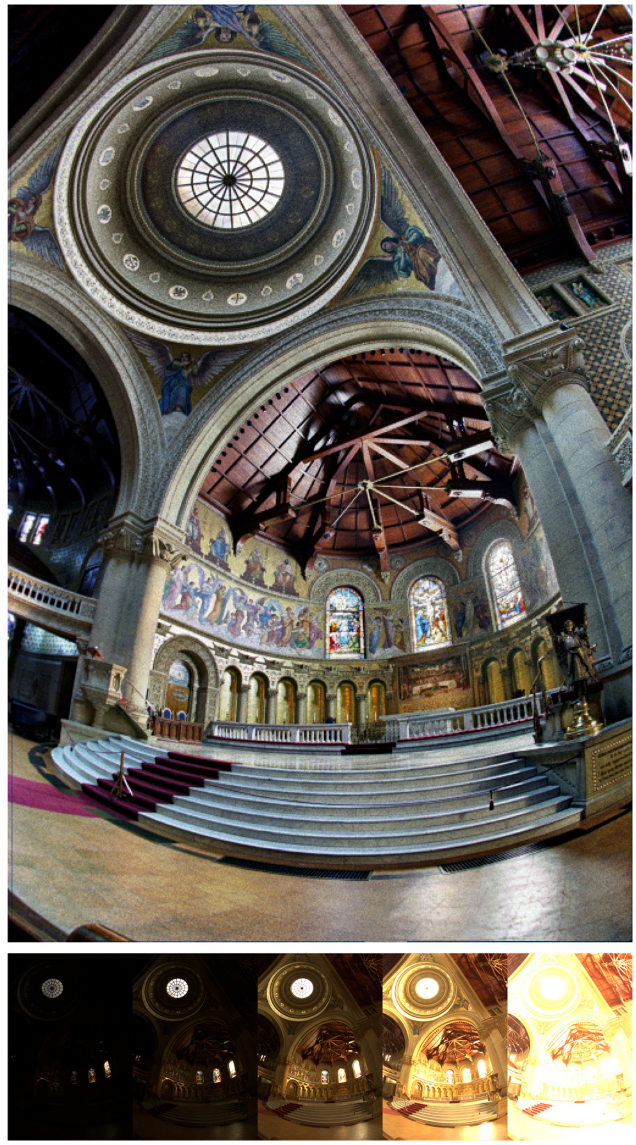

Figure 4. Rendering of the HDR image memorial. The image size is 512×768 pixels, and the parameters used were $R = 300$, $M = 10$, $N = 100$. The small frames show the HDR image at different levels of exposure.

without the generation of artifacts such as halos. However, the resulting images can appear a bit like high-pass filtered images if compressed too much. This is controlled by the parameters R, M, and N (see discussion below). STRESS has been tested on a larger set of HDR images providing stable and satisfactory tone rendering.

Spatial Color Gamut Mapping

In Ref. 25, Kolås and Farup presented a spatial color gamut mapping. First, from the original image with pixel values p_0, a gamut clipped image with pixel values p_c was constructed. The clipping was performed along straight lines toward a neutral point g on the gray axis such that $p_c = (1 - m)p_0 + mg$, m being determined for each pixel independently. Then, the map m was filtered using an edge-preserving blurring increasing filter (a filter that never

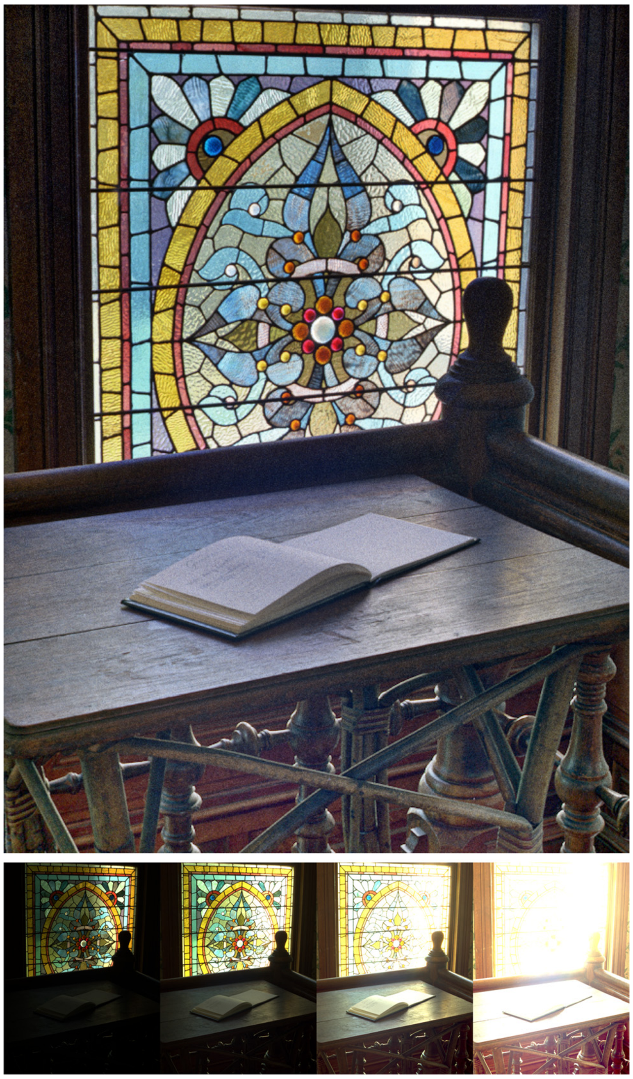

Figure 5. Rendering of the HDR image desk. The image size is 512×768 pixels, and the parameters used were $R=600$, $M=10$, $N=100$. The small frames show the HDR image at different levels of exposure.

reduces the original pixel value). The symmetric nearest neighbor (SNN) filter[26] was used for the purpose. The algorithm provided quite nice results, but the filter was found to be operating in a too local manner when compared to other spatial color gamut mapping algorithms.[27]

The procedure to compute the stress maximum envelope $E_{\max}$ works as an edge-preserving increasing filter and can easily be exchanged with the SNN filter. The resulting spatial gamut mapping procedure is thus to compute $E_{\max}(m)$ and finally to compute the final gamut mapped image as a convex linear combination of the original image and the neutral image,

$$p_{\text{sgma}} = (1 - E_{\max}(m))p_0 + E_{\max}(m)g. \tag{10}$$

By this mapping, the colors are changed only along straight lines in the color space from the original pixel color toward a point on the neutral axis. An example of an image gamut

Figure 6. Gamut clipped image and spatial gamut mapped image using the STRESS algorithm.

mapped to the ISO uncoated gamut is shown in Figure 6. The resulting images are much more natural looking than the ones produced using the SNN filter[25] in that there are no visible artifacts close to sharp edges and no visible over-enhancing of details.

Temporal Color Correction of Movies

For moving pictures, the concept of spatial envelopes can be generalized to the temporal domain. Since the envelopes are computed using an iterative approach, a better and even faster solution for moving pictures will be to perform the iterations over the frame sequence, using a running average. In this way, the local reference maximum and minimum will not only depend on the current frame itself but also on the previous frames.

This can be achieved by exchanging the Eqs. (5) and (6) with

$$\bar{r} = \alpha r + (1-\alpha)\bar{r}_p, \tag{11}$$

$$\bar{v} = \alpha v + (1-\alpha)\bar{v}_p, \tag{12}$$

where $\bar{r}_p$ and $\bar{v}_p$ being the values of $\bar{r}$ and $\bar{v}$ at the previous iteration, respectively. How quickly the local reference minimum and maximum will change in the image will depend upon the choice of the α parameter and the number of iterations on each frame.

With this temporal extension, the algorithm realizes two interesting behaviors: lightness adaptation and after images. If the video stream ranges from very bright to very dark, almost nothing will be seen in the dark to begin with, but after a while, the algorithm will adapt to the darkness and render the details well. The opposite effect will be seen when moving from dark to bright. If the scene is changed from a setting with strong colors or edges to a flat or homogeneous one, an after-image of the first scene will be seen as a negative. An example of such an after-image is shown in Figure 7.

Figure 7. After-image produced by the temporal extension of STRESS.

Color to Grayscale Conversion

A problem that has challenged many researchers is converting a color image to a grayscale image without the loss of chrominance edges[28] and without the introduction of artifacts such as halos.

Converting color to grayscale using only the lightness channel values is a good example of how perceiving a color in context can differ from the color in void condition. A classic phenomenon is the following. Take two gray patches, similar but not identical, and put them on a common background without contact, at a certain distance. They will be perceived as identical or much more similar than if posed on the same background, but in contact on a side, forming an edge. These differences in appearance are detectable when adjacent but not detectable when separated. The same phenomenon takes place also with colored patches.

Thus, the presence of edges can change the perceived lightness. Applying STRESS will result in a more clear differentiation of the edges, even if isoluminant, according to their mutual position. Having access to the local reference minimum and maximum of the image, we can easily define a local gray axis between these two points, taking into account that we are using a linear color space. Then, the pixel color can simply be projected to this local gray axis, and we have a grayscale image in which also the chrominance edges are kept.

Denote the local white point as $\mathbf{w} = [E_R^{\max} E_G^{\max} E_B^{\max}]$ and the local black point as $\mathbf{b} = [E_R^{\min} E_G^{\min} E_B^{\min}]$, respectively, and let p be the vector of the pixel values of the three color channels. Then, the gray value of that pixel is computed as

$$g = \frac{(\mathbf{p} - \mathbf{b}) \cdot (\mathbf{w} - \mathbf{b})}{|\mathbf{w} - \mathbf{b}|^2}. \tag{13}$$

A common example image for testing color to grayscale algorithms is shown in Figure 8.

Figure 8. Example of color to grayscale conversion. The image in the middle has been realized with a simple averaging of the RGB color channels. Below STRESS output, the image size is 745×498 pixels, and the parameters used were $R = 600$, $M = 10$, $N = 100$.

DISCUSSION

Like all the algorithms of the SCA family,[18] STRESS can have a varying behavior according to its parameters. Here, we want to highlight the parameters, together with some comments about the results and visual configuration on which the human visual system exhibits interesting behaviors.

Sampling

Stochastic sampling is a quick and simple way to explore the image context around the pixel in search of the local reference for the pixel adjustment. Changing the sampling method changes the way the algorithm explores the image

Figure 9. Details of STRESS output varying sampling (horizontal) and iterations (vertical). Values are 1, 5, 10, 50, and 100 for both axes.

and consequently its local behavior. The more the samples are collected, the more the higher and lower values will be representative of the neighborhood. This will result in changes in the locality and also in a decrease of noise. However, to suppress noise is more important to increase the number of iteration as will be shortly presented (see Figure 9).

A statistical characterization of the sampling techniques will be the subject of future research. However, an interesting point is that just one sampled pixel (together with the pixel itself) is enough to give to the output a rough noisy version of the image output appearance. This is the lowest sampling possible and moves the issues about noise and image quality from the computation on a single image to the effectiveness of the repetition of the computation across a series of images from the same temporal sequence. In other words, it is possible to reach a satisfactory steady result with the desired quality, both increasing the computation on a single image or alternatively keep computing with a limited number of sampling points and iterations (computationally non expensive) on temporal series of images of the same scene.

This point suggests an interesting direction on further investigation about the model and its possible analogies with the human visual system.

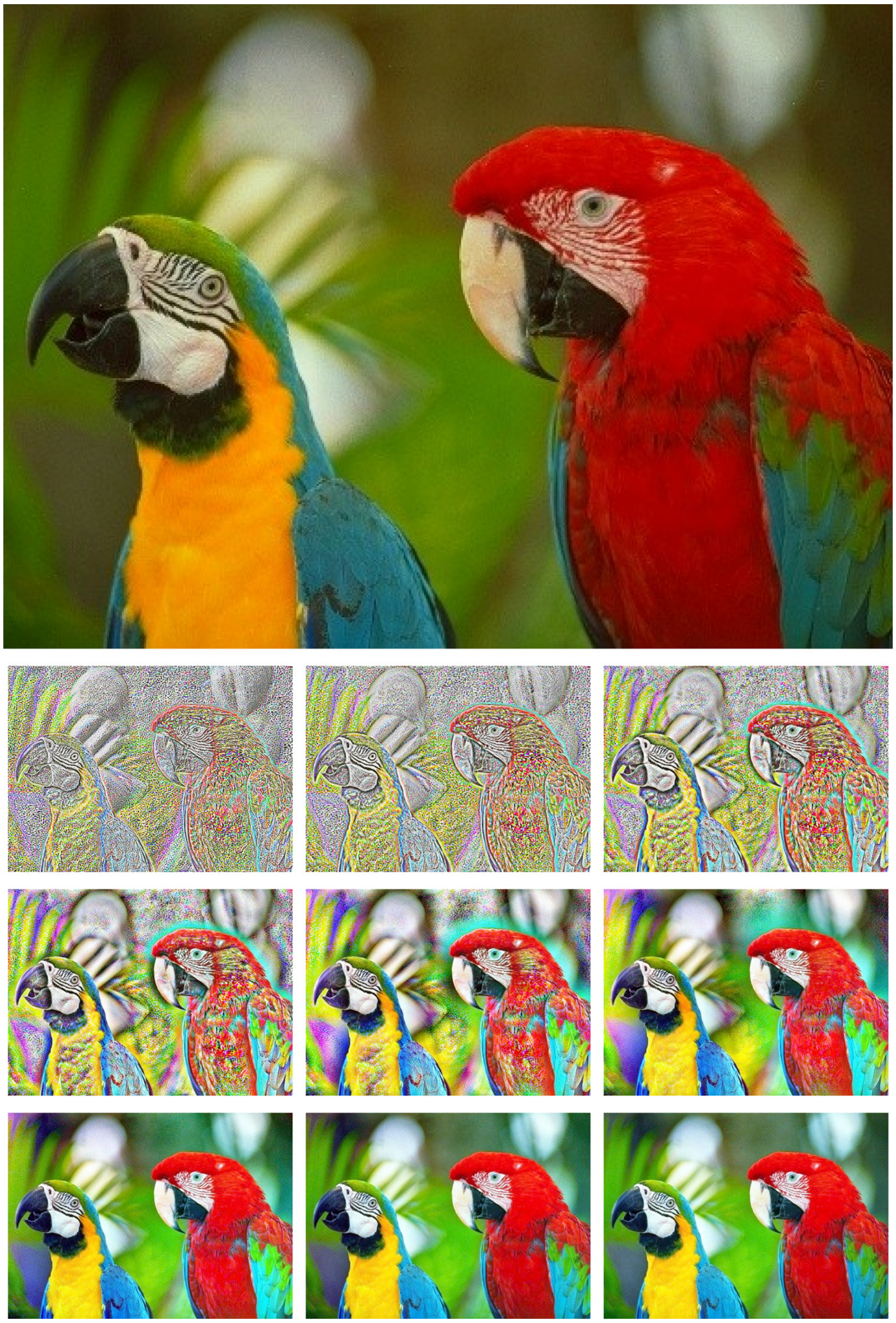

Figure 10. Examples of STRESS filtering changing radius starting from $R = 2$ upper left, doubling the radius for every image, ending at $R = 512$ lower right.

In Fig. 9, the effect of increasing the number of the samples and iterations is presented. A set of combinations of STRESS results varying the number of the sampling point and the number of iteration is presented. First column has one sample point for each pixel, the second five, the third 10, the fourth 50, and the fifth 100 sampling points. The first line shows results from one iteration, the second line five, the third 10, the fourth 50, and the fifth line 100 iterations for each pixel. The original image is visible on the top. A low number of sampling points gives the highest local contrast in the mid tones at the cost of over exposing some bright details. This effect is also visible in Figs. 2 and 3.

When the number of sampling points increases, the behavior of the algorithm gets more and more global. In the limit $M \to \infty$, the envelopes will be constant and equal the global maximum and minimum of the color channels.

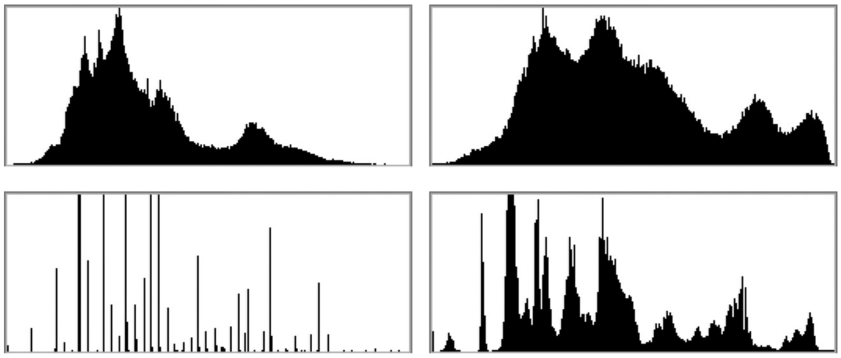

Figure 11. The effect of STRESS on the lightness histogram.

The STRESS algorithm then reduces to global linear contrast stretching. This can also be observed in Fig. 9.

Iterations

To reduce the sampling chromatic noise, the sampling process is iterated several times and averaged. This strongly decreases the noise level at the cost of increased time of computation.

Fig. 9 shows the effect of increasing the number of the samples and iterations. Different from the number of sampling points that is related to the spatial distribution of the local minima and maxima, the number of iterations affects more the variance of the computed pixel and is thus merely a way to reduce noise.

As an initial blind tuning, the number of samples can be set around ten. Lower values will produce salt and pepper noise. Increasing the number of iterations is more important. A number of iterations close to the radius give results with extremely limited noise. This criterion can be used as maximization. Lowering this number can be very useful if saving the computational power is required and down to 10–20 iterations noise presence is not annoying. Parameter tuning, as visual perception itself, depends also on the image content.

Radius

The radius parameter R is the maximum distance from the pixel where the stochastic sampling can be done. It controls the locality of the spatial maxima and minima for the adjustment. It is not a critical parameter as long as it is large enough to sample reasonably across the entire image. For all the example rendered images presented in this article, the radius is chosen to be large enough to avoid the artifacts typically resulting from a too small value of the parameter.

If the radius value decreases significantly, the sampling is localized to a very close and narrow neighborhood around the center pixel. It is interesting to note from Figure 10 how the color information derives from spatial comparisons. For very small radii, only the colors near the edges in the original image are present. Increasing the radius has the effect of spreading color. Fig. 10 shows the results with radii of 2, 4, 8, 16, 32, 64, 128, and 256 pixels (in order from left to right and from top to bottom). The original image is placed on top, and its dimensions are 480×348 pixels.

For practical purposes with real images, R should be chosen large enough to cover the entire image, e.g., equal to the diagonal of the image.

Overall Behavior

The STRESS algorithm shares some properties with both gray world algorithms and white patch algorithms for color correction. The average color of the image is mapped toward gray, whereas, at the same time, the brightest color in the image is mapped to white. This is performed locally and in a way that is edge-preserving.

Like other spatial color algorithms such as Retinex and ACE, STRESS performs a content driven histogram flattening. Figure 11 shows the lightness channel histograms of the original *parrot* image of Fig. 10 (Fig. 11 top left) and the same histogram of the STRESS filtered version (Fig. 11 top right). If the starting image has a reduced number of colors, as visible in the histogram of Fig. 11 bottom left, which refers to the original *parrot* image converted to 256 colors, the effect of STRESS is to produce colors in the larger color range spatially dequantizing the image. This is an interesting property of the spatial color algorithms.[18]

Only one of the three parameters of the algorithms can be chosen freely. R should be set large enough to cover the entire image, e.g., by setting it equal to the diagonal of the image. N should be large enough to avoid visible noise. The parameter M decides how local the behavior of the algorithm is. For extremely large values, STRESS will reduce to global contrast stretching. For extremely low values, STRESS will act somewhat similar to a high-pass filter.

The STRESS algorithm also exhibits a simultaneous contrast type of behavior caused by the spatial comparisons. Figure 12 shows the result of running STRESS with different parameters on a classic simultaneous contrast configuration. As it is visible from the figure, contrast is enhanced qualitatively in the way our visual system does as can be seen from the indicated pixel values.

CONCLUSIONS

In this article, we have presented a new framework for spatially recomputing the color of a digital image. The color of each pixel is recomputed by scaling its channel lightness

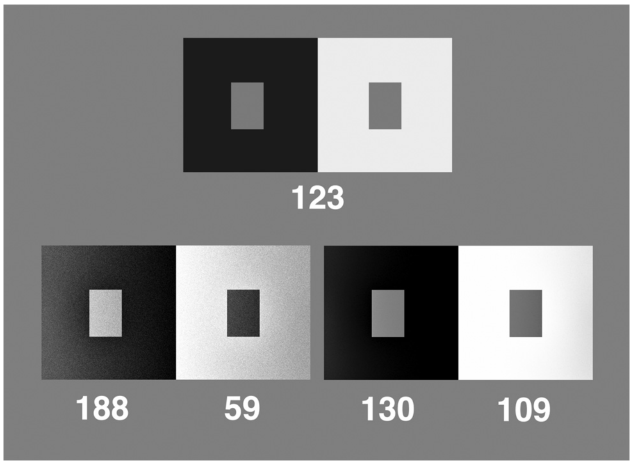

Figure 12. Example of simultaneous contrast filtering.

value according to two upper and lower envelope functions. These envelope functions are obtained sampling a limited number of pixels in the neighbor. The algorithm performs local color and lightness adjustments in an edge-preserving manner by means of spatial comparisons.

The framework can be successfully applied to image processing tasks such as color image equalization and contrast stretching, rendering of high dynamic range images, spatial color gamut mapping, color to grayscale conversion, and temporal color adjustment of movies. An implementation for moving images can be particularly efficient, due to the use of historical data. STRESS enhances the image with minimal user supervision and without any *a priori* information of the input image.

The underlying idea of the framework is simple and easy to implement, and the algorithm is efficient (linear in number of pixels) compared to other relevant spatial color algorithms.

ACKNOWLEDGMENTS

This work has been partially supported by the PRIN-COFIN 2007E7PHM3-003 project by Ministero dell'Università e della Ricerca, Italy, and by the Norwegian Research Council over the SHP program. The HDR image in Fig. 4 was provided by Paul Debevec, and the image in Fig. 5 was provided by Industrial Light and Magic.

REFERENCES

[1] T. Cornsweet, *Visual Perception* (Academic, New York, 1970).
[2] J. J. McCann and K. L. Houston, "Calculating colour sensation from arrays of physical stimuli", IEEE Trans. Syst. Man Cybern., **13**, 1000–1007 (1983).
[3] E. Land and J. McCann, "Lightness and retinex theory", J. Opt. Soc. Am., **61**, 1–11 (1971).
[4] J. J. McCann Ed, "Special session on retinex at 40", J. Electron. Imaging, **13**, 6–145 (2004).
[5] J. J. McCann, S. McKee, and T. Taylor, "Quantitative studies in retinex theory: a comparison between theoretical predictions and observer responses to color mondrian experiments", Vis. Res., **16**, 445–458 (1976).
[6] E. Land, "The retinex theory of color vision", Sci. Am., **237**, 108–128 (1977).
[7] J. Frankle and J. J. McCann, "Method and apparatus of lightness imaging", US Patent 4,384,336. (17 May 1983).
[8] D. Marini and A. Rizzi, "A computational approach to color adaptation effects", Image Vis. Comput., **18**, 1005–1014 (2000).
[9] T. J. Cooper and F. A. Baqai, "Analysis and extensions of the Frankle-McCann retinex algorithm", J. Electron. Imaging, **13**, 85–92 (2004).
[10] B. Funt, F. Ciurea, and J. J. McCann, "Retinex in Matlab", J. Electron. Imaging, **13**, 48–57 (2004).
[11] E. Land, "Recent advances in retinex theory and some implications for cortical computations: color vision and the natural image", Proc. Natl. Acad. Sci. U.S.A., **80**, 5163–5169 (1983).
[12] D. J. Jobson, Z. Rahman, and G. A. Woodel, "Properties and performance of a center/surround retinex", IEEE Trans. Image Process., **6**, 451–462 (1997).
[13] K. Barnard and B. Funt, "Investigations into multi-scale retinex", *Proc. Colour Imaging in Multimedia '98* (Colour & Imaging Institute, University of Derby, Derby, United Kingdom, 1998).
[14] M. Bertalmio and J. Cowan, "Implementing the Retinex algorithm with Wilson–Cowan equations", J. Physiol., Paris, **103**, 69–72 (2009).
[15] R. Kimmel, M. Elad, D. Shaked, R. Keshet, and I. Sobel, "A variational framework for retinex", Int. J. Comput. Vis., **52**, 7–23 (2003).
[16] J. Morel, A. Petro, and C. Sbert, "A PDE formalization of retinex theory", IEEE Trans. Image Process., **19**, 2825–2837 (2010).
[17] E. Provenzi, M. Fierro, A. Rizzi, L. De Carli, D. Gadia, and D. Marini, "Random spray retinex: a new retinex implementation to investigate the local properties of the model", IEEE Trans. Image Process., **16**, 162–171 (2007).
[18] A. Rizzi and J. J. McCann, "On the behavior of spatial models of color", *Proc. SPIE*, **6493**, 649303, (2007).
[19] R. Sobol, "Improving the Retinex algorithm for rendering wide dynamic range photographs", J. Electron. Imaging, **13**, 65–74 (2008).
[20] D. Shaked and R. Keshet, "Robust recursive envelope operators for fast Retinex", Technical Report HPL-2002-74(R.1), (HP Laboratories Israel, Technion City, Haifa, Israel, March 2004).
[21] A. Rizzi, C. Gatta, and D. Marini, "A new algorithm for unsupervised global and local color correction", Pattern Recogn. Lett., **24**, 1663–1677 (2003).
[22] C. Gatta, A. Rizzi, and D Marini, "Local linear lut method for spatial color correction algorithm speed-up", IEE Proc. Vision Image Signal Process., **153**, 357–363 (2006).
[23] C. Gatta, "Human visual system color perception models and applications to computer graphics", PhD thesis, Università degli Studi di Milano, Italia, November 2005.
[24] A. Rizzi and M. Chambah, "Perceptual color film restoration", SMPTE J., **19**, 33–41 (2010).
[25] Ø. Kolås and I. Farup, "Efficient hue-preserving and edge-preserving spatial color gamut mapping", *Proc. IS&T/SID 15th Color Imaging Conference* (IS&T, Springfield, VA, 2007), pp. 207–212.
[26] D. Harwood, M. Subbarao, H. Hakalathi, and L. S. Davis, "A new class of edge-preserving smoothing filters", Pattern Recogn. Lett., **6**, 155–162 (1987).
[27] F. Dugay, I. Farup, and J. Y. Hardeberg, "Perceptual evaluation of color gamut mapping algorithms", Color Res. Appl., **33**, 470–476 (2008).
[28] R. Bala and R. Eschbach, "Spatial color-to-grayscale transform preserving chrominance edge information", *Proc. IS&T/SID 12th Color Imaging Conference* (IS&T, Springfield, VA, 2005), pp. 82–86.