Visual Comput (2008) 24: 505–514
DOI 10.1007/s00371-008-0231-2

ORIGINAL ARTICLE

Giovane R. Kuhn
Manuel M. Oliveira
Leandro A. F. Fernandes

An improved contrast enhancing approach for color-to-grayscale mappings

Published online: 29 May 2008

Electronic supplementary material
The online version of this article (doi:10.1007/s00371-008-0231-2) contains supplementary material, which is available to authorized users.

G.R. Kuhn · M.M. Oliveira · L.A.F. Fernandes (✉)
Instituto de Informática, UFRGS,
Av. Bento Gonçalves, 9500, Caixa Postal 15.064, Porto Alegre – RS, Brazil
{grkuhn, oliveira, laffernandes}@inf.ufrgs.br

Abstract Despite the widespread availability of color sensors for image capture, the printing of documents and books is still primarily done in black-and-white for economic reasons. In this case, the included illustrations and photographs are printed in grayscale, with the potential loss of important information encoded in the chrominance channels of these images. We present an efficient contrast enhancement algorithm for color-to-grayscale image conversion that uses both luminance and chrominance information. Our algorithm is about three orders of magnitude faster than previous optimization-based methods, while providing some guarantees on important image properties. More specifically, our approach preserves gray values present in the color image, ensures global consistency, and locally enforces luminance consistency. Our algorithm is completely automatic, scales well with the number of pixels in the image, and can be efficiently implemented on modern GPUs. We also introduce an error metric for evaluating the quality of color-to-grayscale transformations.

Keywords Color reduction · Color-to-grayscale · Image processing · Error metric

1 Introduction

Multimegapixel digital cameras are commonplace and essentially all pictures taken nowadays are in color, with a few exceptions mostly for artistic purposes. On the other hand, due to economic reasons, the printing of documents and books is still primarily done in "black-and-white", causing the included photographs and illustrations to be printed in shades of gray. Since the standard color-to-grayscale conversion algorithm consists of computing the luminance of the original image, all chrominance information is lost in the process. As a result, clearly distinguishable regions containing isoluminant colors will be mapped to a single gray shade (Fig. 1). As pointed out by Grundland and Dodgson [7], a similar situation happens with some legacy pattern recognition algorithms and systems that have been designed to operate on luminance information only. By completely ignoring chrominance, such methods cannot take advantage of a rich source of information.

In order to address these limitations, a few techniques have been recently proposed to convert color images into grayscale ones with enhanced contrast by taking both luminance and chrominance into account [5, 7, 10, 12]. The most popular of these techniques [5, 12] are based on the optimization of objective functions. While these two methods produce good results in general, they present high computational costs, not scaling well with the number of pixels in the image. Moreover, they do not preserve the gray values present in the original image. Grayscale preservation is a very desirable feature and is satisfied by the traditional techniques that perform color-to-grayscale conversion using luminance only. We present an efficient approach for contrast enhancement during color-to-grayscale conversion that addresses these limitations.

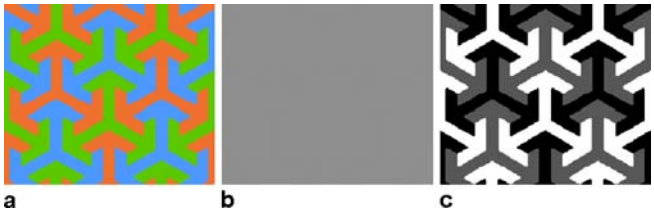

Fig. 1a–c. Color-to-grayscale mapping. **a** Isoluminant color image. **b** Image obtained using the standard color-to-grayscale conversion algorithm. **c** Grayscale image obtained from **a** using our algorithm

The contributions of this paper include:

- A new contrast enhancement algorithm for color-to-grayscale image conversion that uses both luminance and chrominance information (Sect. 3), and presents several desirable features:
  1. Preserves the gray values found in the color image
  2. Ensures that any pixels with the same color in the original image will be mapped to the same shade of gray (*global consistency*)
  3. Maintains local luminance consistency
  4. Is completely automatic
  5. Can be efficiently implemented on modern GPUs
- A new contrast error metric for evaluating the quality of color-to-gray transformations (Sect. 4)

Figure 2d illustrates the result of our technique applied to the color image shown in Fig. 3 and compares it to a luminance image (Fig. 2a), and to the results produced by the Color2Gray algorithm of Gooch et al. [5] (Fig. 2b) and by the technique of Rasche et al. [12] (Fig. 2c). Figure 2e–h show the per-pixel contrast errors associated with the grayscale images on the top row. These errors were computed comparing the inter-pixel contrasts in the color and grayscale images (Sect. 4). The result produced by our technique on this (839×602)-pixel image not only has the smallest contrast error, but it is also the fastest. Our GPU implementation performs the decoloring in 0.435 s. This is three orders of magnitude faster than the Rasche et al. approach and five orders of magnitude faster than the Gooch et al. approach. Our CPU implementation is still $247\times$ faster than Rasche et al. and $25\,279\times$ faster than Gooch et al.

2 Related work

Mapping a color image to grayscale is a dimensionality reduction problem. Traditional techniques use projections or weighted sums to map a three-dimensional color space to a single dimension (e.g., the luminance value of *XYZ*, *YCbCr*, $L^*a^*b^*$, or *HSL* color spaces). They are the common methods implemented in commercial applications [1, 8]. These approaches, however, do not take into account any chrominance information, mapping isoluminant pixels to the same gray value, as shown in Fig. 1b.

A popular dimensionality reduction technique is principal component analysis (PCA) [3]. However, as pointed out by [5, 12] PCA fails to convert color images with variations along many directions. Besides, PCA-based approaches would require an optimization technique to mix the principal components. Grundland and Dogdson [7] convert the original RGB colors to their *YPQ* color

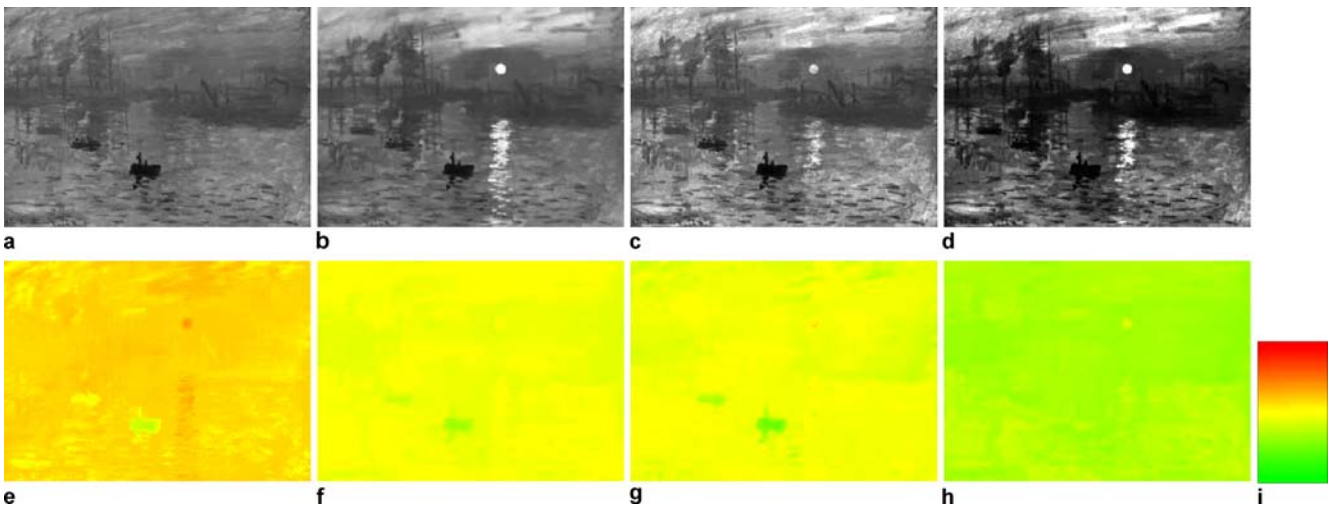

Fig. 2a–i. Four grayscale renditions of Claude Monet's *Impressionist Sunrise* (Fig. 3), with their respective error images obtained using our metric. Images **a**–**d** are grayscale images with their corresponding per-pixel contrast error images **e**–**h**, respectively. **a** Luminance image. **b** Grayscale image produced by the Gooch et al. method using its default parameters. **c** A grayscale image produced by the Rasche et al. approach. **d** Grayscale image produced by our approach. RWMS error images: rwms = 0.582 (**e**), 0.535 (**f**), 0.443 (**g**), 0.365 (**h**). Error scale is shown in **i**, where *red* indicates larger error

Fig. 3. Color image (*Impressionist Sunrise* by Claude Monet, courtesy of Artcyclopedia.com)

space and perform dimensionality reduction using a technique they called *predominant component analysis*, which is similar to PCA. In order to decrease the computational cost of this analysis, they use a local sampling by a Gaussian pairing of pixels that limits the amount of color differences processed and brings the total cost to convert an $N \times N$ image to $O(N^2 \log(N^2))$. This technique is very fast, but its local analysis may not capture the differences between spatially distant colors and, as a result, it may map clearly distinct colors to the same shade of gray. This situation is illustrated in Fig. 4.

Neumann et al. [10] presented an empirical color-to-grayscale transformation algorithm based on the Coloroid system [9]. Based on the results of a user-study, they sorted the relative luminance differences between pairs of seven hues, and interpolated between them to obtain the relative luminance differences among all colors. Their algorithm requires the specification of two parameters, and the reported running times are on the order of five to ten seconds per megapixel (hardware specs unknown).

Gooch et al. [5] find gray levels that best represent the color difference between all pairs of colors by optimizing an objective function. The ordering of the gray levels arising from the original colors with different hues is resolved with a user-provided parameter. The cost to optimize an $N \times N$ image is $O(N^4)$, causing the algorithm to scale poorly with image resolution.

Rasche et al. [12] formulated the color-to-grayscale transformation as an optimization problem in which the perceptual color difference between any pair of colors should be proportional to the perceived difference in their corresponding shades of gray. In order to reduce its computation cost, the authors perform the optimization on a reduced set Q of quantized colors, and this result is then used to optimize the gray levels of all pixels in

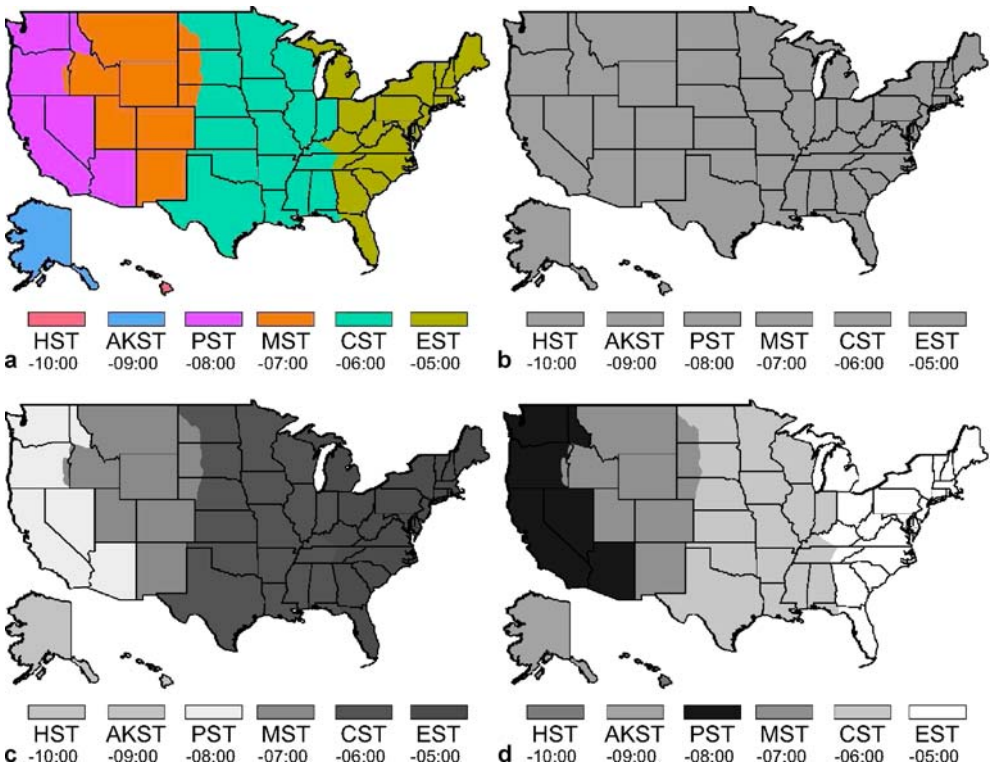

Fig. 4a–d. USA time zone map. **a** Color image. **b** Luminance image. **c** Grayscale image produced by Grundland and Dogdson's [7] method. **d** Grayscale image obtained using our approach. Note in **c** how the color contrast between some spatially distant regions were not preserved by Grundland and Dogdson's approach (e.g., HST and AKST time zones, CST and EST time zones)

the resulting image. The total cost of the algorithm is $O(\|Q\|^2 + \|Q\|N^2)$.

3 The contrast enhancing algorithm

Our contrast enhancing algorithm has three main steps. The first step consists of obtaining a set Q of quantized colors from the set of all colors C in the input image I. This can be performed using any color quantization technique. The second step performs a constrained optimization on the values of the luminance channel of the quantized colors using a mass-spring system. At this stage, the chrominance information is taken into account in the form of constraints that specify how much each particle can move (Sect. 3.1). The final gray values are reconstructed from the set of gray shades produced by the mass-spring optimization (Sect. 3.2). This final step guarantees local luminance consistency preservation.

3.1 Modeling and optimizing the mass-spring system

Our approach for color-to-grayscale mapping is modeled as a mass-spring system whose topology is a complete graph, i.e., each particle P_i is connected to each other particle P_j by a spring S_{ij}. Here, a particle P_i is associated with a quantized color $\boldsymbol{q}_i \in Q$ (represented in the almost perceptually uniform CIE $L^*a^*b^*$ color space) containing some mass m_i. Such particles are only allowed to move along the L^*-axis of the color space and have a gray level initialized with the value of the luminance channel of $\boldsymbol{q}_i$. Between each pair of particles (P_i, P_j), we create a spring with rest length given by

$$l_{ij} = \frac{G_{\text{range}}}{Q_{\text{range}}} \|\boldsymbol{q}_i - \boldsymbol{q}_j\|, \tag{1}$$

where Q_{range} is the maximum difference between any pair of quantized colors in Q, G_{range} is the maximum possible difference between any pair of luminance values, and $\|\boldsymbol{q}_i - \boldsymbol{q}_j\|$ approximates the perceptual difference between colors $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$. Note that since the luminance values are constrained to the L^*-axis, $G_{\text{range}} = 100$.

The instantaneous force applied to a particle P_i is obtained by summing the tensions of all springs connecting P_i to its neighbors P_j, according to Hooke's law:

$$F_i = \sum_{j \in N} k_{ij} \left(1 - \frac{l_{ij}}{l'_{ij}}\right)\left(L_j^* - L_i^*\right), \tag{2}$$

where N is the set of neighbors linked to P_i, and L_i^* and L_j^* are the current luminance values associated to particles P_i and P_j, respectively. The terms l_{ij} and l'_{ij} are, respectively, the rest length (Eq. 1) and current length of the spring linking P_i and P_j, and $k_{ij} = 1$ is the fixed stiffness of that spring.

Given a time step Δt, the new luminance value of particle P_i is computed using Verlet's integration [15] as:

$$L_i^*(t + \Delta t) = \frac{F_i(t)}{m_i} \Delta t^2 + 2L_i^*(t) - L_i^*(t - \Delta t), \tag{3}$$

where $L_i^*(t)$ is the luminance of particle P_i at time t, and m_i is the mass of P_i. At each step of the optimization, we update l'_{ij} as $|L_i^* - L_j^*|$, and the new luminance value L^* according to Eq. 3. The resulting system tends to reach an equilibrium when the perceptual differences between the optimized gray levels are proportional to the perceptual differences among the quantized colors in Q.

In order to enforce grayscale preservation, we set the mass m_i of particle P_i as the reciprocal of the magnitude of $\boldsymbol{q}_i$'s chrominance vector (Fig. 5):

$$m_i = \frac{1}{\left\|\left(a_i^*, b_i^*\right)\right\|}. \tag{4}$$

Note that $d = \|(a_i^*, b_i^*)\|$ is the distance from color $\boldsymbol{q}_i$ to the luminance axis L^*. Thus, less saturated colors present bigger masses and tend to move less. For achromatic colors, whose mass should be infinity, we avoid the division by zero simply by setting $F_i = 0$ (Eq. 2). This keeps achromatic colors stationary.

3.2 Interpolating the final gray image

The last step of the algorithm consists of obtaining the gray values for all pixels of the resulting image. For this task, we have developed two distinct approaches: *per-cluster interpolation* and *per-pixel interpolation*. The choice for one interpolation method depends on the application requirements.

Per-cluster interpolation. Consider the set $\boldsymbol{q}_k \in Q$ of quantized colors and the respective associated set $g_k \in G$ of optimized gray levels. Let $C_k \subset C$ be a cluster composed of all colors in C that in the optimization are represented by the quantized color $\boldsymbol{q}_k$. The final gray level

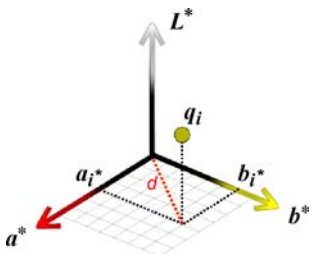

Fig. 5. The mass of a particle associated with a quantized color $\boldsymbol{q}_i$ is computed as the reciprocal of its distance d to the luminance axis L^*: $m_i = 1/(\|(a_i^*, b_i^*)\|)$. This enforces grayscale preservation, as achromatic colors will remain stationary

associated to the mth color $\boldsymbol{c}_m^k \in C_k$ is then obtained as

$$\text{gray}(\boldsymbol{c}_m^k) = \begin{cases} g_k + s_k \|\boldsymbol{q}_k - \boldsymbol{c}_m^k\| & \text{lum}(\boldsymbol{c}_m^k) \geq \text{lum}(\boldsymbol{q}_k) \\ g_k - s_k \|\boldsymbol{q}_k - \boldsymbol{c}_m^k\| & \text{otherwise,} \end{cases} \quad (5)$$

where lum is the function that returns the coordinate L^* of a color in the $L^*a^*b^*$ color space, and s_k is Shepard's [13] interpolation of ratios computed as

$$s_k = \frac{\sum_{i=1}^{\|Q\|} w_{ki} \frac{|g_k - g_i|}{\|\boldsymbol{q}_k - \boldsymbol{q}_i\|}}{\sum_{i=1}^{\|Q\|} w_{ki}}, \quad \text{for } i \neq k. \quad (6)$$

The term s_k indicates how close the gray value g_k is to the optimal solution, and $w_{ki} = 1/\|\boldsymbol{q}_k - \boldsymbol{q}_i\|^2$ is the distance-weighted term. For the quantized color $\boldsymbol{q}_k$ that represents the cluster, all gray values inside the kth cluster are computed with respect to the optimized gray level g_k. Therefore, this transformation ensures local luminance consistency.

Given the set Q of quantized colors, the cost of computing all cluster ratios using Eq. 6 on the CPU is $O(\|Q\|^2)$, while the cost of interpolating each pixel of an image with $N \times N$ pixels is $O(N^2)$.

Per-pixel interpolation. In this approach, each pixel's final shading is computed by optimizing it against the set $g_k \in G$ of previously optimized gray levels. This is achieved by using a mass-spring system, with springs connecting the current pixel (which is treated as a particle initialized with the pixel's luminance value) and all optimized gray levels g_k. In this refined optimization stage, the particles associated with the optimized gray levels are kept stationary by setting the forces that act on them to zero (F_i in Eq. 2). Equation 4 is then used to obtain the mass of the pixel being optimized. In this stage, all pixels with achromatic colors end up having infinite masses, remaining stationary. This ensures that all gray shades in the original color image will be preserved in the resulting grayscale image.

For an $N \times N$ image, the cost of this optimization procedure ($O(\|Q\|N^2)$) is higher than the mapping defined by Eq. 5. However, as the final gray level of each pixel can be obtained independently from all other pixels, the computation can be efficiently implemented in a fragment program.

4 Error metric for color-to-grayscale mappings

We introduce an error metric to evaluate the quality of color-to-grayscale transformations. Note that the proposed error metric is not a perceptual one. It measures whether the difference between any pairs of colors $(\boldsymbol{c}_i, \boldsymbol{c}_j)$ in the original color image has been mapped to the corresponding proper target difference in the grayscale image. For this purpose, we defined an error function using the root weighted mean square (RWMS):

$$\text{rwms}(i) = \sqrt{\frac{1}{\|K\|} \sum_{j \in K} \frac{1}{\delta_{ij}^2} (\delta_{ij} - |\text{lum}(\boldsymbol{c}_i) - \text{lum}(\boldsymbol{c}_j)|)^2}, \quad (7)$$

where, $\text{rwms}(i)$ is the error computed for the ith pixel of the input color image I, K is the set of all pixels in I, $\|K\|$ is the number of pixels in I, $\delta_{ij} = (G_{\text{range}}/C_{\text{range}})\|\boldsymbol{c}_i - \boldsymbol{c}_j\|$ is the target difference in gray levels for a pair of colors $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$, and lum is the function that returns the component L^* of a color. Since the differences are computed in the CIE $L^*a^*b^*$ color space, $G_{\text{range}} = 100$ and C_{range} is the maximum distance between any two colors in the color image I. The weight $(1/\delta_{ij}^2)$ is used to suppress the bias toward large values of δ_{ij}. For an $N \times N$ image, evaluating Eq. 7 for every pixel of I would take $O(N^4)$, which becomes impractical for large values of N. We can obtain a very good approximation to this error function by restricting the computation to the set $q_j \in Q$ of quantized colors, as shown in Eq. 8:

$$\text{rwms}_q(i) = \sqrt{\frac{1}{\|K\|} \sum_{j \in Q} \frac{\|K_j\|}{\delta_{ij}^{q\,2}} \left(\delta_{ij}^q - |\text{lum}(\boldsymbol{c}_i) - \text{lum}(\boldsymbol{q}_j)|\right)^2}. \quad (8)$$

$K_j \subset K$ is the cluster of pixels represented by the quantized color $\boldsymbol{q}_j$, $\delta_{ij}^q = (G_{\text{range}}/Q_{\text{range}})\|\boldsymbol{c}_i - \boldsymbol{q}_j\|$, $\boldsymbol{c}_i$ is the color of the ith pixel, and Q_{range} is the maximum distance between any two quantized colors in Q. We have compared the RWMS values produced by Eqs. 7 and 8 for a set of 50 images. From this study, we found that the average relative difference between the two results was only 1.47%. Given its significantly smaller cost $O(\|Q\|N^2)$, all contrast errors shown in the paper were computed using the metric represented by Eq. 8. Also, in all error images shown in this paper, the green shade shown at the bottom of the error color ramp indicates $\text{rwms} = 0.0$, while the red shade at the top represents $\text{rwms} = 1.2$.

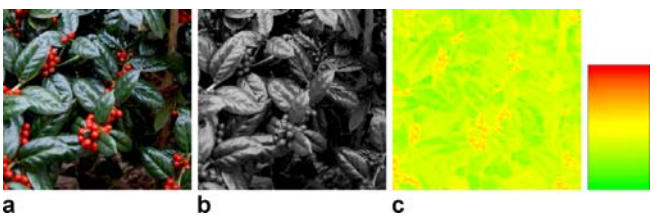

Fig. 6a–c. Example of our contrast error metric. **a** Color image. **b** Luminance image of **a**. **c** Error image computed using Eq. 8. The largest contrast errors concentrate on the berry pixels

Figure 6 illustrates the use of our contrast error metric. Figure 6c is the error image for the pair of color and grayscale images shown on Fig. 6a and Fig. 6b, respectively, using a set of 64 quantized colors. The grayscale image was obtained as the luminance of Fig. 6(a). As expected, the largest errors concentrate on the berry pixels, since these present the biggest contrast lost. Smaller errors are spread over the several green shades of the leaves.

5 Results

Per-cluster interpolation can be implemented quite efficiently on a CPU due to its lower computational cost. The quality of the results depends directly on the quality of the quantization algorithm used. According to our experience, per-cluster interpolation produces excellent results in combination with k-means. On the other hand, per-pixel interpolation has a higher computation cost, since it optimizes all pixels with respect to the set Q of quantized colors. Fortunately, its computation can be efficiently implemented on a GPU. Due to its refined optimization procedure, per-pixel interpolation can be used in combination with a less expensive and faster quantization algorithm, like uniform quantization.

We implemented the described algorithms in C++ and GLSL, and used them to decolorize a very large number of images. The reported times were measured using a 2.2 GHz PC with 2 GB of memory and on a GeForce 8800 GTX with 768 MB of memory. One should note that ours is not the first mass-spring implementation on a GPU. This has been done before by other researchers [2, 4, 14], but the specific needs and topology of our application lends itself to a more efficient GPU implementation.

Figure 7 compares the times for quantization and decolorizing images with various resolutions using different algorithms. *MS-PC CPU* is our mass-spring algorithm using per-cluster interpolation in combination with k-means, and *MS-PP GPU* is our mass-spring algorithm using per-pixel interpolation with uniform quantization. In the case of k-means, we used a set of 128 colors. In the case of uniform quantization, we discretized the RGB space using a uniform $10 \times 10 \times 10$ grid. Figure 7 shows

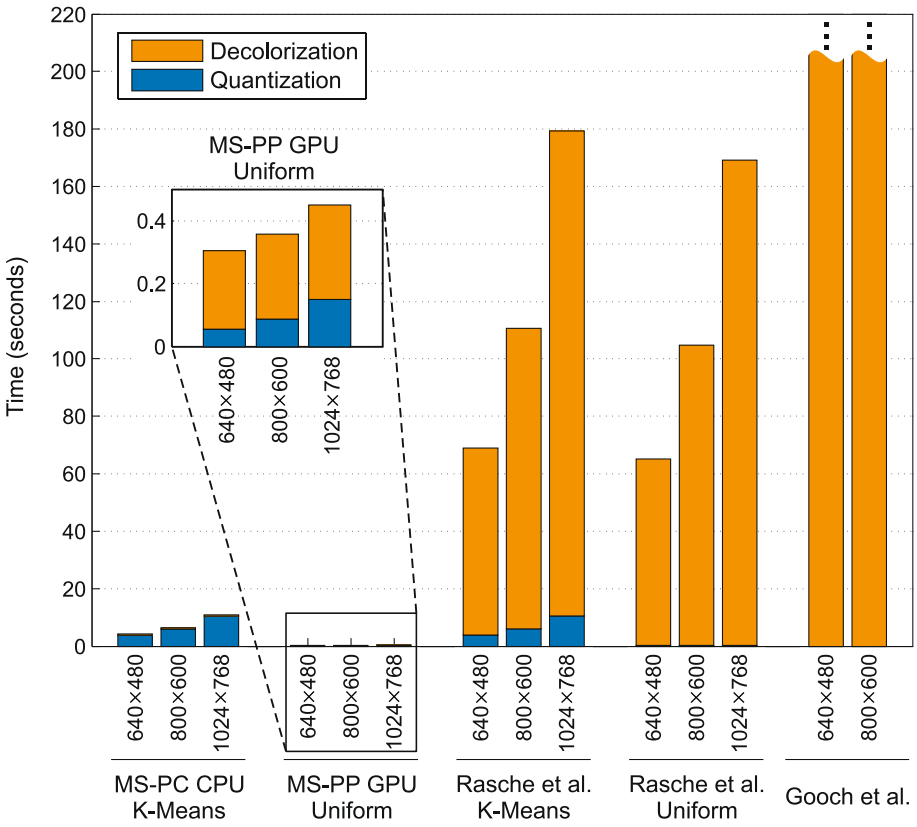

Fig. 7. Performance comparison of various algorithms on a 2.2 GHz PC with 2 GB of memory and on a GeForce 8800 GTX GPU using images of different resolutions. Gooch et al. performed in 12 276 and 30 372 s for (640×480) and (800×600)-pixel images, respectively. Except for Gooch et al., all other techniques used a set of 128 quantized colors. Our mass-spring (MS) approaches optimized the set of quantized colors using 1000 iterations. The *GPU* version obtained the final gray levels by optimizing each pixel with 100 iterations. Its results are detailed for better visualization. Note how the proposed approach scales well with the size of the input images

that in all of its variations, our approach is a few orders of magnitude faster than both Gooch et al. and Rasche et al. approaches. All images and execution times shown in this paper regarding the techniques of Gooch et al. [5] and Rasche et al. [12] were obtained using software provided by these authors at [6] and [11], respectively.

Figure 8 compares the results produced by various techniques with respect to grayscale preservation. One should note that only the luminance image (Fig. 8b) and the result produced by our method (Fig. 8f) are capable of preserving the original shades of gray. The luminance image, however, failed to distinguish the shades of the various isoluminant circles. Gooch et al. (Fig. 8c) and Rasche et al. (Fig. 8d, e) techniques changed the original gray shades in the resulting images.

Figures 2, 9–11 compare the results, performance, and the overall contrast errors produced by the various algorithms. Table 1 summarizes these data. Following the authors comments on image quality, we did not use any quantization with the approach from Gooch and coworkers. For the Rasche et al. and our approach, the input images were quantized as shown in the second column of Table 1.

Table 1 also shows that our approach simultaneously presents the smallest RWMS error and is by far faster than the Gooch and Rasche techniques. The luminance image, on the other hand, presents the biggest overall contrast errors, which is something that was already expected, since the color-to-luminance mapping completely ignores the chrominance information of the original image.

Figure 2 shows four grayscale renditions of Claude Monet's *Impressionist sunrise* (Fig. 3), with their respective contrast error images obtained using our metric. This example illustrates the robustness of our technique to han-

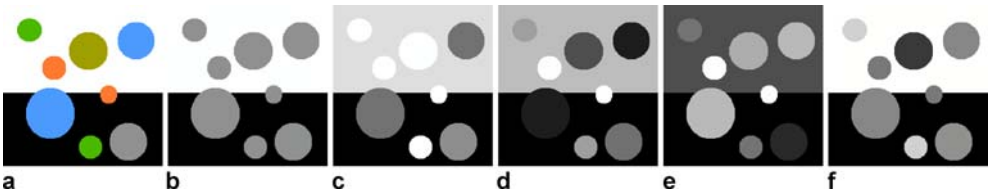

Fig. 8a–f. Example of grayscale preservation. **a** Original color image with isoluminant circles. **b** Luminance image obtained from **a**. Note it maps all isoluminant circles to the same shade of *gray*. **c** Result produced by the Gooch et al. technique. Note that the two shades of *green* and the shade of *orange* turned into *white* in the resulting image; **d** and **e** are two results produced by the Rasche et al. approach. **f** Grayscale image produced by our approach. Note that only the luminance image (**b**) and the result produced by our approach (**f**) preserved the original *gray shades*. The results shown in **d–f** took a set of seven uniformly quantized colors as input

Fig. 9a–j. Pablo Picasso's *Lovers*. **a** Color image (courtesy of Artcyclopedia.com). **b–e** Grayscale images with their per-pixel contrast RWMS error images **g–j**, respectively. **b** Luminance image. **c** Grayscale image produced by the Gooch et al. method using its default parameters. **d** A grayscale image produced by the Rasche et al. approach. Note that it is hard to distinguish between the lady's yellow skirt and the man's red clothes. **e** Grayscale image produced by our approach. **f** Error scale, where *red* indicates a larger error

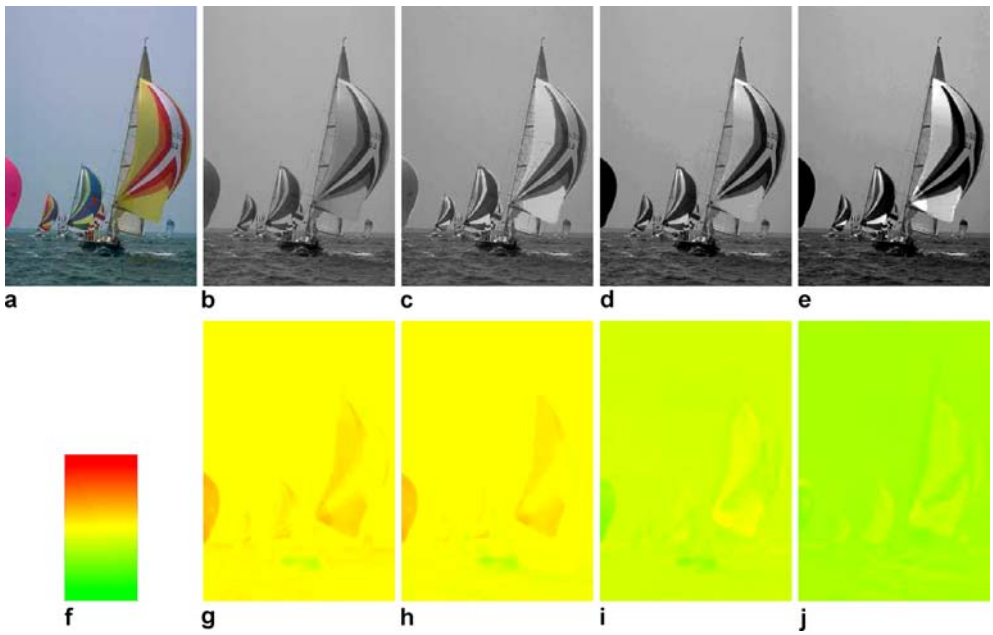

Fig. 10a–j. Photograph of a natural scene. **a** Color image. **b–e** Grayscale images with their per-pixel contrast RWMS error images **g–j**, respectively. **b** Luminance image. **c** Grayscale image produced by the Gooch method using its default parameters. **d** A grayscale image produced by the Rasche approach. **e** Grayscale image produced by our approach. **f** Error scale, where *red* indicates a larger error

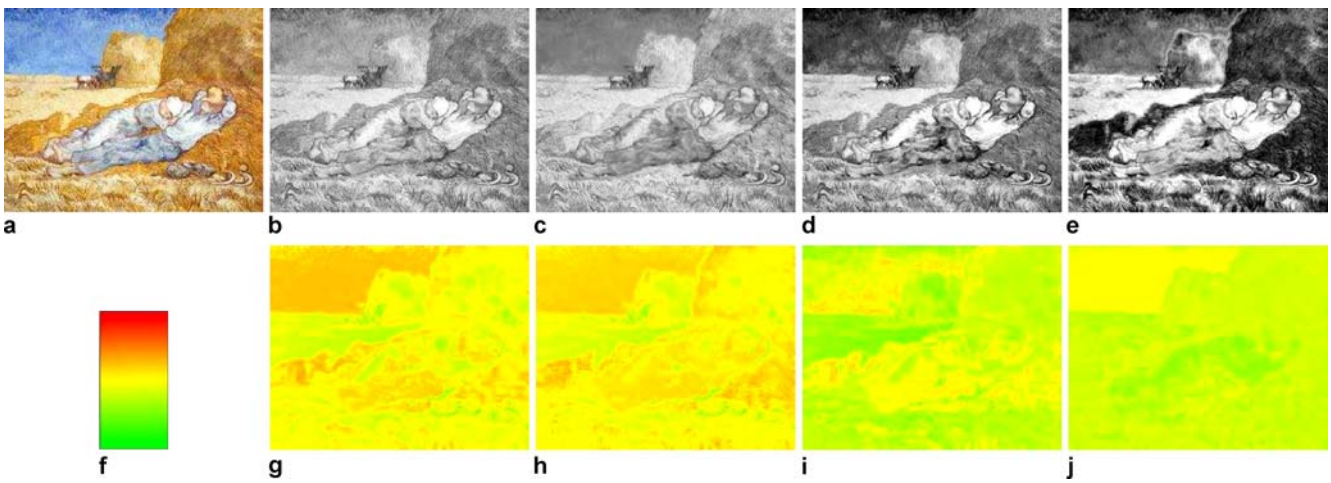

Fig. 11a–j. Claude Monet's *Midday Rest*. **a** Color image (courtesy of Artcyclopedia.com). **b–e** Grayscale images with their per-pixel contrast RWMS error images **g–j**, respectively. **b** Luminance image. **c** Grayscale image produced by the Gooch method using its default parameters. **d** A grayscale image produced by the Rasche approach. **e** Grayscale image produced by our approach. **f** Error scale, where *red* indicates a larger error

dle large images. The *Sunrise* has (839×602) pixels and our GPU implementation performs the decolorization in 0.435 s. This is 751× faster than the Rasche approach and 77 910× faster than the Gooch approach. Our CPU implementation is still 247× faster than the Rasche approach and 25 379× faster than the one by Gooch et al.

Picasso's *Lovers* provides an example for which the result produced by the Gooch technique presents a large contrast error (Fig. 9h). For this same image, the Rasche approach produced a relatively small contrast error, but in the resulting grayscale image it is hard to distinguish between the lady's yellow skirt and the man's red clothes. For the photograph shown in Fig. 10, the overall contrast error produced by Gooch (Fig. 10h) is about the same as the one found in the luminance image (Fig. 10g).

Table 1. Summary of the performance and overall contrast error produced by the various techniques when applied to the various images. Time measured in seconds. Our approach presents the smallest RWMS error for all examples and is significantly faster than the other techniques. The speedups increase with the image sizes. For the *Sunrise* image, with (839 × 602) pixels, our GPU implementation is 751× faster than the Rasche (CPU) approach and 77 910× faster than the Gooch approach (CPU)

Image (size)	Quant. (No. of colors)	Lum. RWMS	Gooch et al. Time	Gooch et al. RWMS	Rasche et al. Time	Rasche et al. RWMS	MS-PC CPU Time	MS-PC CPU RWMS	MS-PP GPU Time	MS-PP GPU RWMS
Sunrise (839 × 602)	Uniform (264)	0.707	33 501.4	0.557	326.78	0.564	1.32	0.429	0.43	0.425
Lovers (301 × 407)	K-means (255)	0.690	1882.5	0.699	87.36	0.498	0.96	0.486	0.36	0.477
Boats (193 × 282)	Uniform (141)	0.634	328.3	0.624	20.10	0.513	0.35	0.432	0.17	0.428
Midday (275 × 216)	K-means (128)	0.641	1383.0	0.649	41.71	0.521	0.38	0.520	0.18	0.517

Figure 11 shows an example for which, according to the proposed metric, the result produced by the Gooch technique (Fig. 11h) is similar to the luminance image (Fig. 11g). For this example, the overall errors produced by Rasche et al. (Fig. 11i) and by our technique (Fig. 11j) are the smallest ones. However, our GPU approach produced its result 232× faster than Rasche et al. and our CPU implementation was still 110× faster than Rasche et al.

6 Conclusions

We presented an efficient mass-spring-based approach for contrast enhancement during color-to-grayscale image conversion. Our method is more than three orders of magnitude faster than previous optimization-based techniques [5, 12], while producing superior results both in terms of contrast preservation and image guarantees. Our algorithm satisfies a global consistency property, preserves grayscale values present in the color image, maintains local luminance consistency, is completely automatic, and can be efficiently implemented on modern GPUs.

We have also introduced an error metric for evaluating the quality of color-to-grayscale transformations. Our metric is based on a RWMS error that measures whether the difference between any pairs of colors in the original image have been mapped to the corresponding target difference in the grayscale image.

Although our algorithms guarantee a continuous mapping among gray shades in any cluster, it does not deal with discontinuity across different clusters. However, after extensive tests on a great variety of images, we were unable to notice any visual artifacts caused by this limitation.

The proposed approach was designed to deal with static images. We are exploring ways to extend our technique to perform video sequences decolorization. Preliminary results show that we can enforce temporal coherence by initializing the mass-spring optimization with particles computed for previous frames, and by keeping those particles stationary. Temporal coherence is not preserved by related techniques [5, 7, 10, 12].

The unique combination of high-fidelity capture of color differences, grayscale preservation, global consistency, local luminance consistency, and speed makes our technique a good candidate for replacing standard luminance-based color-to-grayscale algorithms in printing and pattern recognition applications.

Acknowledgement We would like to thank Karl Rasche, Amy Gooch, and their collaborators for making their code available, and Nvidia for generously donating the GeForce 8800 GTX used in this research. Figure 10 is a courtesy of the Berkeley Segmentation Dataset. We are grateful to Carlos Dietrich for the discussions about mass-spring systems and for the base code to solvers on the CPU and GPU. Bárbara Bellaver and Eduardo Pons produced the supplemental video. This work was partially sponsored by CAPES (Brazil). We acknowledge the following CNPq-Brazil fellowships: 305613/2007-3 (Manuel M. Oliveira) and 142627/2007-0 (Leandro A.F. Fernandes). Microsoft Brazil provided additional support.

References

1. Brown, R.: Photoshop: Converting color to black-and-white. http://www.russellbrown.com/tips_tech.html (2006). Accessed 2007
2. Dietrich, C.A., Comba, J.L.D., Nedel, L.P.: ShaderX 5. In: Storing and Accessing Topology on the GPU: A Case Study on Mass-Spring Systems, pp. 565–578. Charles River Media, Boston (2006)
3. Dunteman, G.: Principal Components Analysis. Sage, Thousand Oaks, CA (1989)
4. Georgii, J., Westermann, R.: Mass-spring systems on the GPU. Simul. Modell. Pract. Theory **13**, 693–702 (2005)
5. Gooch, A.A., Olsen, S.C., Tumblin, J., Gooch, B.: Color2gray: salience-preserving color removal. ACM Trans. Graph. **24**(3), 634–639 (2005)
6. Gooch, A.A., Olsen, S.C., Tumblin, J., Gooch, B.: Color2gray: Salience-preserving color removal. http://www.color2gray.info (2005). Accessed 2007
7. Grundland, M., Dodgson, N.A.: Decolorize: Fast, contrast enhancing, color to grayscale conversion. Pattern Recognit. **40**(11), 2891–2896 (2007)
8. Jeschke, E.R.: Gimp: Converting color images to b&w. http://www.gimp.org/tutorials/Color2BW/ (2002). Accessed 2007
9. Nemcsics, A.: Coloroid color system. Color Res. Appl. **5**, 113–120 (1980)
10. Neumann, L., Cadik, M., Nemcsics, A.: An efficient perception-based adaptive color to gray transformation. In: Proceedings of Computational Aesthetics 2007, pp. 73–80. Eurographics

Association, Banff, Canada (2007)
11. Rasche, K.: Detail preserving color transformation. http://www.fx.clemson.edu/rkarl/c2g.html (2005). Accessed 2007
12. Rasche, K., Geist, R., Westall, J.: Re-coloring images for gamuts of lower dimension. Comput. Graph. Forum **24**(3), 423–432 (2005)
13. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 1968 23rd ACM National Conference, pp. 517–524. ACM, New York, NY (1968)
14. Tejada, E., Ertl, T.: Large steps in GPU-based deformable bodies simulation. Simulation Practice and Theory, Special Issue on Programmable Graphics Hardware **13**(8), 703–715 (2005)
15. Verlet, L.: Computer "experiments" on classical fluids. I. thermodynamical properties of Lennard–Jones molecules. Phys. Rev. **159**(1), 98 (1967)

G.R. KUHN is a M.Sc. student at the Federal University of the Rio Grande do Sul (UFRGS) under the supervision of Dr. Manuel M. Oliveira. He received his B.S. degree from the Regional University of Blumenau (FURB) in 2005. His research interests lie in all aspects of image processing, especially on perceptually based algorithms and perceptual metrics.

M.M. OLIVEIRA is a faculty member at the Federal University of Rio Grande do Sul (UFRGS), in Brazil. He received his Ph.D. in computer science from the University of North Carolina at Chapel Hill, in 2000. Before joining UFRGS, he was an Assistant Professor at SUNY Stony Brook from 2000 to 2002. His research interests cover most aspects of computer graphics, but especially in the frontiers among graphics, vision, and image processing. This includes image-based and real-time rendering, 3D photography, 3D scanning, representation and rendering of surface details, and surface reconstruction from point clouds.

L.A.F. FERNANDES is a Ph.D. student at UFRGS, under the supervision of Dr. Manuel M. Oliveira. He received his M.Sc. degree in computer science from the Federal University of Rio Grande do Sul (UFRGS) in 2006 and B.S. degree in computer science from the Regional University of Blumenau (FURB) in 2002. His research interests include image processing, image metrology, image-based modeling, image-based tracking, and real-time rendering.