FAST IMAGE AND VIDEO COLORIZATION USING CHROMINANCE BLENDING

By

Liron Yatziv

and

Guillermo Sapiro

IMA Preprint Series # 2010

(December 2004)

INSTITUTE FOR MATHEMATICS AND ITS APPLICATIONS

UNIVERSITY OF MINNESOTA

514 Vincent Hall
206 Church Street S.E.
Minneapolis, Minnesota 55455–0436

Phone: 612/624-6066 Fax: 612/626-7370
URL: http://www.ima.umn.edu

Fast Image and Video Colorization using Chrominance Blending

Liron Yatziv and **Guillermo Sapiro**

Electrical and Computer Engineering

University of Minnesota

Minneapolis, MN 55455

{liron,guille}@ece.umn.edu

Abstract

Colorization, the task of coloring a gray-scale 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 so 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 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 here introduced included the capability of changing colors of an existing color image or video as well as changing the underlying luminance.

Keywords: Colorization, recolorization, gradient, intrinsic distance, interpolation, chrominance blending.

EDICS: 2-INTR

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 [14]. It was practiced in motion pictures in the early 1900's by the French Company Pathe, where many films were colored by hand. It was widely practiced also for filmstrips into the 1930s. Computer-assisted process was first introduced by Wilson Markle in 1970 for adding colors to black and white movies [2].

As neatly presented by Sykora *et al.* [19] (their work also includes an outstanding overview of the literature on the subject), various early computer-based colorization techniques include straight forward approaches such as *luminance keying* [6]. This method uses a user-defined look-up table which transforms gray-scale into color. Welsh *et al.* [21], inspired by work of Reinhard *et al.* [15] and Hertzmann *et al.* [8], extended this idea by matching luminance and texture rather than just the gray-scale values.

Chen *et al.* [4] used manual segmentation to divide the gray-scale 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.* [19] 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 gray-scale image indicates homogeneity in the color. In other words, as detailed in [16], the geometry of the image is provided by the geometry of the gray-scale information (see also [3], [5], [11]). Often in these methods, in addition to the gray-scale data, color hints are provided by the user. Horiuchi [9] used a probabilistic relaxation method while Levin *et al.* [12] solved an optimization problem that minimizes a quadratic cost function of the difference of color between a pixel and it's weighted average neighborhood colors. Sapiro [16] proposed to inpaint the colors constrained by the gray-scale gradients and the color scribbles that serve as boundary conditions. The method reduces to solving linear or non-linear 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 [10], 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 [12], the technique here proposed is easily extended to video without the optical flow computation, further improving in the computational cost, at no sacrifice in the image quality.

The scheme here proposed in 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

Similarly to other colorization methods, e.g., [12], [16], we use luminance/chrominance color systems. We present our method in the *YCbCr* color space, although other color spaces such as *YIQ* or *YUV* could be used as well. Moreover, work can be done also directly on the RGB space. Let $Y(x, y, \tau) : \Omega \times [0, T) \to \Re^+$ 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 $Cb(x, y, \tau) : \Omega \times [0, T) \to \Re^+$ and $Cr(x, y, \tau) : \Omega \times [0, T) \to \Re^+$ 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 [12]). These values are often provided by the user or borrowed from other data.

Let s and t be two points in Ω and let $C(s) : [0, 1] \to \Omega \backslash \Omega_c$ be a curve in Ω. Let also $C_{s,t}$ be 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:

$$d(s, t) := \min_{C_{s,t}} \int_{s=0}^{1} |\nabla Y \cdot \dot{C}(s)| ds. \tag{1}$$

This intrinsic distance gives a measurement of how “flat” is the flattest curve between any two points in the luminance channel.[1]

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 [3], [5], [11]. 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 has been reported in the above mentioned works, there is a close relationship between the geometry of the luminance and chrominance channels. Exploiting this assumption, a change

[1]This geodesic distance can be efficiently and accurately computed using recently developed fast numerical techniques [7], [17], [18], [20]. We found that for the application at hand, even simpler techniques such as a *best first* one, integrated in the pseudo-code below, are sufficient.

in luminance causes a related change in chrominance. This has been used in different fashions in [12], [16], as well as in [1] 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.[2]

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 to any point in Ω. We define the intrinsic distance from a certain chrominance c as the minimum distance from any point of the same chrominance c in Ω_c:

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

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:

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

where $chrominances(A)$ stands for all the different unique chrominance in the region A and $W(\cdot)$ is a function of the intrinsic distance that translates it into a blending weight. The function $W(\cdot)$ should hold some basic properties:

1) $\lim_{r \to 0} W(r) = \infty$
2) $\lim_{r \to \infty} W(r) = 0$
3) $\lim_{d \to \infty} W(d+c)/W(d) = 1$

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 even be the blending of all chrominance. For the experiments reported below we used

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

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

[2]It is important to notice that the goal of colorization is not to restore the original color of the image or scene, but as in image inpainting, to produce visually pleasant and compelling colored images.

A. Algorithm Pseudocode

To complete the description of the proposed colorization technique, we provide a pseudocode that further emphasizes the simplicity of the proposed technique:

- **Input:**
  - Gray-scale video/image.
  - List of observed pixels Ω_c and their chrominance.
  - Empty pixels in Ω $(\Omega \backslash \Omega_c)$.
- **Output:** Colored video/image.
- **Definitions:**
  - A **pixel** contains a list of chrominances, the distance of the source of each chrominance, and it's gray-scale value.
  - A **link** points to a pixel, and contains a chrominance and the intrinsic distance from the chrominance origin in Ω_c.
- **The algorithm:**

```
1) L $\leftarrow$ {all possible links to neighboring pixels of $\Omega_c$}
2) while $L \neq \emptyset$
    a) $\lambda \leftarrow$ the link with smallest distance in L
    b) L $\leftarrow$ L $\setminus \lambda$
    c) p $\leftarrow$ the pixel $\lambda$ links to
    d) if p does not contain chrominance of $\lambda$
        i) add chrominance and distance of $\lambda$ to p
        ii) L $\leftarrow$ L $\bigcup$ {all links to neighboring pixels of p using the same chrominance of $\lambda$}
3) for all pixels in $\Omega$ set color by using the gray-level value and the chrominance generated
   by the blending of chrominance following Equation (3)
```

B. Performance and Relaxation

The described colorization algorithm has average time and space complexity of $O(|\Omega| \cdot |chrominances(\Omega_c)|)$. 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 of observed 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 humans perception of 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 2 or 3 most significant chrominance to get satisfactory results. Such a relaxation reduces both 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 [13] 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 step 2(d) in the pseudocode presented in Section II-A.

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/

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 512KB RAM running under Windows 2000. Figure 1 shows examples of still image colorization using our proposed algorithm. The algorithm run time for all the examples in Figure 1, measured once the images where loaded into memory, is less than 7 μsec per pixel.

Figures 2 and 3 compare our method with the one recently proposed by Levin *et al.* [12] (this work partially inspired our own). The method minimizes the difference between a pixel's color and the weighted average color of it's 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 Figure 2 we observe that we achieve the same visual quality at a fraction of the computational cost (more comparisons are provided in the above mentioned web site, all supporting the same finding). Overall the method proposed in [12] performs very well on many images, yet Figure 3 demonstrates that it can perform poorly when colorizing relatively far from the provided color constrains. In order to match visual quality with our technique, the method proposed in [12] needs more user input, meaning additional color scribbles. We also found that the inspiring technique developed in [12] has a sensible scale parameter and often fails at strong edges, since these provide zero or very limited weight/influence in their formulation.

Figures 4 and 5 show how our technique can be applied to video. Given a gray-scale 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 [12] (and [19]), where not only each frame is computed significantly faster, but also there is no need for optical flow. Figure 4 demonstrates the colorization of an animated scene from the movie *Finding Nemo*. Figure 5 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. In order to demonstrate how automatic and robust the algorithm is, we set it to be $b = 4$ for all examples in this paper. Better results may have been archived if we would have selected a different value per image and video.

A. Recolorization and Extensions

Recolorization is the art of replacing the colors of an image by new colors. Figure 6 shows that it is possible to change colors of an existing image or video just by slightly modifying our original colorization algorithm. When colorizing gray-scale images we based the process on the assumption that homogeneity of the gray-scale 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 gray-scale assumption or the original color assumption or both, as demonstrated in Figure 7. The intrinsic distance can be measured also on the *Cb* and *Cr* channels rather than just on the intensity channel as done in Equation 1.

Recolorization is just one possible extensions of our method. It is also possible to further generalize

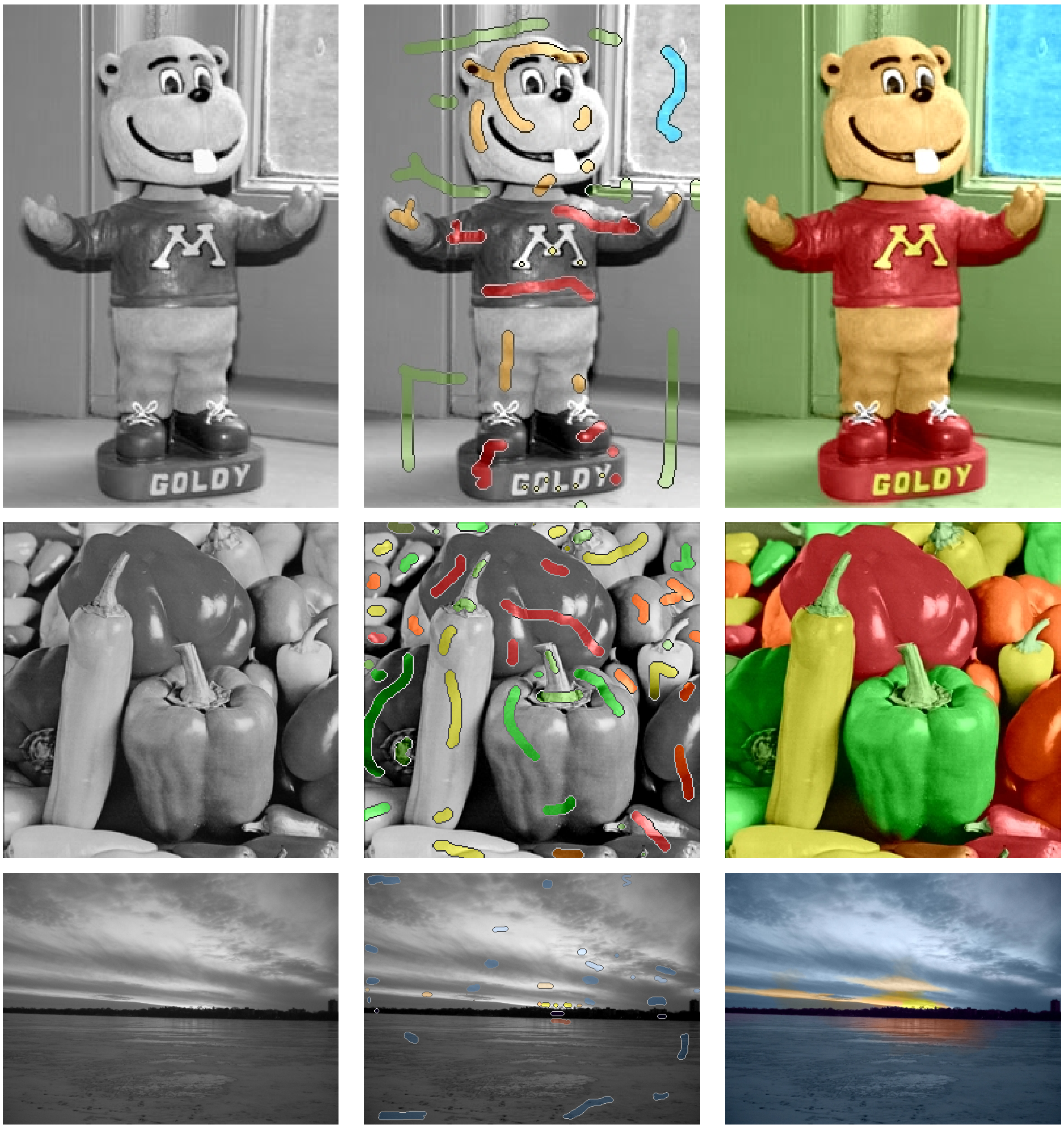


Fig. 1. Still image colorization examples. Given a gray-scale image (left), the user marks chrominance scribbles (center), and our algorithm provides a colorized image (right). The image size/run time top to bottom are: 230x345/less than 0.42 seconds, 256x256/less than 0.36 seconds, and 600x450/less than 1.73 seconds

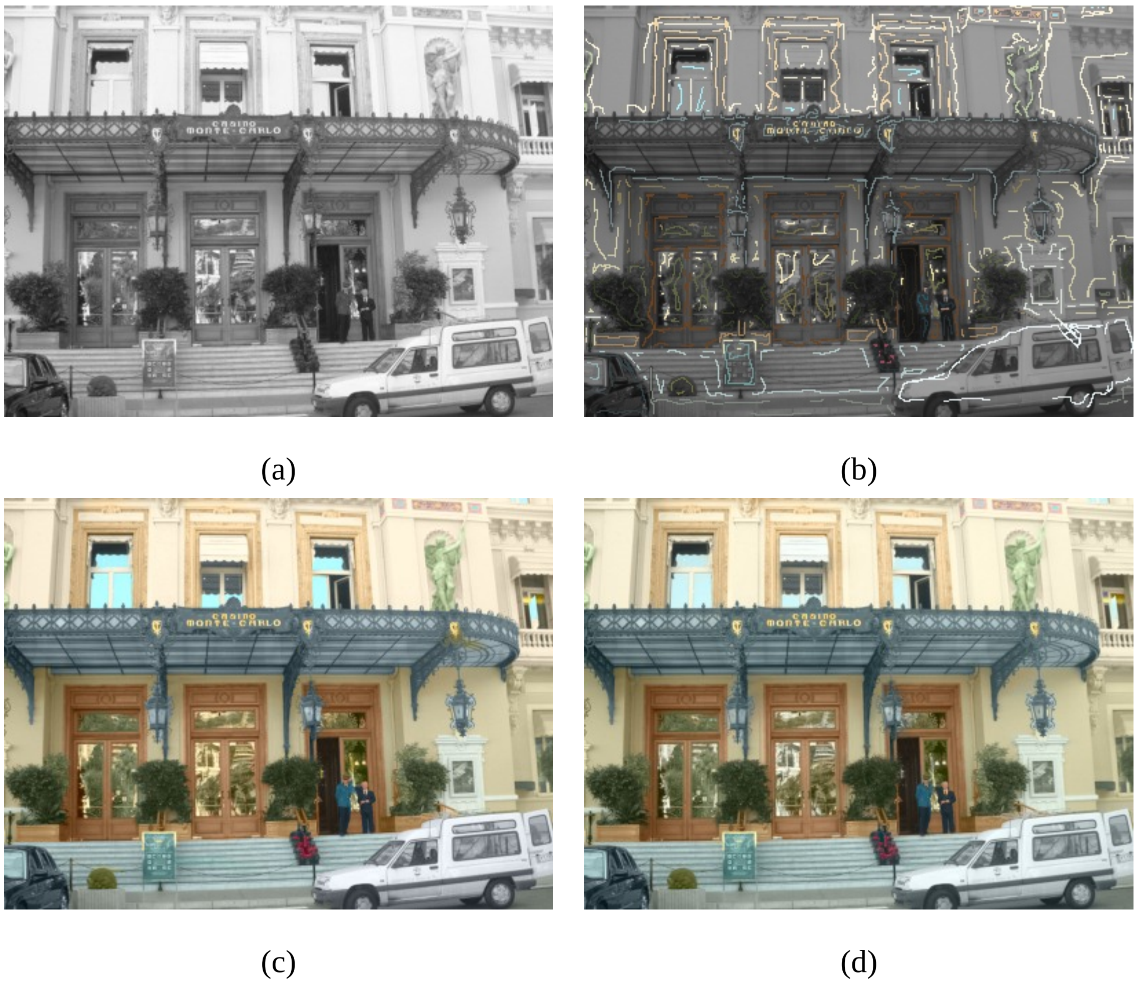


Fig. 2. Comparison of visual quality with the technique proposed in [12]. (a) The given gray-scale image, (b) the user marks chrominance scribbles (the selected scribbles are obtained from the work in [12]), (c) our algorithm results with CPU run-time of 0.76 sec, (d) Levin *et al.* approach with CPU run-time of 24 sec using their supplied fast implementation based on multi-grid solver (611 sec are needed for their standard Matlab implementation). We observe the same quality at a significantly reduced computational cost.

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. Figure 8 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

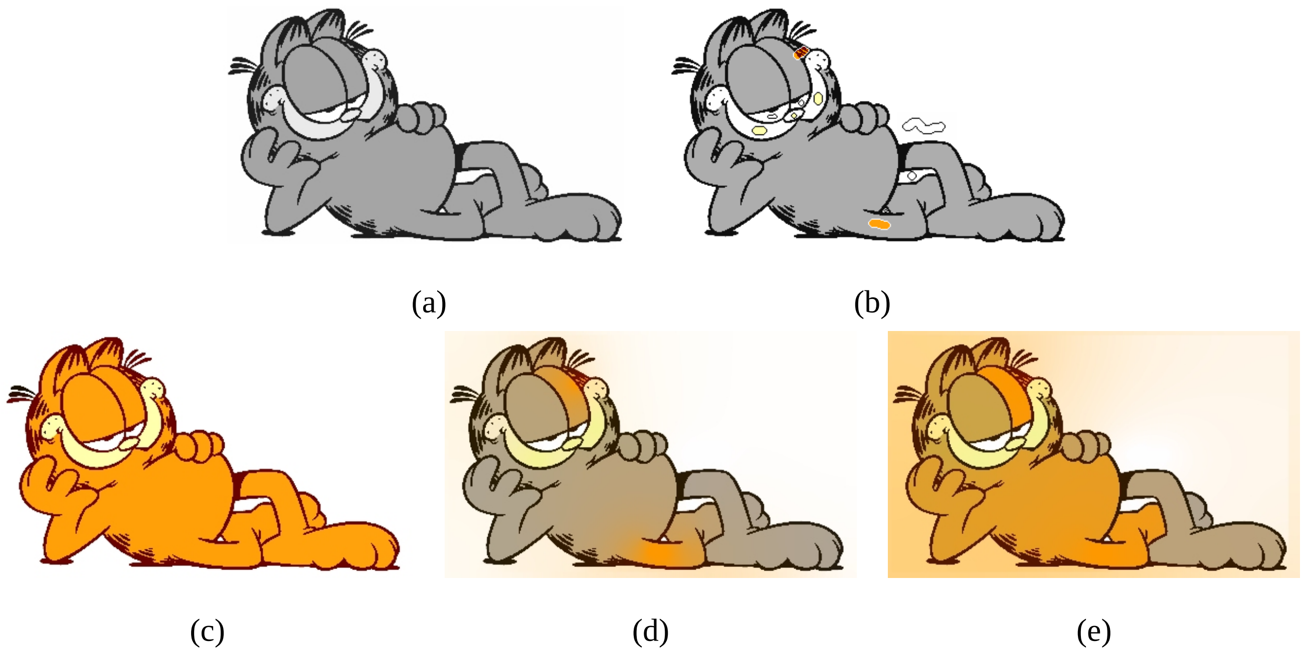


Fig. 3. Comparison of visual quality with the technique proposed in [12]. (a) The given gray-scale image (400x240), (b) the user marks chrominance scribbles, (c) our algorithm results with CPU run-time of 1.20 sec, (d) Levin *et al.* approach with CPU run-time of 22.8 sec using their supplied fast implementation of multi-grid solver, (e) Levin *et al.* approach using a slower exact Matlab least squares solver also provided by the authors.

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 [12], [16], and can be used interactively due to its high speed. We also showed that simple modifications in the algorithm lead to recolorization and other special effects.

V. Acknowledgments

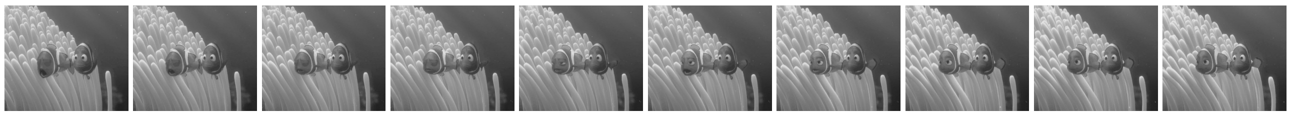
The authors would like to thank Gustavo Brown, Alberto Bartesaghi, and Kedar Patwardhan for their inspiring and useful remarks. This work is partially supported by the Office of Naval Research, the National Science Foundation, and the National Geospatial-Intelligence Agency.

References

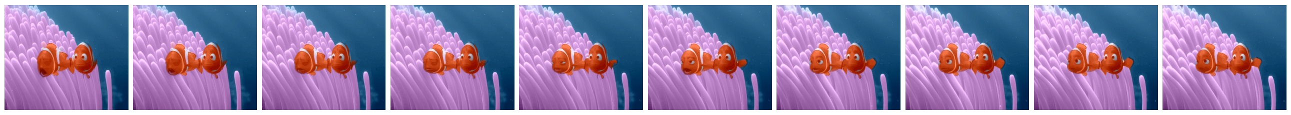
[1] C Ballester, V. Caselles, J.Verdera, and B. Rouge, “A variational model for P+XS image fusion,” *Workshop on Variational and Level Set Methods*, Nice, October 2003.

[2] G. Burns, “Colorization,” Museum of Broadcast Communication: Encyclopedia of Television.

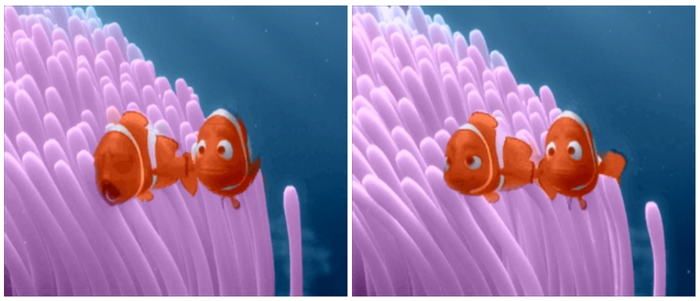
[3] V. Caselles, B. Coll, and J-M. Morel, “Geometry and color in natural images,” *Journal of Mathematical Imaging and Vision* **16**, pp. 89-105, 2002.



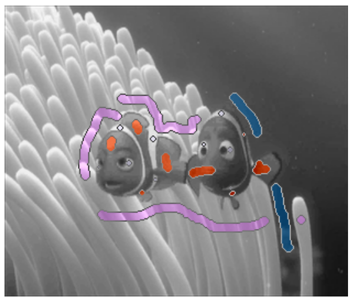
(a)



(b)



(c)



(d)

Fig. 4. Animated video colorization example. (a) Given the 10 frame gray-scale 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 were a few chrominance scribbles on a single frame. The user marked chrominance scribbles on the 7th frame of this sequence. The size of each frame is 312x264 and the algorithm total run time for the whole sequence is 6.8 seconds. The movie can be found in the web site for this project.

[4] T. Chen, Y. Wang, V. Schillings, and C. Meinel, "Gray-scale image matting and colorization," *Proceedings of Asian Conference on Computer Vision*, 2004, pp. 1164-169.

[5] D. H. Chung and G. Sapiro, "On the level-lines and geometry of vector-valued images," *IEEE Signal Processing Letters* **7**, pp. 241-243, September 2000.

[6] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd Edition, Addison-Wesley Publishing, Reading, Massachusetts, 1987.

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

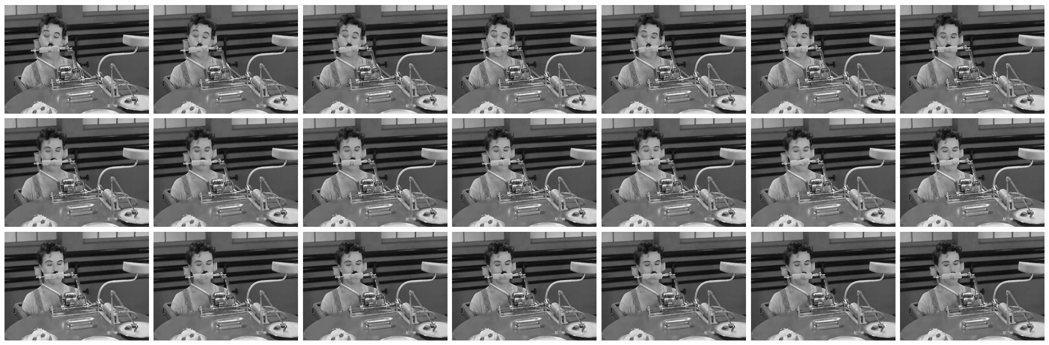
[8] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," *ACM SIGGRAPH 2001 Conference Proceedings*, 2001, pp. 327-340.

[9] T. Horiuchi, "Estimation of color for gray-level image by probabilistic relaxation," *Proceedings of IEEE International Conference on Pattern Recognition*, 2002, pp. 867-70.

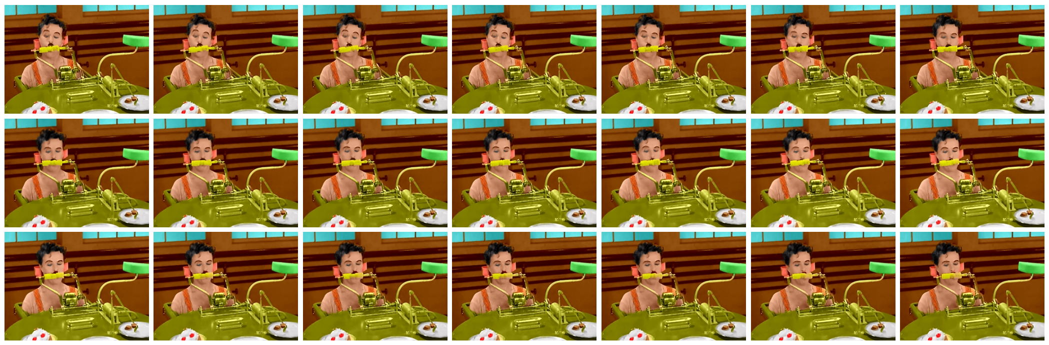
[10] T. Horiuchi and S. Hirano, "Colorization algorithm for grayscale image by propagating seed pixels," *Proceedings of IEEE International Conference on Pattern Recognition*, 2003, pp. 457-60.

[11] R. Kimmel, "A natural norm for color processing," *Proc. of Third Asian Conf. on Computer Vision*, Hong Kong, January 8-11, 1998.

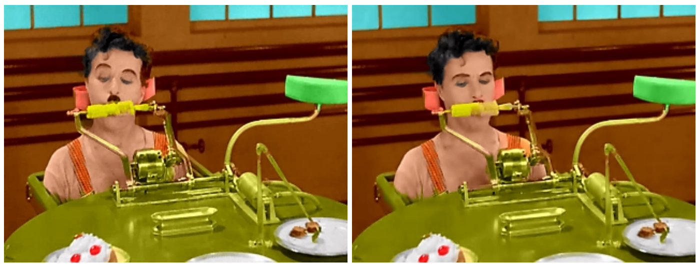
[12] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM SIGGRAPH 2004 Conference Proceedings*, 2004, pp.



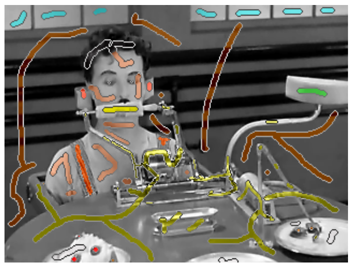
(a)



(b)

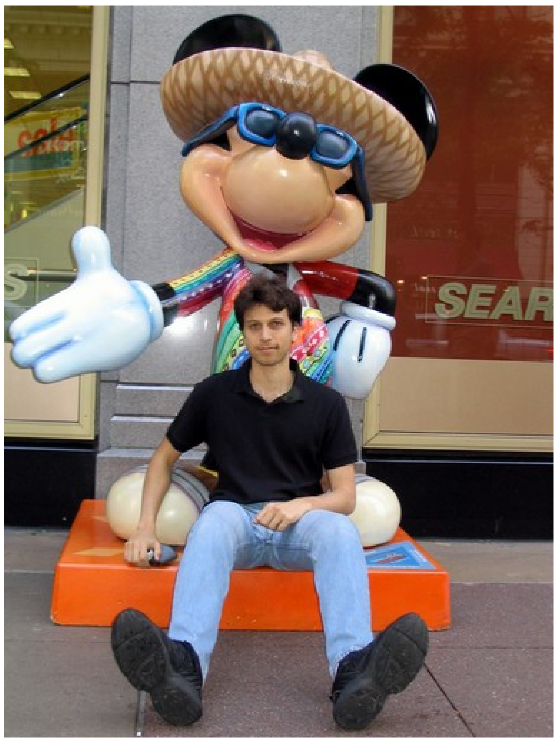


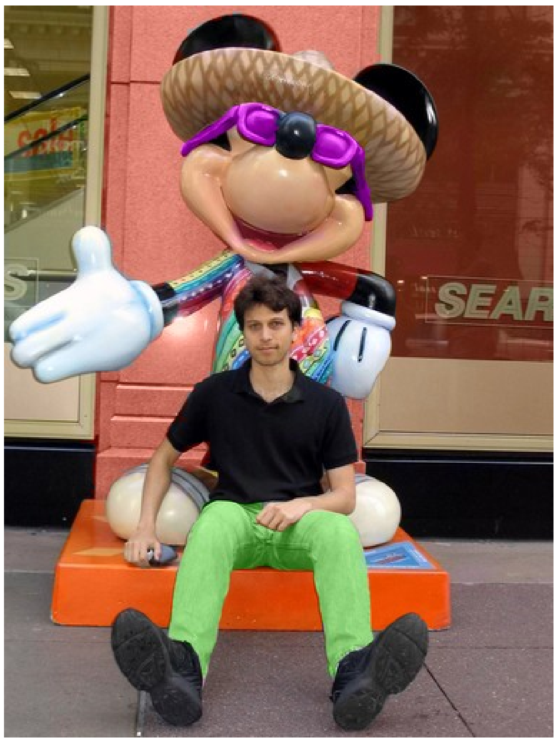
(c)



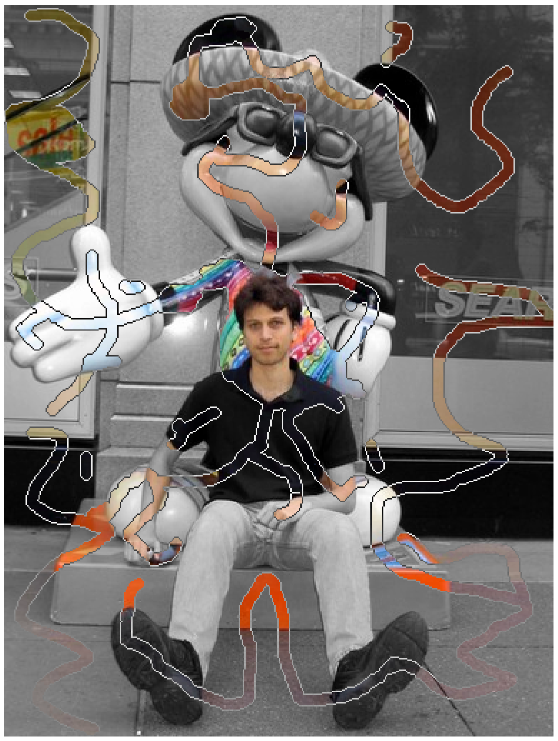
(d)

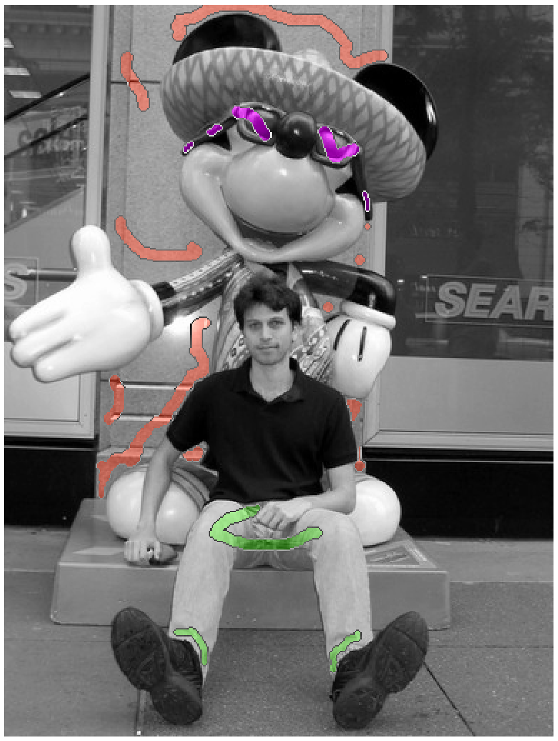
Fig. 5. Video colorization example. (a) Given a 21 frame gray-scale sequence, (b) our algorithm provides a colorized video, (c) the first and last frame of the colorized video are enlarged to show the color content, (d) to colorize the 21 frames all that was needed are a few chrominance scribbles on a single frame. The user marked chrominance scribbles on the 11th frame of this sequence. The size of each frame is 320x240 and the algorithm total run time for the whole sequence is 14 seconds. The movie can be found in the web site for this project.





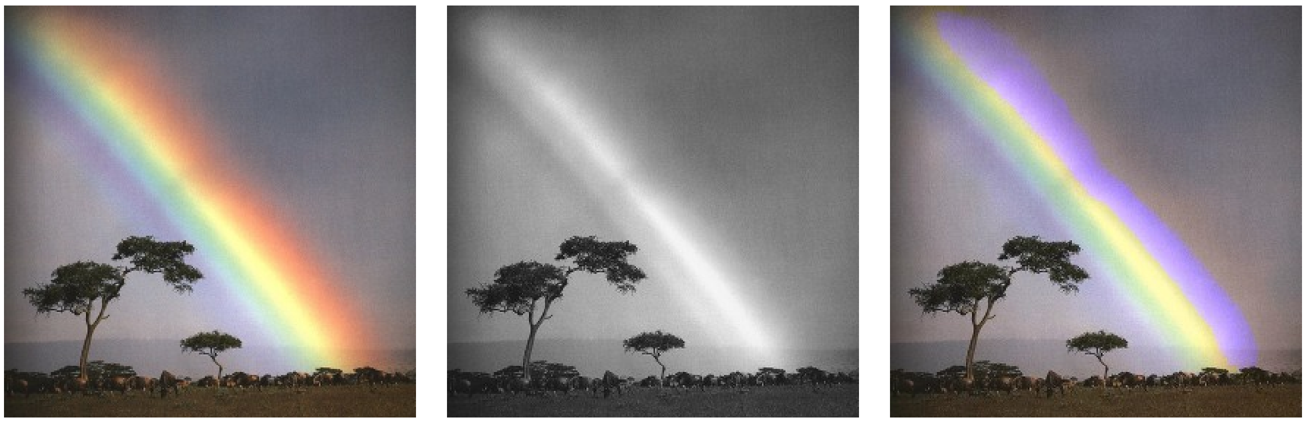
(a) (b)





(c) (d)

Fig. 6. Recolorization example. (a) Original color image, (b) shows the image after recolorization. Note that the marble wall, the sunglasses and the trouser, all changed color. The recolorization process, similarly to the colorization one, needs user scribbles as input. In recolorization there are two kinds of scribbles, those that mark where the original chrominance should be kept as shown in (c), and those of new chrominance (d). The scribbles in (c) are actually a binary mask and can also cover large areas. The scribbles in (d) may have different chrominance exactly in the same way as in colorization. The two kinds of scribbles where placed on separate images just for illustration purposes. It is more convenient to mark all scribbles on a single image since the two kinds do not overlap.



(a) (b) (c)

Fig. 7. Recolorization example using the *Cr* channel for measurement (M) rather than the intensity channel. (a) is the original color image, (b) is the 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 a quite unique example of this effect). On the other hand, both the *Cb* and *Cr* do change significantly between the stripes and therefore can be used for recolorization. (c) Recolored image where the red stripe of the rainbow was replaced by a purple one.

689-694.

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

[14] P. Marshall, “Any Colour You Like,” about.com

[15] E. Reinhard, M. Ashikhmin, B. Gooch, and Peter Shirley, “Color transfer between images,” *IEEE Computer Graphics and Applications* **21:5**, 2001, pp. 34-41.

[16] G. Sapiro, “Inpainting the colors,” *IMA Preprint Series* **1979**, Institute for Mathematics and it Applications, University of Minnesota, May 2004 (www.ima.umn.edu).

[17] J. Sethian, “Fast marching level set methods for three-dimensional photolithography development,” *Proc. SPIE International Symposium on Microlithography*, Santa Clara, California, March, 1996.

[18] J. A. Sethian, “A fast marching level-set method for monotonically advancing fronts,” *Proc. Nat. Acad. Sci.* **93:4**, pp. 1591-1595, 1996.

[19] D. Sýkora, J. Buriánek, and J. Zára, “Unsupervised colorization of black-and-white cartoons,” *Proc. of the 3rd International Symposium on NPAR’04*, pp. 121-127, Annecy, France, June 2004.

[20] J. N. Tsitsiklis, “Efficient algorithms for globally optimal trajectories,” *IEEE Transactions on Automatic Control* **40** pp. 1528-1538, 1995.

[21] T. Welsh, M. Ashikhmin, and K. Mueller, “Transferring color to greyscale images,” *ACM SIGGRAPH 2002 Conference Proceedings*, 2002, pp. 277-280.

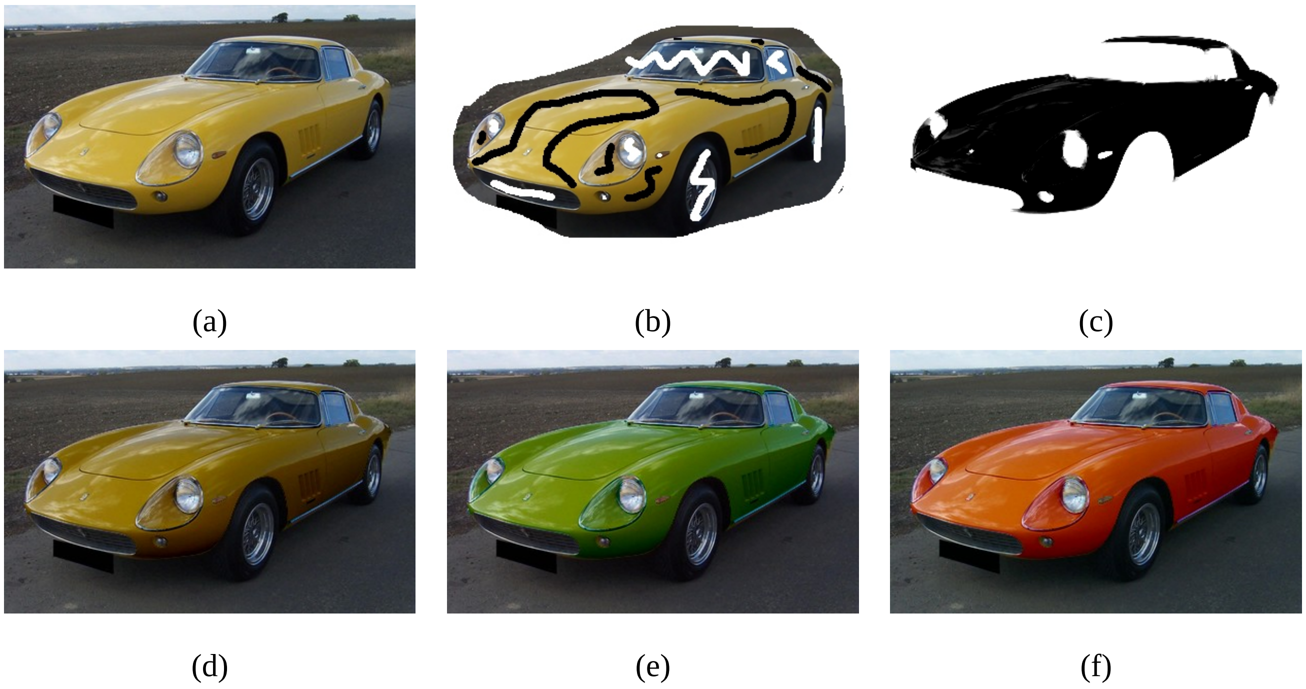


Fig. 8. Example of the generalization of our method to other image processing effects. (a) The original color image. Our goal is to change the color of the yellow car into a darker color. (b) We define the blending medium by roughly marking areas we do not wish to change in white and areas we do want 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 gray-scale 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 grey-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 grey-level matte from the intensity channel, where white means no change and black is the maximum preselected change. It is possible to further process the image using the same matte, (e) and (f) demonstrate this by similarly adding the grey-level matte to the *Cb* and *Cr* channels respectively.