Fast Image and Video Colorization Using Chrominance Blending

Liron Yatziv and Guillermo Sapiro

***Abstract*—Colorization, the task of coloring a grayscale image or video, involves assigning from the single dimension of intensity or luminance a quantity that varies in three dimensions, such as red, green, and blue channels. Mapping between intensity and color is, therefore, not unique, and colorization is ambiguous in nature and requires some amount of human interaction or external information. A computationally simple, yet effective, approach of colorization is presented in this paper. The method is fast and it can be conveniently used "on the fly," permitting the user to interactively get the desired results promptly after providing a reduced set of chrominance scribbles. Based on the concepts of luminance-weighted chrominance blending and fast intrinsic distance computations, high-quality colorization results for still images and video are obtained at a fraction of the complexity and computational cost of previously reported techniques. Possible extensions of the algorithm introduced here included the capability of changing the colors of an existing color image or video, as well as changing the underlying luminance, and many other special effects demonstrated here.**

***Index Terms*—Chrominance blending, colorization, gradient, interpolation, intrinsic distance, recolorization, special effects.**

I. Introduction

COLORIZATION is the the art of adding color to a monochrome image or movie. The idea of "coloring" photos and films is not new. Ironically, hand coloring of photographs is as old as photography itself. There exists such examples from 1842 and possibly earlier [19]. It was practiced in motion pictures in the early 1900s by the French Company Pathe, where many films were colored by hand. It was also widely practiced for filmstrips into the 1930s. The computer-assisted process was first introduced by Markle in 1970 for adding colors to black and white movies [3].

As neatly presented by Sykora *et al.* [26] (their work also includes an outstanding overview of the literature on the subject), various early computer-based colorization techniques include straightforward approaches such as *luminance keying* [9]. This method is based on a user-defined lookup table which transforms grayscale into color. Welsh *et al.* [28], inspired by work of Reinhard *et al.* [21] and Hertzmann *et al.* [11], extended this idea by matching luminance and texture rather than just the grayscale values.

Chen *et al.* [5] used manual segmentation to divide the grayscale image into a set of layers. Then an alpha channel was estimated using Bayesian image matting. This decomposition allows to apply colorization using Welsh's approach. The final image is constructed using alpha-blending. Recently, Sykora *et al.* [26] have similarly used a segmentation method optimized for the colorization of black and white cartoons.

Other approaches, including our own, assume that homogeneity of the grayscale image indicates homogeneity in the color. In other words, as detailed in [23], the geometry of the image is provided by the geometry of the grayscale information (see also [4], [6], and [15]). Often, in these methods, in addition to the grayscale data, color hints are provided by the user via scribbles. Horiuchi [12] used a probabilistic relaxation method while Levin *et al.* [17] solved an optimization problem that minimizes a quadratic cost function derived from the color differences between a pixel and its weighted average neighborhood colors. Sapiro [23] proposed to inpaint the colors constrained by the grayscale gradients and the color scribbles that serve as boundary conditions. The method reduces to solving linear or nonlinear Poisson equations.

The main shortcoming of these previous approaches is their intensive computational cost, needed to obtain good-quality results. Horiuchi and Hirano addressed this issue in [13], where they presented a faster algorithm that propagates colored seed pixels in all directions and the coloring is done by choosing from a preselected list of color candidates. However, the method produces visible artifacts of block distortion since no color blending is performed. While Horiuchi's method colorizes a still image within a few seconds, we present in this paper a propagation method that colorizes a still image within a second or less, achieving even higher quality results. In contrast with works such as those in [17], the technique proposed here is easily extended to video without the optical flow computation, further improving in the computational cost, at no sacrifice in the image quality. Other special effects are also easily obtained using our framework, as demonstrated here and further exemplified at http://mountains.ece.umn.edu/~liron/colorization/.

The scheme proposed here is based on the concept of color blending. This blending is derived from a weighted distance function efficiently computed from the luminance channel. The underlying approach can be generalized to produce other effects such as recolorization. In the remainder of this paper, we describe the algorithm and present a number of examples.

II. Fast Colorization Framework

Similar to other colorization methods, e.g., [17] and [23], we use luminance/chrominance color systems. We present our

Manuscript received December 1, 2004; revised May 24, 2005. This work was supported in part by the Office of Naval Research, in part by the National Science Foundation, and in part by the National Geospatial-Intelligence Agency. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Stanley J. Reeves.

The authors are with the Electrical and Computer Engineering Department, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: liron@ece.umn.edu; guille@ece.umn.edu).

Digital Object Identifier 10.1109/TIP.2005.864231

method in the *YCbCr* color space, although other color spaces such as *YIQ* or *YUV* could be used as well. Moreover, work can also be done directly on the RGB space. Let $Y(x,y,\tau):\Omega\times[0,T]\rightarrow\Re^+$ be the given monochromatic image ($T=0$) or video ($T>0$) defined on a region Ω. Our goal is to complete the *Cb* and *Cr* channels $Cb(x,y,\tau):\Omega\times[0,T]\rightarrow\Re^+$ and $Cr(x,y,\tau):\Omega\times[0,T]\rightarrow\Re^+$, respectively. For clarity of the exposition, we refer to both channels as the chrominance. The proposed technique also uses as input observed values of the chrominance channels in a region $\Omega_c\in\Omega$ which is significantly smaller than Ω (see [17]). These values are often provided by the user or borrowed from other data.

Let s and t be two points in Ω and let $C(p):[0,1]\rightarrow\Omega$ be a curve in Ω. Also, let $C_{s,t}$ represent a curve connecting s and t such that $C(0)=s$ and $C(1)=t$. We define the intrinsic (geodesic) distance between s and t by

$$d(s,t):=\min_{C_{s,t}}\int_0^1|\nabla Y\cdot\dot{C}(p)|\,dp. \qquad (1)$$

This intrinsic distance gives a measurement of how "flat" is the flattest curve between any two points in the luminance channel. The integral in the equation above is basically integrating the luminance (Y) gradient in the direction of the curve $C(p)$ (as given by projecting the gradient ∇Y into the tangent $\dot{C}$ to this curve). When considering the minimum over all paths $C_{s,t}$, we then keep the one with the smallest overall gradient in this direction, thereby the flattest path connecting the two points s and t (the path that goes from s to t with minimal overall gradient). Note that the minimal path needs not to be unique, but we only care about the intrinsic length $d(s,t)$ of this path, so this does not affect the algorithm.

Geodesic distances of this type can be efficiently and accurately computed using recently developed fast numerical techniques [10], [14], [24], [25], [27], [29], [31]. We found that for the application at hand, even simpler techniques such as a *best first* one (in particular, Dijkstra [8]), integrated in the pseudocode below, are sufficient. This will be further detailed in the Section III.

Even though a mapping between luminance and chrominance is not unique, a close relationship between the basic geometry of these channels is frequently observed in natural images (see, for example, [4], [6], and [15]). Sharp luminance changes are likely to indicate an edge in the chrominance, and a gradual change in luminance often indicates that the chrominance is also not likely to have an edge but rather a moderate change. In other words, as reported in the aforementioned works, often, there is a close relationship between the geometries of the luminance and chrominance channels. Exploiting this assumption, a change in luminance causes a related change in chrominance. This has been used in different fashions in [17] and [23], as well as in [2] for super resolution. From this, for the proposed colorization approach we assume that the smaller the intrinsic distance $d(s,t)$ between two points (s,t), the more similar chrominance they would have.[1]

[1]It is important to notice that the goal of colorization is not to restore the original color of the image or scene, which is, in general, not available, but as in image inpainting, to produce visually pleasant and compelling colored images (see Section IV for more on this).

Since the chrominance data is often given in whole regions and not necessarily in single isolated points, we would like to get an idea of the distance from a certain known chrominance ("scribbles" with a given uniform color) to any point t in Ω. We then define the intrinsic distance $d_c(t)$ from a point t (to be colored) to a certain chrominance c, as the minimum distance $d(s,t)$ from t to any point s of the same chrominance c in the set of given colors Ω_c

$$d_c(t):=\min_{\forall s\in\Omega_c:\mathrm{chrominance}(s)=c} d(s,t). \qquad (2)$$

This gives the distance from a point t to be colored to scribbles (from the provided set Ω_c) with the same color c, or, in other words, the distance from a point to a provided color.

Our idea for colorization is to compute the *Cb* and *Cr* components (chrominance) of a point t in the region where they are missing ($\Omega\backslash\Omega_c$) by blending the different chrominance in Ω_c according to their intrinsic distance to t

$$\mathrm{chrominance}(t)\leftarrow\frac{\sum_{\forall c\in\mathrm{chrominances}(\Omega_c)}W(d_c(t))c}{\sum_{\forall c\in\mathrm{chrominances}(\Omega_c)}W(d_c(t))} \qquad (3)$$

where $\mathrm{chrominances}(\Omega_c)$ stands for all the different unique chrominance in the set Ω_c, and $W(\cdot)$ is a function of the intrinsic distance that translates it into a blending weight. In other words, the above blending expression assigns to any point t to be colored, a color that is a weighted average of the different colors in the provided set of scribbles Ω_c. For every distinct color c in the set Ω_c, the distance to it from t is computed following (2) [which uses (1)], and this distance is used to define the weight of the color c at the point t (the blending proportion of this color). The denominator in the above equation is just a normalization factor.

The function $W(\cdot)$ that provides the weighting factor from the distance $d_c(t)$ should hold some basic properties:

1) $\lim_{r\rightarrow 0}W(r)=\infty$;
2) $\lim_{r\rightarrow\infty}W(r)=0$;
3) $\lim_{r\rightarrow\infty}W(r+r_0)/W(r)=1, r_0$ a constant.

The first two requirements are obvious. Requirement 3) is necessary when there are two or more chrominance sources close by, but the blending is done relatively far from all sources. The desired visual result would be the even blending of all chrominance. For the experiments reported below, we used

$$W(r)=r^{-b} \qquad (4)$$

where b is the blending factor, typically $1\leq b\leq 6$. This factor defines how smooth is the chrominance transition.

Note that, in theory, following the equations above, a point t to be colored will be influenced by all distinct colors c in Ω_c, since $d_c(t)<\infty$.

A. Implementation Details

A key component of our algorithm is the computation of the intrinsic, gradient weighted, distance provided by (1). We should then first concentrate on how to compute this distance. Geodesic distances of this type have been widely studied in the scientific computing community, in general, as well as in the control and image-processing communities, in particular. For example, expressions of this type have been used for edge detection in [16]. They constitute a modification of the popular geodesic active contours for object detection (see [22] and references therein for details). The popularity of these expressions in other disciplines means that their efficient computation has been widely studied, e.g., [10], [14], [24], [25], [27], [29], and [31], where the reader is referred for more details, including exact numerical computations of the gradient in (1) using up-wind schemes. These algorithms are inspired by the classical Dijkstra graph algorithm [8],[2] fast sweeping techniques [7], and special data structures such as buckets and priority queues [29]. A straightforward and very fast and accurate implementation of our algorithm consists in computing for every point t to be colored in $\Omega \backslash \Omega_c$ the distance $d_c(t)$ to every distinct chrominance c in the given Ω_c, using these fast techniques, thereby obtaining the set of weights needed for the blending (3). We found that there is no real need to implement the computation of the weighted distance using the aforementioned accurate techniques, and simply using classical Dijkstra is sufficient. Thereby, in principle, we create a graph where each image pixel is a vertex connected to its neighbors with edges that, following (1), have weights equal to the absolute gray value derivatives between the neighbors, and work with Dijkstra's algorithm on this graph. Following the results in [29] (see also the references therein), we could use more accurate distance computations at minimal additional cost, but we found this not to be necessary to obtain the high quality results reported here.

To complete the description of the proposed colorization technique, we provide a pseudocode that exemplifies one of the possible ways to efficiently implement the proposed algorithm [(1)–(3)] and further emphasizes the simplicity of the proposed technique. This is just a standard form of implementing the classical Dijkstra-based ideas mentioned above.

```
• **Input:**
    — Grayscale video/image.
    — List of observed pixels $\Omega_c$ and their chrominance (user provided scribbles).
• **Output**: Colored video/image.
• **Definitions:**
    — For each pixel in $\Omega$ a *blend* structure is created that contains:
        1) $\chi$—list of chrominances (the chrominances that reached this blend)
        2) $\omega$—list of distances (the corresponding intrinsic distance to each chrominance in $\chi$)
    — A *link* structure contains:
        1) ptr—pointer to a pixel (the destination pixel of the link)
        2) $c$—a chrominance (the chrominance the link attempts to transfer)
        3) $d$—distance (the intrinsic distance from the chrominance origin in $\Omega_c$)
• **The basic algorithm:**
    1) create an empty *blend* structure for each pixel in $\Omega$
    2) $L \leftarrow \emptyset$
    3) for each pixel $p$ in $\Omega_c$
        a) let $c_p$ be the chrominance of the scribble in $p$
        b) create link $\lambda$
        c) $(\lambda.\mathrm{ptr}, \lambda.c, \lambda.d) \leftarrow (p, c_p, 0)$
        d) $L \leftarrow L \bigcup \lambda$
    4) while $L \neq \emptyset$
        a) $\lambda \leftarrow$ the link with smallest distance in $L$ (see Footnote (4)).
        b) $L \leftarrow L \backslash \lambda$
        c) $b \leftarrow$ the *blend* structure of $\lambda.\mathrm{ptr}$
        d) if $\lambda.c \notin b.\chi$
            i) add $\lambda.c$ to $b.\chi$
            ii) add $\lambda.d$ to $b.\omega$
            iii) for all pixels $q$ neighboring $p$
                A) create link $\lambda_q$
                B) $(\lambda_q.\mathrm{ptr}, \lambda_q.c, \lambda_q.d) \leftarrow (q, \lambda.c, \lambda.d + \mathrm{distance}(p, q))$[3]
                C) $L \leftarrow L \bigcup \lambda_q$
    5) for all pixels in $\Omega$; generate the chrominance channels by applying (3) with the pixels' *blend* structures.
```

B. Performance and Relaxation

The described colorization algorithm has average time and space complexity of $O(|\Omega| \cdot |\mathrm{chrominances}(\Omega_c)|)$.[4] The algorithm passes over the image/video for each different chrominance observed in Ω_c and needs a memory in the order of the number of different chrominances observed in Ω_c, times the input image/video size. If there are a large number scribbles of different chrominances, the algorithm could be relatively slow and pricey in memory (although still more efficient that those previously reported in the literature).

Fortunately, since human perception of color blending is limited, high blending accuracy is not fully necessary to obtain satisfactory results. Experimental results show that it is enough just to blend the most significant chrominance (the chrominance with the closest intrinsic distance to their observed source). We found that, in natural images, it is enough to blend just the two or three most significant chrominance to get satisfactory results. Such a relaxation reduces both the time and space complexity to $O(|\Omega|)$, thereby linear in the amount of data. Therefore, we do not include in the blend chrominances that their weight in the blending equation is small relatively to the total weight. Additional quality improvements could be achieved if an adaptive threshold, following results such as those from the *MacAdam* ellipses [18], is used. Any color lying just outside an ellipse is at the "just noticeable difference" (jnd) level, which is the smallest perceivable color difference under the experiment conditions. A possible use of this is to define an adaptive threshold that would filter out chrominance that if added to the blend would not cause a jnd.

This proposed algorithm relaxation of limiting the number of contributors to the blending equation gives a tight restriction on how far the chrominance will propagate to be included in the blend. The restriction can be easily implemented adding conditions to d) in 4) in the pseudocode.

[2]The Dijkstra algorithm is not consistent and has a bias when measuring distances on a grid. The aforementioned techniques are correct for this bias.

[3]The function $\mathrm{distance}(p, q)$ is the distance between p and q. Following (1) and the Dijkstra algorithm, this is obtained by a numerical approximation of the absolute value of the corresponding luminance derivative; e.g., just absolute luminance difference for a simple approximation.

[4]Using a priority queue with $O(1)$ average complexity per operation, as done in [1], [27], and [29]; a heap sort data structure as used in the original Dijkstra algorithm would slightly increase the run-time complexity.

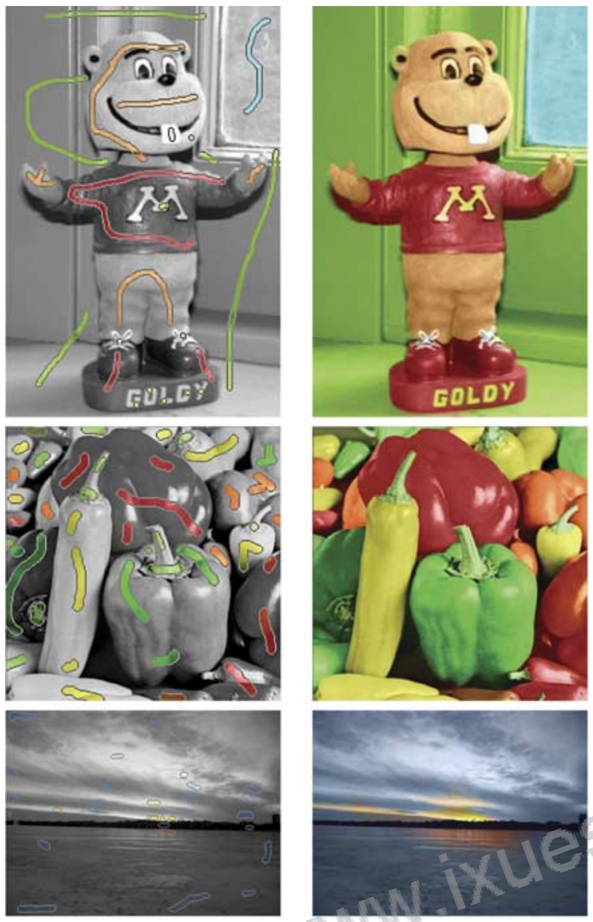


Fig. 1. Still-image colorization examples. Given a grayscale image, (left) the user marks chrominance scribbles, and (right) our algorithm provides a colorized image. The image size/run time from top to bottom are 230 × 345/less than 0.42 s, 256 × 256/less than 0.36 s, and 600 × 450/less than 1.73 s.

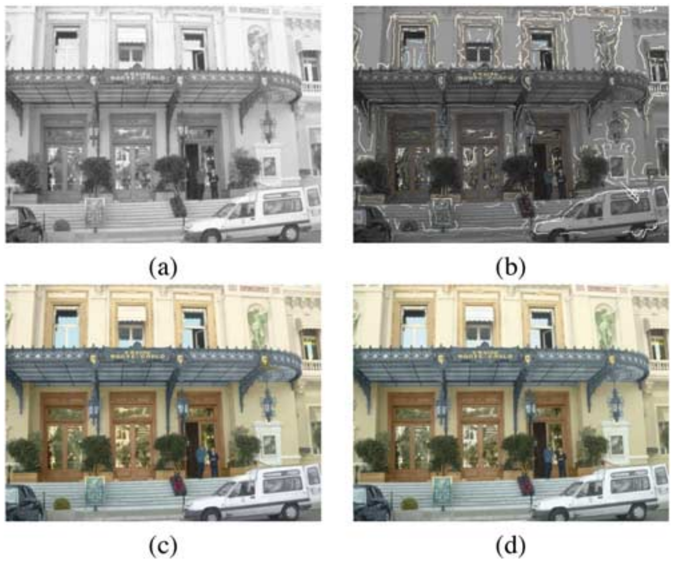


Fig. 2. Comparison of visual quality with the technique proposed in [17]. (a) Given grayscale image; (b) the user marks chrominance scribbles (the selected scribbles are obtained from the work in [17]); (c) our algorithm results with CPU run-time of 0.76 s; (d) Levin *et al.* approach with CPU run-time of 24 s using their supplied fast implementation based on multigrid solver (611 s are needed for their standard Matlab implementation). We observe the same quality at a significantly reduced computational cost.

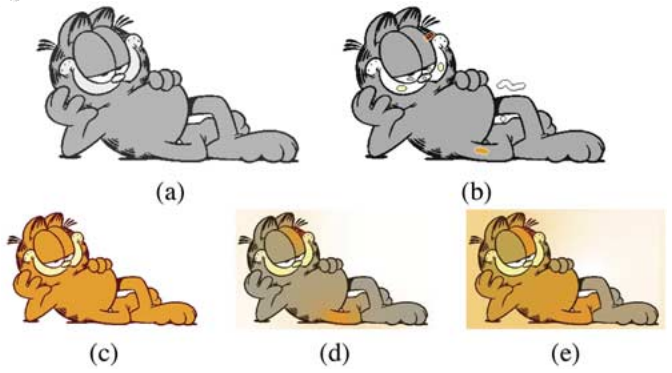


Fig. 3. Comparison of visual quality with the technique proposed in [17]. (a) Given grayscale image (400 × 240); (b) the user marks chrominance scribbles; (c) our algorithm results with CPU run time of 1.20 s; (d) Levin *et al.* approach with CPU run time of 22.8 s using their supplied fast implementation of multigrid solver; (e) Levin *et al.* approach using a slower exact Matlab least squares solver also provided by the authors.

III. Colorization Results

We now present examples of our image and video colorization technique. Additional examples, comparisons, and movies, as well as software for testing our approach, can be found at http://mountains.ece.umn.edu/~liron/colorization/. This site also contains examples beyond colorization, such as video recoloring, depth effects, decolorizing, and depth-of-field effects.

The proposed algorithm has been implemented in C++ as a stand alone win32 application so it could be tested for speed and quality. For timing we used an Intel Pentium III with 512-MB RAM running under Windows 2000. Fig. 1 shows examples of still image colorization using our proposed algorithm. The algorithm run time for all the examples in Fig. 1, measured once the images were loaded into memory, is less than 7 μs per pixel.

Figs. 2 and 3 compare our method with the one recently proposed by Levin *et al.* [17] (this work partially inspired our own). The method minimizes the difference between a pixel's color and the weighted average color of its neighboring pixels. The weights are provided by the luminance channel. The minimization is an optimization problem, subject to constraints supplied by the user as chrominance scribbles. Solving this is computationally costly and slower than our proposed technique. First, in Fig. 2, we observe that we achieve the same visual quality at a fraction of the computational cost (more comparisons are provided in the aforementioned web site, all supporting the same finding). Overall, the method proposed in [17] performs very well on many images, yet Fig. 3 demonstrates that it can perform poorly when colorizing relatively far from the provided color constrains. Following the equations in the previous section [see (3)], even far away pixels will receive color from the scribbles with our approach (in particular from the closest ones, in weighted distance, see section on relaxation above). In order to match visual quality with our technique, the method proposed in [17] needs more user input, meaning additional color scribbles. We also found that the inspiring technique developed in [17] has a sensible scale parameter and often fails at strong edges,

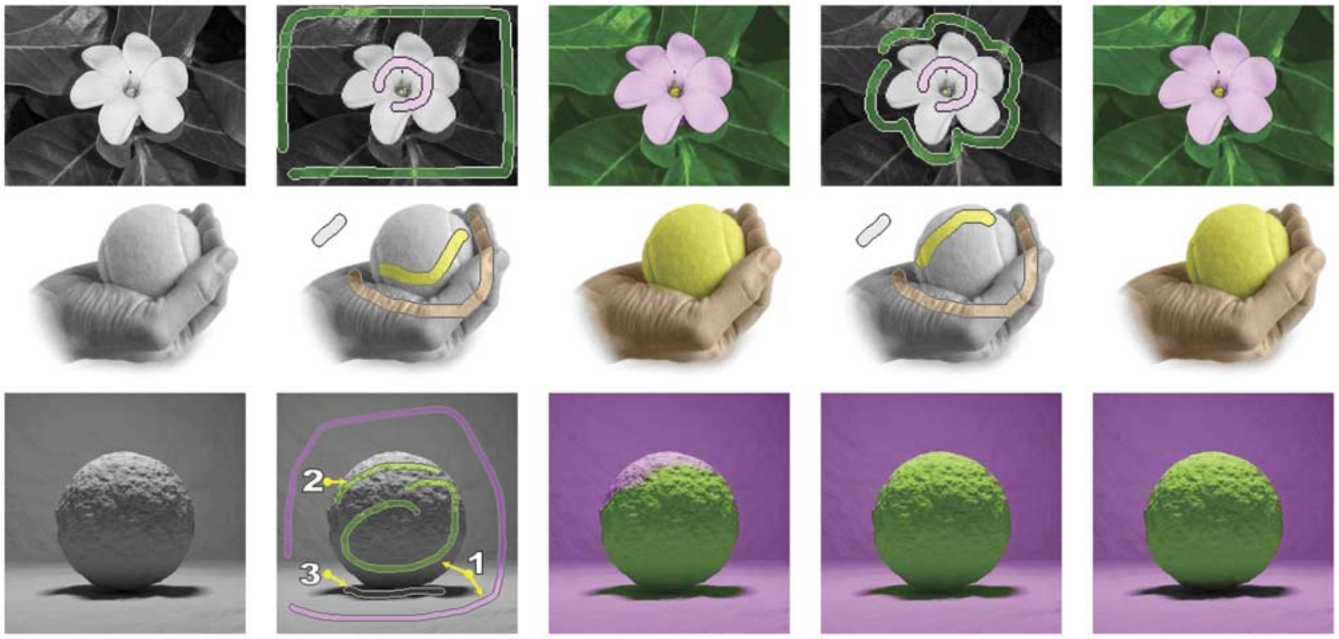


Fig. 4. Testing the robustness of the proposed algorithm. The first row shows an example where different sets of scribbles are used, obtaining visually identical results. The original monochromatic image is shown first, followed by the first set of scribbles and the resulting colorized result. This is followed by a second set of scribbles and the corresponding colorized results. Note how the two results are virtually identical. The next row repeats the same experiment for a different image. Next, we show (third row) how the result is evolving with the addition of new scribbles by the user. The original image is presented first, followed by the set of scribbles labeled by their order of application. The third figure shows the result when only the two scribbles labeled **1** are used. Then, scribbled **2** is added to improve the results on the top of the ball, obtaining the fourth figure. Finally, the scribble labeled **3** was added to keep the shadow gray instead of green, and the result is provided in the last figure.

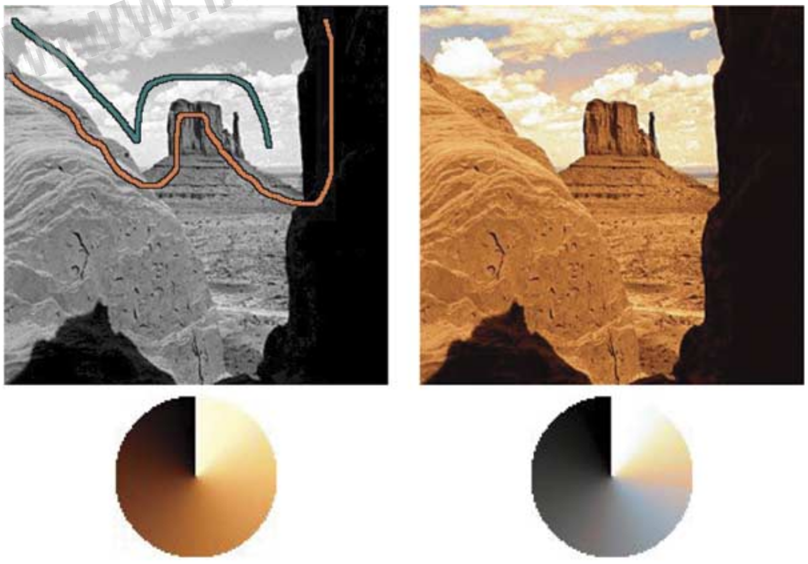


Fig. 5. Colorization with palettes. After the scribble is propagated, the gray value is used to key the palette. The original image is followed by the colored one in the first row. The second row shows the palettes for the mountain and the sky, respectively.

since these provide zero or very limited weight/influence in their formulation.

In Fig. 4, we study the robustness of our proposed approach with respect to the scribbles placement (location of the set Ω_c). Before describing these examples, let us provide some basic comments that will help to understand the observed robustness of this technique. Assume that we know the "ideal" position to place a scribble (see Section IV for more on this). What happens if, instead of placing this scribble at the optimal place, we place it at a different location? If the ideal scribble and the placed one are both inside the same object, and the region between them is relatively homogenous, then using our gradient weighted metric, the distance between the two scribbles will be relatively small. From the triangle inequality, we can then bound the distance from the placed scribble to a given pixel to be colored, simply using the sum of the distance from the ideal scribble to such pixel and the distance between the scribbles. Since the latter is small, then the distance from the ideal

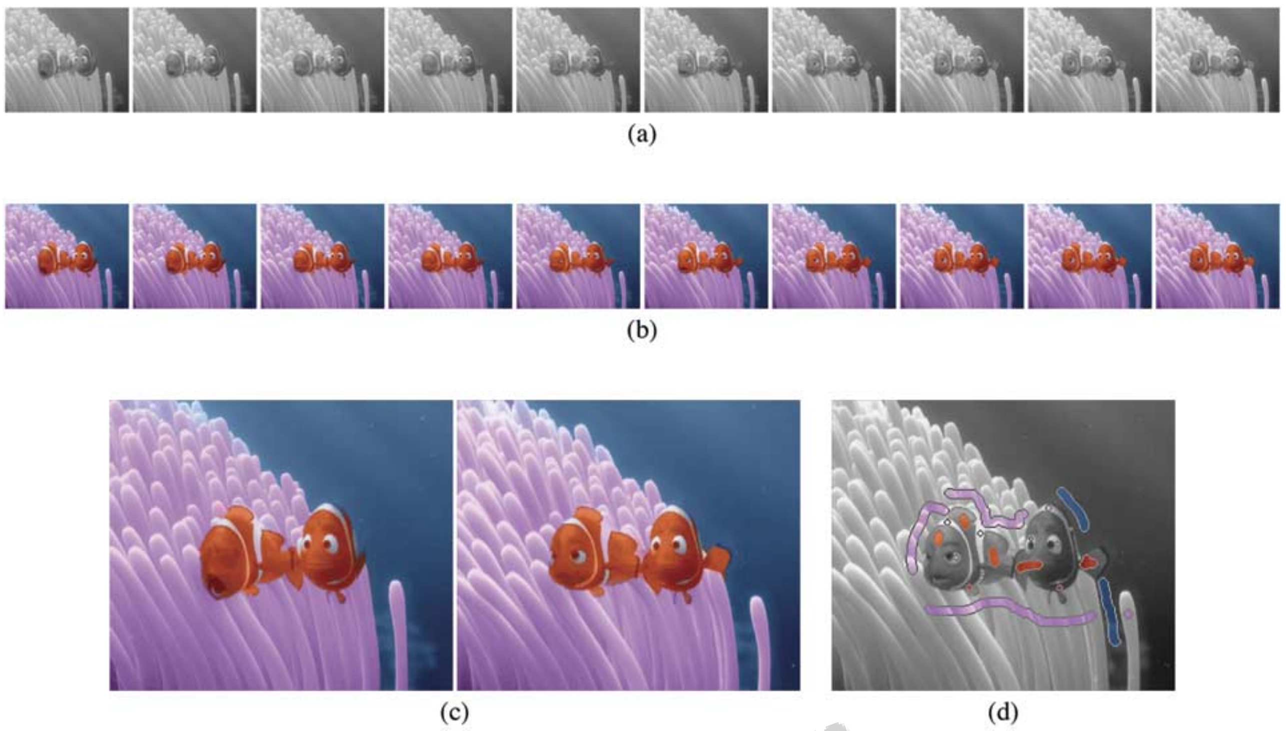


Fig. 6. Animated video colorization example. (a) Given the ten-frame grayscale sequence, (b) our algorithm provides a colorized video. (c) The first and last frame of the colorized video are enlarged to show the color content. (d) To colorize the ten frames, all that was needed were a few chrominance scribbles on a single frame. The user marked chrominance scribbles on the seventh frame of this sequence. The size of each frame is 312 × 264, and the algorithm total run time for the whole sequence is 6.8 s. The movie can be found on the web site for this project.

scribble and the one from the placed one to the pixel to be colored are very similar, and, as such, the result of the colorization algorithm. If the placed scribble is located "on the wrong side of the edge" (or with high gradients between it and the ideal scribble), then, of course, the distance between the ideal and the placed scribbles will be large and as such the algorithm will produce very different results. Of course, this will almost mean an intentional mistake by the user. Moreover, interactive colorization is possible thanks to the fast speed of the proposed technique, thereby permitting the user to correct errors and to add or move scribbles as needed (this can be easily experimented with the public domain software mentioned above). The first row of Fig. 4 shows an example of an image colored with different sets of scribbles. Notice how the results are visually indistinguishable. This is repeated for a second example in the next row. The last row of the figure shows the evolution of the result as the user adds scribbles, a common working scenario for this type of application, which is allowed by the speed and robustness of our proposed technique (this is also part of the public domain implementation mentioned above).

Fig. 5 shows a simple example of special effects that can be easily obtained with our proposed framework, colorization with palettes. After the scribble is propagated, the gray value is used to key the palette. The original image is followed by the colored one on the first row. The second row shows the palettes for the mountain and the sky, respectively.

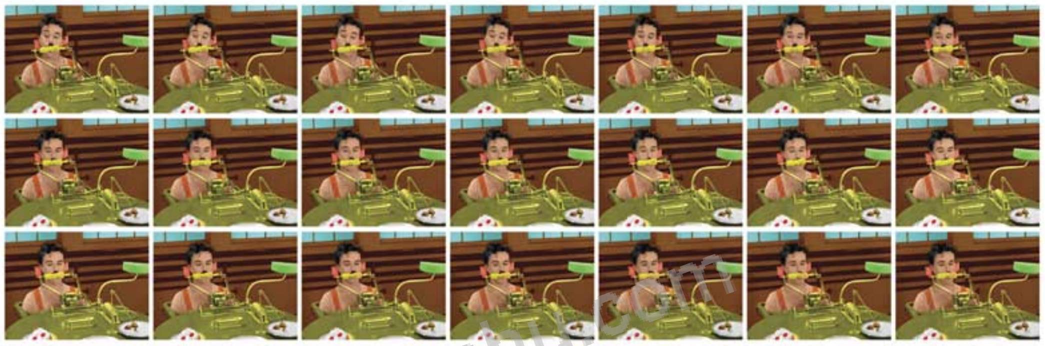
Figs. 6 and 7 show how our technique can be applied to video. Given a grayscale video and some chrominance scribbles anywhere in the video, our algorithm colors the whole video within seconds. This is a significant computational complexity reduction compared to [17] and [26], where not only is each frame computed significantly faster, but there is also no need for optical flow. Fig. 6 demonstrates the colorization of an animated scene from the movie *Finding Nemo*. Fig. 7 shows a colorization example of an old film. In both examples, we obtained very good results just by marking a few chrominance scribbles in a single frame.

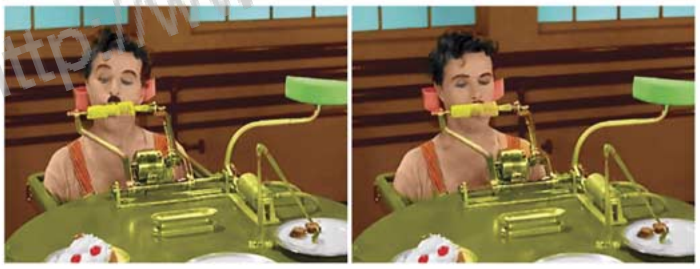
The *blending factor* is the only free parameter of the algorithm as currently implemented. We set it to be $b = 4$ for all examples in this paper and the additional ones in the aforementioned web page. Better results may have been archived selecting a different value per image and video, although we did not find this necessary to obtain the high quality results presented here.

A. Recolorization and Extensions

Recolorization is the art of replacing the colors of an image by new colors. Fig. 8 shows that it is possible to change colors of an existing image or video just by slightly modifying our original colorization algorithm. When colorizing grayscale images, we based the process on the assumption that homogeneity of the grayscale indicates homogeneity in the chrominance. In recolorization, we have more clues about how the chrominance should vary. We assume that homogeneity in the original chrominance indicates homogeneity in the new chrominance. The chrominance propagation can be based on the grayscale assumption or the original color assumption or both, as demonstrated in Fig. 9. The intrinsic distance can also be measured on the *Cb* and *Cr* channels rather than just on the intensity channel as done in (1). This is done simply by replacing Y by Cb or

(a)


(b)


(c)

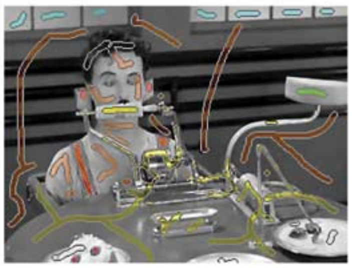

(d)

Fig. 7. Video colorization example. (a) Given a 21 frame grayscale sequence, (b) our algorithm provides a colorized video. (c) The first and last frame of the colorized video are enlarged to show the color content. (d) To colorize the 21 frames, all that was needed are a few chrominance scribbles on a single frame. The user marked chrominance scribbles on the eleventh frame of this sequence. The size of each frame is 320 × 240, and the algorithm total run time for the whole sequence is 14 s. The movie can be found in the web site for this project.

Cr in this equation, thereby using the gradients of these (now available) chroma channels to control the blending weight of the new color. Video recolorization examples can be seen in the aforementioned web site.

Recolorization is just one possible extensions of our method. It is also possible to further generalize it by defining the measurement medium $M(x,y,\tau) : \Omega\times[0,T) \rightarrow \Re$ on which the intrinsic distance is measured. M can be any channel of the input image, a mix of the channels, or any other data that will make sense for weighting the intrinsic distance. The blending medium $\mathcal{B}(x,y,\tau) : \Omega \times [0,T) \rightarrow \Re$ is then the data that is actually blended. Both in colorization and recolorization, we selected $\mathcal{B}$ to be the chrominance. Yet, $\mathcal{B}$ can be any image processing effect or any data that makes sense to blend for a given application. Fig. 10 follows this idea and gives an example of object brightness change.

IV. Concluding Remarks

In this paper, we have introduced a fast colorization algorithm for still images and video. While keeping the quality at least as good as previously reported algorithms, the introduced technique manages to colorize images and movies within a second, compared to other techniques that may reach several minutes. The proposed approach needs less manual effort than techniques such as those introduced in [17] and [23] and can be used interactively due to its high speed. We also showed that simple

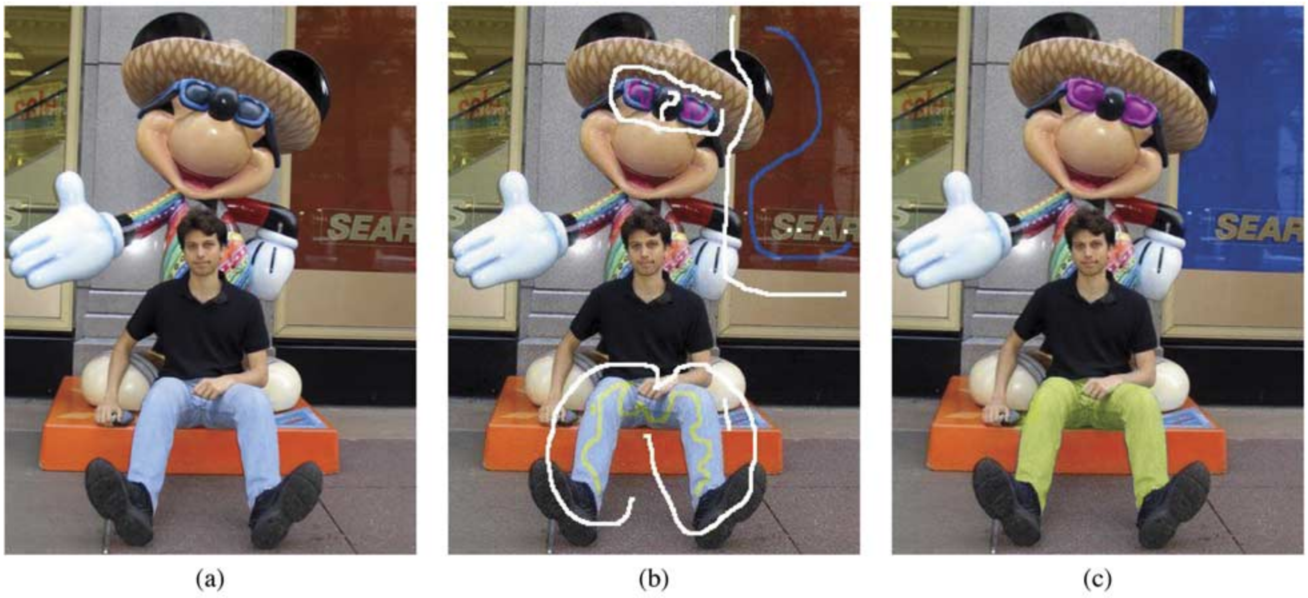


Fig. 8. Recolorization example. (a) Original color image, (b) scribbles, and (c) image after recolorization (note the pants, glasses, and window). The white scribbles represent a "mask," used to keep the original colors.

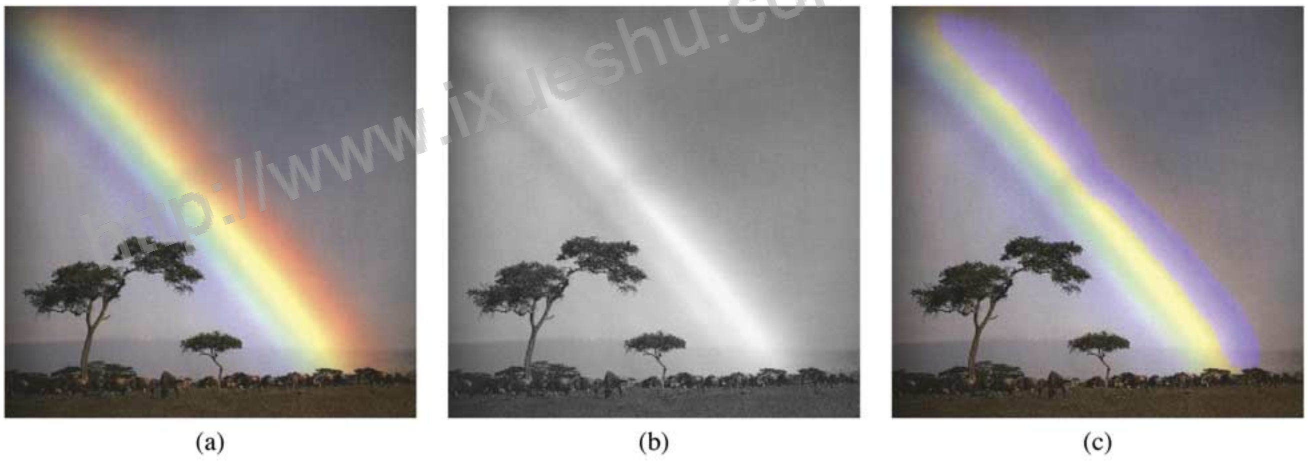


Fig. 9. Recolorization example using the *Cr* channel for measurement (M) rather than the intensity channel. (a) Original color image; (b) intensity channel of the image. It can be clearly seen that the intensity image does not contain significant structural information on the red rainbow strip (this is quite a unique example of this effect). On the other hand, both the *Cb* and *Cr* do change significantly between the stripes and can, therefore, be used for recolorization. (c) Recolored image, the red stripe of the rainbow was replaced by a purple one.

modifications in the algorithm lead to recolorization and other special effects [30].

A number of directions can be pursued as a result of the framework introduced here. First, as mentioned before, other special effects can be obtained following this distance-based blending of image attributes approach. We are working on including in this framework a crude (fast) computation of optical flow, to use mainly when the motion is too fast for the weighted distance used in the examples presented here. Another interesting problem is to investigate the use of this colorization approach for image compression. As shown in the examples presented here, a few color samples are often enough to produce visually pleasant results. This can be exploited for compression, in the same spirit as done in [20], where the encoder voluntarily drops information that can be efficiently reconstructed by the decoder. In order to do this, we need to understand what is the simplest set of color scribbles that when combined with our algorithm, manages to reconstruct the color without any visual artifacts. In the same spirit, for the colorization of originally mono-chromatic data, or the recoloring of color data, it is important to understand the best position to place the scribbles so that the user's effort is minimized. For example, an edge map on the luminance data might help to guide the user. Although, as demonstrated with the examples provided in this paper and the additional ones in our web site, the robustness and speed of the algorithm make it possible to work in an interactive fashion, reducing/minimizing user intervention is always an important goal. Results in these research directions will be reported elsewhere.

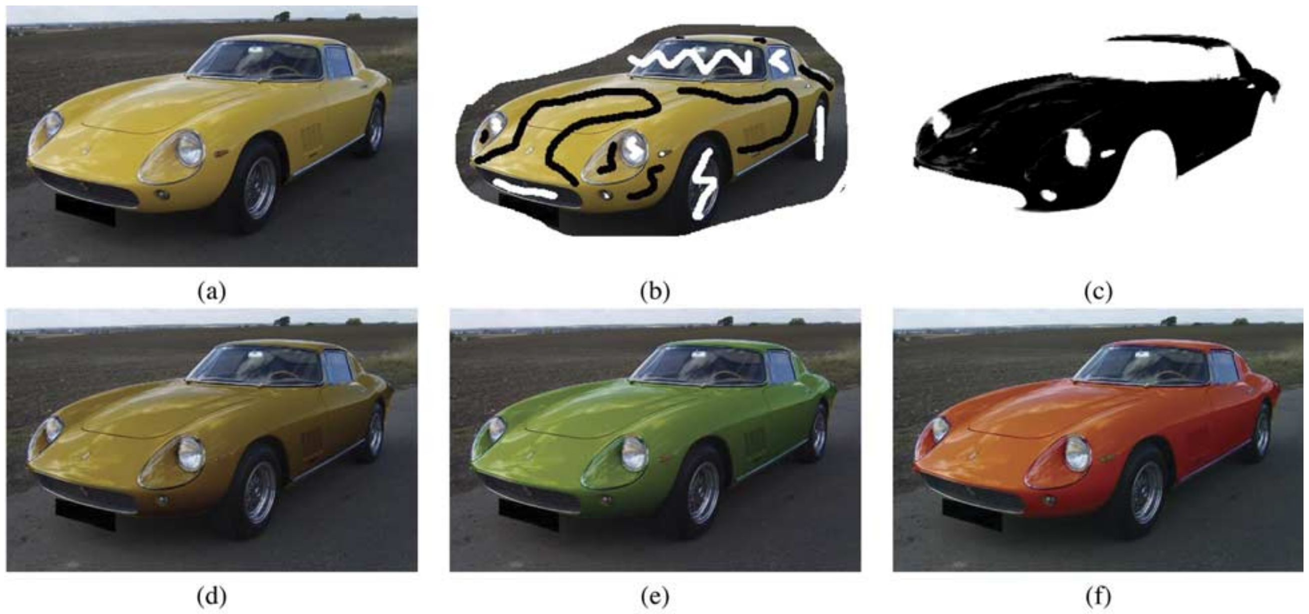


Fig. 10. Example of the generalization of our method to other image processing effects. (a) The original color image. Our goal is to change the color of the yellow car into a darker color. (b) We define the blending medium by roughly marking areas we do not wish to change in white and areas we do want to change in black; we do so by placing scribbles or by just marking whole areas. (c) Using our colorization method, we propagate the markings (white and black colors) and get a grayscale matte (we only keep the blending channel). With the matte, it is possible to apply an effect to the original image with a magnitude proportional to the gray level of the matte. In this case, we chose to change the brightness. (d) Image after applying the darkening. Note that the intensity only changed in the desired parts of the image. The darkening is done simply by subtracting the gray-level matte from the intensity channel, where white means no change and black is the maximum preselected change. (e) It is possible to further process the image using the same matte and (f) demonstrate this by similarly adding the gray-level matte to the *Cb* and *Cr* channels, respectively.

ACKNOWLEDGMENT

The authors would like to thank G. Brown, A. Bartesaghi, and K. Patwardhan for their inspiring and useful remarks.

REFERENCES

[1] R. k. Ahuja, K. Melhrhorn, J. B. Orlin, and R. E. Tarjan, "Faster algorithms for the shortest path problem," *J. AMC*, vol. 37, pp. 213–223, 1990.

[2] C. Ballester, V. Caselles, J. Verdera, and B. Rouge. A variational model for P + XS image fusion. presented at Workshop on Variational and Level Set Methods. [Online]. Available: http://www.iua.upf.es/vcaselles/

[3] *Museum of Broadcast Communication: Encyclopedia of Television*, [Online]. Available: http://www.museum.tv/archives/etv/C/htmlC/colorization/colorization.htm.

[4] V. Caselles, B. Coll, and J.-M. Morel, "Geometry and color in natural images," *J. Math. Imag. Vis.*, vol. 16, pp. 89–105, 2002.

[5] T. Chen, Y. Wang, V. Schillings, and C. Meinel, "Gray-scale image matting and colorization," in *Proc. Asian Conf. Computer Vision*, 2004, pp. 1164–1169.

[6] D. H. Chung and G. Sapiro, "On the level-lines and geometry of vector-valued images," *IEEE Signal Process. Lett.*, vol. 7, no. 9, pp. 241–243, Sep. 2000.

[7] P. Danielsson, "Euclidean distance mapping," *Comput. Graph. Image Process.*, vol. 14, pp. 227–248, 1980.

[8] E. Dijkstra, "A note on two problems in connection with graphs," *Numer. Math*, vol. 1, pp. 269–271, 1959.

[9] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Reading, MA: Addison-Wesley, 1987.

[10] J. Helmsen, E. G. Puckett, P. Collela, and M. Dorr, "Two new methods for simulating photolithography development in 3D," in *Proc. SPIE Microlithography*, vol. IX, 1996, p. 253.

[11] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," in *Proc. ACM SIGGRAPH Conf.*, 2001, pp. 327–340.

[12] T. Horiuchi, "Estimation of color for gray-level image by probabilistic relaxation," in *Proc. IEEE Int. Conf. Pattern Recognition*, 2002, pp. 867–870.

[13] T. Horiuchi and S. Hirano, "Colorization algorithm for grayscale image by propagating seed pixels," in *Proc. IEEE Int. Conf. Pattern Recognition*, 2003, pp. 457–460.

[14] (2003, Dec.) Univ. California, Los Angeles, CAM Rep.. [Online]. Available: http://www.math.ucla.edu/applied/cam/index.html

[15] R. Kimmel, "A natural norm for color processing," presented at the 3rd Asian Conf. Computer Vision, Hong Kong, Jan. 8–11, 1998.

[16] R. Kimmel and A. M. Bruckstein, "On regularized Laplacian zero crossings and other optimal edge integrators," *Int. J. Comput. Vis.*, vol. 53, no. 3, pp. 225–243, 2003.

[17] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *Proc. ACM SIGGRAPH Conf.*, 2004, pp. 689–694.

[18] D. MacAdam, *Sources of Color Science*. Cambridge, MA: MIT Press, 1970.

[19] P. Marshall, "Any Color You Like," [Online]. Available: http://photography.about.com/library/weekly/aa061 002a.htm, Jun. 11, 2002.

[20] S. Rane, G. Sapiro, and M. Bertalmio, "Structure and texture filling-in of missing image blocks in wireless transmission and compression applications," *IEEE Trans. Image Process.*, vol. 12, no. 2, pp. 296–303, Feb. 2003.

[21] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Comput. Graph. Appl.*, vol. 21, no. 5, pp. 34–41, May 2001.

[22] G. Sapiro, *Geometric Partial Differential Equations and Image Processing*. Cambridge, U.K.: Cambridge Univ. Press, Jan. 2001.

[23] ——, "Inpainting the colors," in *IMA Preprint Series 1979*: Inst. Math. Appl., Univ. Minnesota, Minneapolis, May 2004.

[24] J. Sethian, "Fast marching level set methods for three-dimensional photolithography development," presented at the SPIE Int. Symp. Microlithography, Santa Clara, CA, Mar. 1996.

[25] J. A. Sethian, "A fast marching level-set method for monotonically advancing fronts," *Proc. Nat. Acad. Sci.*, vol. 93, no. 4, pp. 1591–1595, 1996.

[26] D. Sýkora, J. Buriánek, and J. Zára, "Unsupervised colorization of black-and-white cartoons," in *Proc. 3rd Int. Symp. NPAR*, Annecy, France, Jun. 2004, pp. 121–127.

[27] J. N. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," *IEEE Trans. Autom. Control*, vol. 40, no. 10, pp. 1528–1538, Oct. 1995.

[28] T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to grayscale images," in *Proc. ACM SIGGRAPH Conf.*, 2002, pp. 277–280.

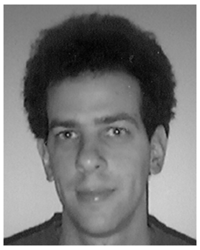
[29] L. Yatziv, A. Bartesaghi, and G. Sapiro. (2005, Feb.) O(N) Implementation of the Fast Marching Algorithm. [Online]. Available: http://www.ima.umn.edu/preprints/feb2005/feb2005.html

[30] L. Yatziv and G. Sapiro, "Image and Video Data Belnding Using Intrinsic Distances," Patent Pending, 2005.

[31] H. K. Zhao, "Fast sweeping method for Eikonal equations," *Math. Comput.*, vol. 74, pp. 603–627, 2004.

Liron Yatziv was born in Haifa, Israel, on January 21, 1975. He received the B.Sc. degree (summa cum laude) from the Department of Electrical Engineering, Technion–Israel Institute of Technology, Haifa, in 2001, and the M.Sc. degree from the University of Minnesota, Minneapolis, in 2005.

He is currently an Associate Member of Technical Staff, Siemens Corporate Research, Princeton, NJ. In his current position at Siemens, he is responsible for developing image processing algorithms that will be used in advanced medical equipment. His current interests are in the fields of image processing and algorithms, about which he published several papers.



Guillermo Sapiro was born in Montevideo, Uruguay, on April 3, 1966. He received the B.Sc. (summa cum laude), M.Sc., and Ph.D. degrees from the Department of Electrical Engineering, Technion–Israel Institute of Technology, Haifa, in 1989, 1991, and 1993, respectively.

After postdoctoral research at the Massachusetts Institute of Technology, Cambridge, he became a Member of the Technical Staff at the research facilities of HP Labs, Palo Alto, CA. He is currently with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis. He works on differential geometry and geometric partial differential equations, both in theory and applications in computer vision, computer graphics, medical imaging, and image analysis. He recently coedited a special issue on image analysis in the *Journal of Visual Communication and Image Representation*. He has authored and coauthored numerous papers in image anaylsis and he has written a book.

Dr. Sapiro is a member of SIAM. He recently co-edited a special issue of IEEE TRANSACTIONS ON IMAGE PROCESSING. He was awarded the Gutwirth Scholarship for Special Excellence in Graduate Studies in 1991, the Ollendorff Fellowship for Excellence in Vision and Image Understanding Work in 1992, the Rothschild Fellowship for Postdoctoral Studies in 1993, the Office of Naval Research Young Investigator Award in 1998, the Presidential Early Career Awards for Scientist and Engineers (PECASE) in 1998, and the National Science Foundation Career Award in 1999.