Colorization using Optimization

Anat Levin Dani Lischinski Yair Weiss

School of Computer Science and Engineering
The Hebrew University of Jerusalem*

Figure 1: Given a grayscale image marked with some color scribbles by the user (left), our algorithm produces a colorized image (middle). For reference, the original color image is shown on the right.

Abstract

Colorization is a computer-assisted process of adding color to a monochrome image or movie. The process typically involves segmenting images into regions and tracking these regions across image sequences. Neither of these tasks can be performed reliably in practice; consequently, colorization requires considerable user intervention and remains a tedious, time-consuming, and expensive task.

In this paper we present a simple colorization method that requires neither precise image segmentation, nor accurate region tracking. Our method is based on a simple premise: neighboring pixels in space-time that have similar intensities should have similar colors. We formalize this premise using a quadratic cost function and obtain an optimization problem that can be solved efficiently using standard techniques. In our approach an artist only needs to annotate the image with a few color scribbles, and the indicated colors are automatically propagated in both space and time to produce a fully colorized image or sequence. We demonstrate that high quality colorizations of stills and movie clips may be obtained from a relatively modest amount of user input.

CR Categories: I.4.9 [Image Processing and Computer Vision]: Applications;

Keywords: colorization, recoloring, segmentation

1 Introduction

Colorization is a term introduced by Wilson Markle in 1970 to describe the computer-assisted process he invented for adding color to black and white movies or TV programs [Burns]. The term is now used generically to describe any technique for adding color to monochrome stills and footage.

Colorization of classic motion pictures has generated much controversy [Cooper 1991], which partially accounts for the fact that not many of these movies have been colorized to date. However, there are still massive amounts of black and white television shows that could be colorized: the artistic controversy is often irrelevant here, while the financial incentives are substantial, as was succinctly pointed out by Earl Glick[1] in 1984: "You couldn't make Wyatt Earp today for $1 million an episode. But for $50,000 a segment, you can turn it into color and have a brand new series with no residuals to pay" [Burns]. Colorization of still images also appears to be a topic of considerable interest among users of image editing software, as evidenced by multiple colorization tutorials on the World Wide Web.

A major difficulty with colorization, however, lies in the fact that it is an expensive and time-consuming process. For example, in order to colorize a still image an artist typically begins by segmenting the image into regions, and then proceeds to assign a color to each region. Unfortunately, automatic segmentation algorithms often fail to correctly identify fuzzy or complex region boundaries, such as the boundary between a subject's hair and her face. Thus, the artist is often left with the task of manually delineating complicated boundaries between regions. Colorization of movies requires, in addition, tracking regions across the frames of a shot. Existing tracking algorithms typically fail to robustly track non-rigid regions, again requiring massive user intervention in the process.

In this paper we describe a new interactive colorization technique that requires neither precise manual segmentation, nor accurate tracking. The technique is based on a unified framework applicable to both still images and image sequences. The user indicates how each region should be colored by scribbling the desired color in the interior of the region, instead of tracing out its precise boundary. Using these user supplied constraints our technique automatically propagates colors to the remaining pixels in the image sequence. This colorization process is demonstrated in Figure 1. The underlying algorithm is based on the simple premise that nearby pixels

*e-mail: {alevin,danix,yweiss}@cs.huji.ac.il

[1] Chairman, Hal Roach Studios.

in space-time that have similar gray levels should also have similar colors. This assumption leads to an optimization problem that can be solved efficiently using standard techniques.

Our contribution, thus, is a new simple yet surprisingly effective interactive colorization technique that drastically reduces the amount of input required from the user. In addition to colorization of black and white images and movies, our technique is also applicable to selective recoloring, an extremely useful operation in digital photography and in special effects.

1.1 Previous work

In Markle's original colorization process [Markle and Hunt 1987] a color mask is manually painted for at least one reference frame in a shot. Motion detection and tracking is then applied, allowing colors to be automatically assigned to other frames in regions where no motion occurs. Colors in the vicinity of moving edges are assigned using optical flow, which often requires manual fixing by the operator.

Although not much is publicly known about the techniques used in more contemporary colorization systems used in the industry, there are indications [Silberg 1998] that these systems still rely on defining regions and tracking them between the frames of a shot. BlackMagic, a commercial software for colorizing still images [NeuralTek 2003], provides the user with useful brushes and color palettes, but the segmentation task is left entirely to the user.

Welsh *et al.* [2002] describe a semi-automatic technique for colorizing a grayscale image by transferring color from a reference color image. They examine the luminance values in the neighborhood of each pixel in the target image and transfer the color from pixels with matching neighborhoods in the reference image. This technique works well on images where differently colored regions give rise to distinct luminance clusters, or possess distinct textures. In other cases, the user must direct the search for matching pixels by specifying swatches indicating corresponding regions in the two images. While this technique has produced some impressive results, note that the artistic control over the outcome is quite indirect: the artist must find reference images containing the desired colors over regions with similar textures to those that she wishes to colorize. It is also difficult to fine-tune the outcome selectively in problematic areas. In contrast, in our technique the artist chooses the colors directly, and is able to refine the results by scribbling more color where necessary. Also, the technique of Welsh *et al.* does not explicitly enforce spatial continuity of the colors, and in some images it may assign vastly different colors to neighboring pixels that have similar intensities.

2 Algorithm

We work in YUV color space, commonly used in video, where Y is the monochromatic luminance channel, which we will refer to simply as *intensity*, while U and V are the chrominance channels, encoding the color [Jack 2001].

The algorithm is given as input an intensity volume $Y(x,y,t)$ and outputs two color volumes $U(x,y,t)$ and $V(x,y,t)$. To simplify notation we will use boldface letters (e.g. $\mathbf{r},\mathbf{s}$) to denote (x,y,t) triplets. Thus, $Y(\mathbf{r})$ is the intensity of a particular pixel.

As mentioned in the introduction, we wish to impose the constraint that two neighboring pixels $\mathbf{r},\mathbf{s}$ should have similar colors if their intensities are similar. Thus, we wish to minimize the difference between the color $U(\mathbf{r})$ at pixel $\mathbf{r}$ and the weighted average of the colors at neighboring pixels:

$$J(U) = \sum_{\mathbf{r}} \left(U(\mathbf{r}) - \sum_{\mathbf{s}\in N(\mathbf{r})} w_{\mathbf{rs}} U(\mathbf{s}) \right)^2 \quad (1)$$

where $w_{\mathbf{rs}}$ is a weighting function that sums to one, large when $Y(\mathbf{r})$ is similar to $Y(\mathbf{s})$, and small when the two intensities are different. Similar weighting functions are used extensively in image segmentation algorithms (e.g. [Shi and Malik 1997; Weiss 1999]), where they are usually referred to as *affinity functions*.

We have experimented with two weighting functions. The simplest one is commonly used by image segmentation algorithms and is based on the squared difference between the two intensities:

$$w_{\mathbf{rs}} \propto e^{-(Y(\mathbf{r})-Y(\mathbf{s}))^2/2\sigma_{\mathbf{r}}^2} \quad (2)$$

A second weighting function is based on the normalized correlation between the two intensities:

$$w_{\mathbf{rs}} \propto 1 + \frac{1}{\sigma_{\mathbf{r}}^2} \left(Y(\mathbf{r}) - \mu_{\mathbf{r}}\right)\left(Y(\mathbf{s}) - \mu_{\mathbf{r}}\right) \quad (3)$$

where $\mu_{\mathbf{r}}$ and $\sigma_{\mathbf{r}}$ are the mean and variance of the intensities in a window around $\mathbf{r}$.

The correlation affinity can also be derived from assuming a local linear relation between color and intensity [Zomet and Peleg 2002; Torralba and Freeman 2003]. Formally, it assumes that the color at a pixel $U(\mathbf{r})$ is a linear function of the intensity $Y(\mathbf{r})$: $U(\mathbf{r}) = a_i Y(\mathbf{r}) + b_i$ and the linear coefficients a_i, b_i are the same for all pixels in a small neighborhood around $\mathbf{r}$. This assumption can be justified empirically [Zomet and Peleg 2002] and intuitively it means that when the intensity is constant the color should be constant, and when the intensity is an edge the color should also be an edge (although the values on the two sides of the edge can be any two numbers). While this model adds to the system a pair of variables per each image window, a simple elimination of the a_i, b_i variables yields an equation equivalent to equation 1 with a correlation based affinity function.

The notation $\mathbf{r} \in N(\mathbf{s})$ denotes the fact that $\mathbf{r}$ and $\mathbf{s}$ are neighboring pixels. In a single frame, we define two pixels as neighbors if their image locations are nearby. Between two successive frames, we define two pixels as neighbors if their image locations, *after accounting for motion*, are nearby. More formally, let $v_x(x,y), v_y(x,y)$ denote the optical flow calculated at time t. Then the pixel (x_0,y_0,t) is a neighbor of pixel $(x_1,y_1,t+1)$ if:

$$\left\| (x_0 + v_x(x_0), y_0 + v_y(y_0)) - (x_1,y_1) \right\| < T \quad (4)$$

The flow field $v_x(x_0), v_y(y_0)$ is calculated using a standard motion estimation algorithm [Lucas and Kanade 1981]. Note that the optical flow is only used to define the neighborhood of each pixel, not to propagate colors through time.

Now given a set of locations $\mathbf{r}_i$ where the colors are specified by the user $u(\mathbf{r}_i) = u_i, v(\mathbf{r}_i) = v_i$ we minimize $J(U), J(V)$ subject to these constraints. Since the cost functions are quadratic and the constraints are linear, this optimization problem yields a large, sparse system of linear equations, which may be solved using a number of standard methods.

Our algorithm is closely related to algorithms proposed for other tasks in image processing. In image segmentation algorithms based on normalized cuts [Shi and Malik 1997], one attempts to find the second smallest eigenvector of the matrix $D - W$ where W is a $npixels \times npixels$ matrix whose elements are the pairwise affinities between pixels (i.e., the $\mathbf{r},\mathbf{s}$ entry of the matrix is $w_{\mathbf{rs}}$) and D is a diagonal matrix whose diagonal elements are the sum of the affinities (in our case this is always 1). The second smallest eigenvector of any symmetric matrix A is a unit norm vector x that minimizes $x^{\mathrm{T}}Ax$ and is orthogonal to the first eigenvector. By direct inspection, the quadratic form minimized by normalized cuts is exactly our cost function J, that is $x^{\mathrm{T}}(D-W)x = J(x)$. Thus, our algorithm minimizes the same cost function but under different constraints. In image denoising algorithms based on anisotropic diffusion [Perona and Malik 1989; Tang et al. 2001] one often minimizes a function similar to equation 1, but the function is applied to the image intensity as well.

3 Results

The results shown here were all obtained using the correlation based window (equation 3, or equivalently using the local linearity assumption). The mean and variance μ, σ for each pixel were calculated by giving more weight to pixels with similar intensities. Visually similar results were also obtained with the Gaussian window (equation 2). For still images we used Matlab's built in least squares solver for sparse linear systems, and for the movie sequences we used a multigrid solver [Press et al. 1992]. Using the multigrid solver, the run time was approximately 15 seconds per frame. The threshold T in equation 4 was set to 1 so that the window used was $3 \times 3 \times 3$.

Figure 2 shows some still grayscale images marked by the user's color scribbles next to the corresponding colorization results. Since automating the *choice* of colors was not our goal in this work, we used the original color channels of each image when picking the colors. As can be seen, very convincing results are generated by our algorithm even from a relatively small number of color scribbles.

Typically, the artist may want to start with a small number of color scribbles, and then fine-tune the colorization results by adding more scribbles. Figure 3 demonstrates such a progression on a still image.

Figure 4 shows how our technique can be applied to recoloring. To change the color of an orange in the top left image to green, the artist first defines a rough mask around it and then scribbles inside the orange using the desired color. Our technique is then used to propagate the green color until an intensity boundary is found. Specifically, we minimize the cost (equation 1) under two groups of constraints. First, for pixels covered by the user's scribbles, the final color should be the color of the scribble. Second, for pixels outside the mask, the color should be the same as the original color. All other colors are automatically determined by the optimization process. In this application the affinity between pixels is based not only on similarity of their intensities, but also on the similarity of their colors in the original image. Note that unlike global colormap manipulations, our algorithm does not recolor the other orange in the image, since colors are not propagated across intensity boundaries. The bottom row of the figure shows another example.

Figures 5 and 6 show selected frames from colorized movie clips. Even though the total number of color scribbles is quite modest, the resulting colorization is surprisingly convincing. We have also successfully colorized several short clips from the television show "I Love Lucy" and from Chaplin's classic movie *Modern Times*. The original clips were obviously in black and white, so in these examples we did not have a color reference to pick the colors from.

Figures 7 and 8 compare our method to two alternative methods. In figure 7 the alternative method is one where the image is first segmented automatically and then the scribbled colors are used to "flood fill" each segment. Figure 7a shows the result of automatic segmentation computed using a version of the normalized cuts algorithm [Shi and Malik 1997]. Segmentation is a very difficult problem and even state-of-the-art methods may fail to automatically delineate all the correct boundaries, such as the intricate boundary between the hair and the forehead, or the low contrast boundary between the lips and the face. Consequently, the colorization achieved with this alternative method (figure 7b) is noticeably worse than the one computed by our method (figure 7c). In both cases, the same color scribbles were used. Distinctive colors were deliberately chosen so that flaws in the colorization would be more apparent.

Figure 8 compares our method for colorizing image sequences to an alternative method where a single frame is colorized and then optical flow tracking is used to propagate the colors across time. Since our method uses optical flow only to define the local neighborhood, it is much more robust to tracking failures.

In both cases, either using automatic segmentation or using tracking to propagate colors across time, the results could be improved using more sophisticated algorithms. In other words, if the automatic segmentation had been perfect then flood filling segments would have produced perfect results. Likewise, if dense optical flow had been perfect then propagating colors from a single frame would have also worked perfectly. Yet despite many years of research in computer vision, state-of-the-art algorithms still do not work perfectly in an automatic fashion. An advantage of our optimization framework is that we use segmentation cues and optical flow as "hints" for the correct colorization but the colorization can be quite good even when these hints are wrong.

4 Summary

Despite considerable progress in image processing since 1970, colorization remains a manually intensive and time consuming process. In this paper, we have suggested a method that helps graphic artists colorize films with less manual effort. In our framework, the artist does not need to explicitly delineate the exact boundaries of objects. Instead, the artist colors a small number of pixels in selected frames and the algorithm propagates these colors in a manner that respects intensity boundaries. We have shown that excellent colorizations can be obtained with a surprisingly small amount of user effort.

An attractive feature of phrasing colorization as an optimization problem is that it clarifies the relationship between this problem and other problems in image processing. Specifically, we have shown that our algorithm minimizes the same cost function that is minimized in state of the art segmentation algorithms but under different constraints. In future work, we will build on this equivalence and import advances in image segmentation (e.g. more sophisticated affinity functions, faster optimization techniques) into the problem of colorization. Additionally, we plan to explore alternative color spaces and propagation schemes that treat hue and saturation differently. We are optimistic that these additional improvements will enable us to perform convincing colorizations with an even smaller number of marked pixels.

Acknowledgments

We would like to thank Raanan Fattal for his help with our multigrid solver. This work was supported in part by the Israel Science Foundation founded by the Israel Academy of Sciences and Humanities and by the Israeli Ministry of Science and Technology.

References

BURNS, G. Colorization. Museum of Broadcast Communications: Encyclopedia of Television, http://www.museum.tv/archives/etv/index.html.

COOPER, R. 1991. Colorization and moral rights: Should the United States adopt unified protection for artists? *Journalism Quarterly* (Urbana, Illinois), Autumn.

JACK, K. 2001. *Video Demystified*, 3rd edition ed. Elsevier Science & Technology.

LUCAS, B., AND KANADE, T. 1981. An iterative image registration technique with an application to stereo vision. In *Proc. Int. Joint Conf. AI*, 674–679.

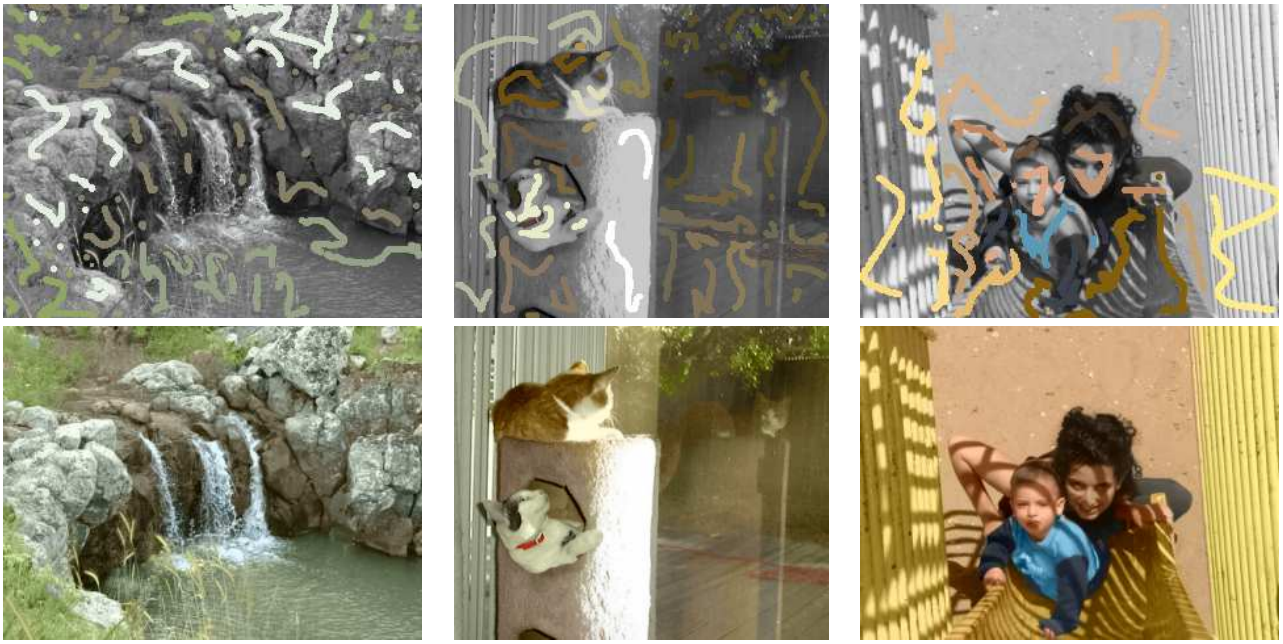


Figure 2: Still image colorization examples. Top row: the input black-white image with scribbled colors. Bottom row: resulting color image.

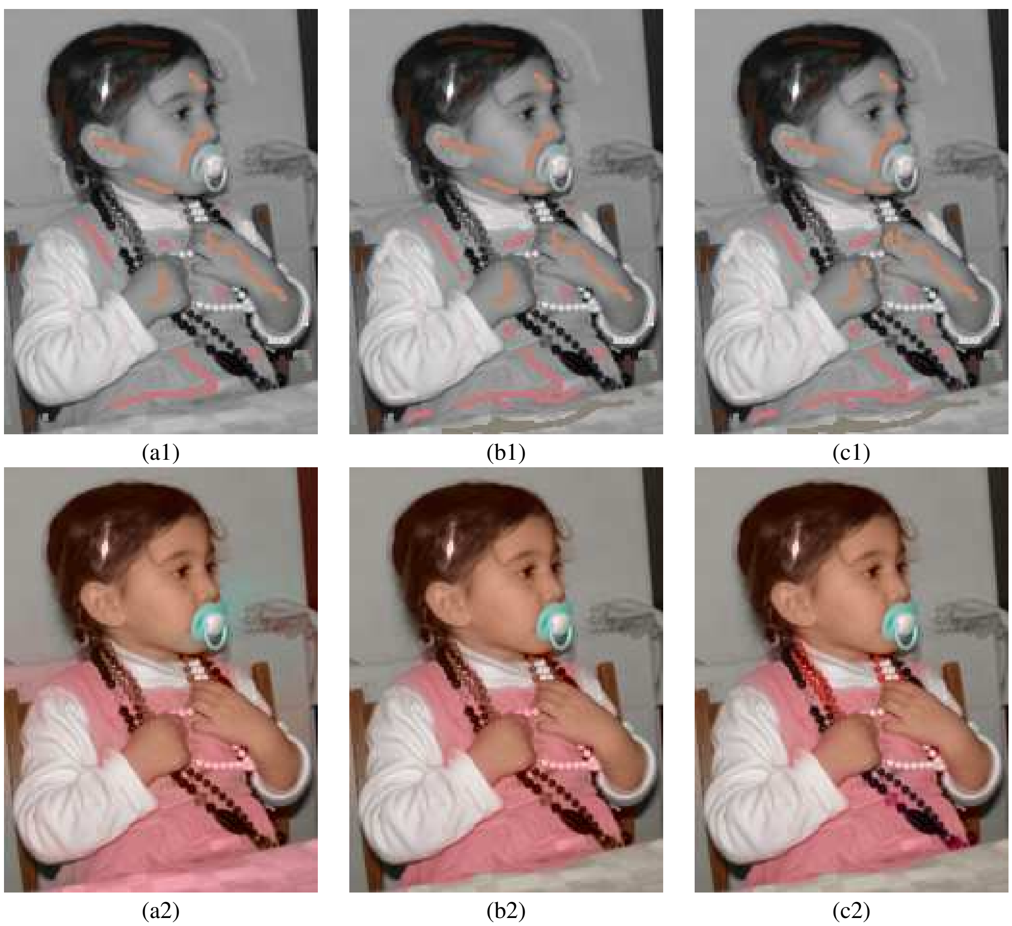
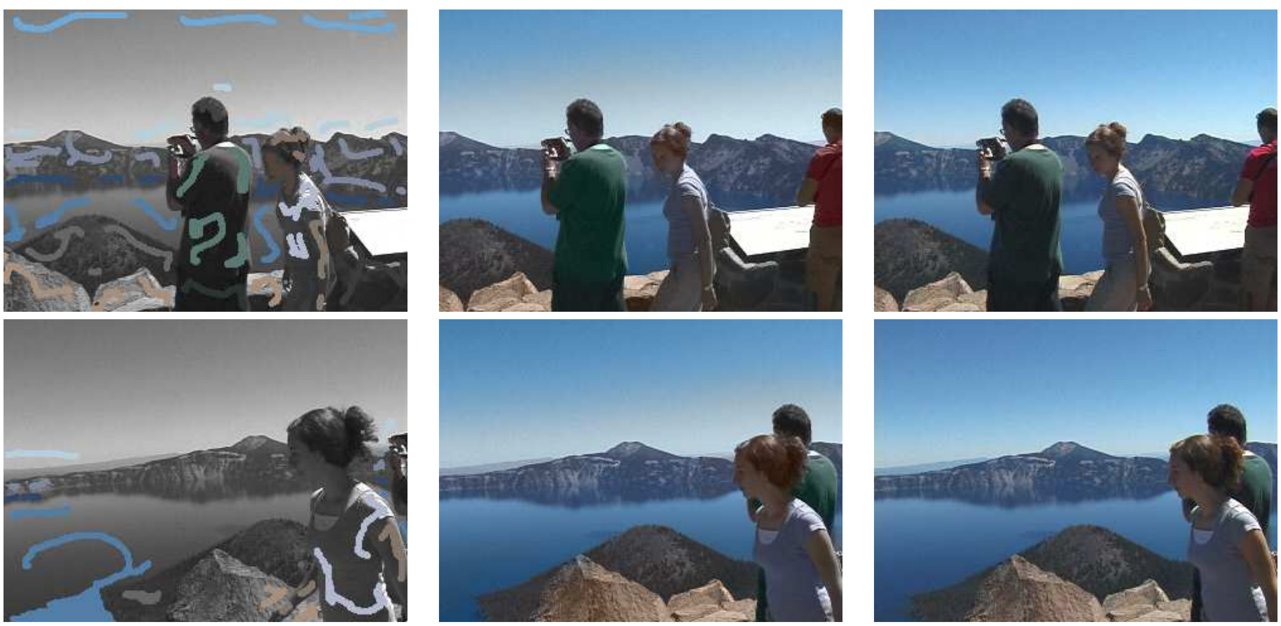


Figure 3: Progressively improving a colorization. The artist begins with the scribbles shown in (a1), which yield the result in (a2). Note that the table cloth gets the same pink color as the girl's dress. Also, some color is bleeding from the cyan pacifier onto the wall behind it. By adding color scribbles on the table cloth and on the wall (b1) these problems are eliminated (b2). Next, the artist decides to change the color of the beads by sprinkling a few red pixels (c1), yielding the final result (c2). Note that it was not necessary to mark each and every bead.

(a) (b) (c)

Figure 4: Recoloring of still images. (a) the input image; (b) pixels marked in white are constrained to keep their original colors; (c) resulting image.



(a) (b) (c)

Figure 5: Video clip colorization example. This 83-frame clip was colorized using 7 marked frames. (a) two of the marked frames; (b) two colorized frames; (c) for comparison, the corresponding frames from the original clip, before color was removed. The input clip, the full set of marked frames, and the resulting colorized clip are available on the ACM SIGGRAPH 2004 Full Conference DVD-ROM.

MARKLE, W., AND HUNT, B., 1987. Coloring a black and white signal using motion detection. Canadian patent no. 1291260, Dec.

NEURALTEK, 2003. BlackMagic photo colorization software, version 2.8. http://www.timebrush.com/blackmagic.

PERONA, P., AND MALIK, J. 1989. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on PAMI 8*, 5, 565–593.

PRESS, W., TEUKOLSKY, S., VETTERLING, W., AND FLANNERY, B. 1992. *Numerical Recipes in C: The art of scientific computing*. Cambridge University Press.

SHI, J., AND MALIK, J. 1997. Normalized cuts and image segmentation. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 731–737.

SILBERG, J., 1998. The Pleasantville post production team that focussed on the absence of color. Cinesite Press Article, http://www.cinesite.com/core/press/articles/1998/10_00_98-team.html.

(a) (b) (c) (d)

Figure 6: Another video colorization example. (a-b) two of the five marked frames; (c-d) two of the 43 colorized frames.

(a) (b) (c)

Figure 7: A comparison with automatic segmentation. For visualization purposes disctinctive colors were used. (a) Segmented image. (b) Result of coloring each segment with a constant color. Segmenting fuzzy hair boundary is a difficult task for typical segmentation methods. (c) Our result.

(a) (b)

Figure 8: A comparison with traditional flow-based tracking. Dense optical flow was computed on the grayscale images, and the color channels from the first frame (colorized using our technique for stills) were warped accordingly. In this experiment, only the frame shown in Figure 6a was marked. (a) Frame 13 from the colored clip, using our technique. (b) Flow-based colorization of the same frame: note the color bleeding on the floor (around the left foot and between the right arm and the rug). While our technique also experiences some difficulties when occluded areas appear, flow-based colorization diverges much earlier.

TANG, B., SAPIRO, G., AND CASSELES, V. 2001. Color image enhancement via chromaticity diffusion. *IEEE Transactions on Image Processing 10*, 5, 701–708.

TORRALBA, A., AND FREEMAN, W. T. 2003. Properties and applications of shape recipes. In *IEEE Computer Vision and Pattern Recognition (CVPR)*.

WEISS, Y. 1999. Segmentation using eigenvectors: A unifying view. In *Proceedings ICCV*, 975–982.

WELSH, T., ASHIKHMIN, M., AND MUELLER, K. 2002. Transferring color to greyscale images. *ACM Transactions on Graphics 21*, 3 (July), 277–280.

ZOMET, A., AND PELEG, S. 2002. Multi-sensor super resolution. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*.