# Manifold Preserving Edit Propagation

Xiaowu Chen* Dongqing Zou* Qinping Zhao
State Key Laboratory of Virtual Reality Technology & Systems, Beihang University

Ping Tan
National University of Singapore

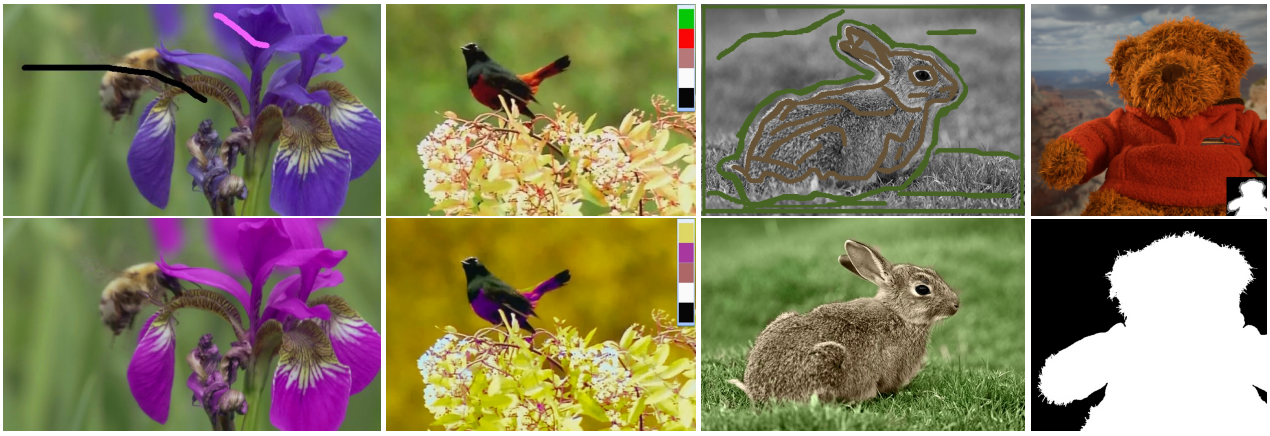

**Figure 1:** *Our edit propagation algorithm can be applied for various applications including color manipulation and matting. From left to right, these are the results of video recoloring, video color theme editing, image colorization and matting. Shown on the top are the input images (with a few user edits). Below are their corresponding output.*

## Abstract

We propose a novel edit propagation algorithm for interactive image and video manipulations. Our approach uses the locally linear embedding (LLE) to represent each pixel as a linear combination of its neighbors in a feature space. While previous methods require similar pixels to have similar results, we seek to maintain the manifold structure formed by all pixels in the feature space. Specifically, we require each pixel to be the same linear combination of its neighbors in the result. Compared with previous methods, our proposed algorithm is more robust to color blending in the input data. Furthermore, since every pixel is only related to a few nearest neighbors, our algorithm easily achieves good runtime efficiency. We demonstrate our manifold preserving edit propagation on various applications.

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

**Keywords:** Manifold Preserving, Edit Propagation, Recoloring, Colorization, Matting

**Links:** DL PDF

*corresponding authors: chen@buaa.edu.cn, zoudq@vrlab.buaa.edu.cn

## 1 Introduction

Efficiently propagating sparse user edits to a whole image (or video) is an important problem with many applications such as interactive color and tonal editing, colorization, matting or defogging, etc. Typically, this propagation favors nearby similar pixels to be edited in a similar way. It can be achieved by either global optimization [Levin et al. 2004; Lischinski et al. 2006; Pellacini and Lawrence 2007; An and Pellacini 2008; Xu et al. 2009] or smooth function interpolation [Li et al. 2010]. To evaluate the affinity between pixels, previous methods often rely on the Euclidean distance [An and Pellacini 2008; Li et al. 2010] or diffusion distance [Farbman et al. 2010] in some feature space.

We perform edit propagation from a different perspective. Instead of evaluating affinity for far apart pixel pairs, we seek to maintain the manifold structure formed by all pixels in a feature space. Specifically, we follow the Locally Linear Embedding (LLE) [Roweis and Saul 2000] [de Ridder and Duin 2002] to represent each pixel as a linear combination of a few of its nearest neighbors in a feature space. We seek to maintain this linear relationship between every pixel and its neighbors during edit propagation. Essentially, our method tries to preserve the manifold structure formed by all pixels as a whole, instead of looking into individual pixel-pairs.

Real images and videos contain objects with different colors. Pixels at object boundaries have blended color of neighboring objects. Semi-transparency, motion and unfocused blur also cause pixels with color blending. These pixels are dissimilar from those on main image objects, where the user tends to put edit scribbles. As a result, pixels with color blending are often less constrained according to the affinity-based edit propagation, which causes distracting artifacts in results. An example is shown in Figure 2, where the pixel $A$ lies on an image edge and has blended color of the flower and the background. From the 3D color space, we can clearly see that the pixel $A$ is dissimilar to the pixels covered by the user drawn scribbles. Hence, when affinity-based edit propagation methods such as [Xu et al. 2009] are applied to recolor the image, the pixel $A$ will tend to keep its original red hue, which causes a distracting 'halo'

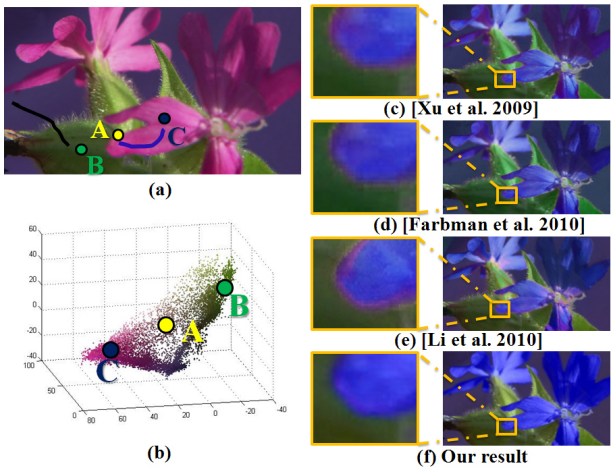


**Figure 2:** *(a) A video frame with some user specified recoloring strokes. (Black means the color should stay unchanged.) (b) The distribution of pixels in the LAB color space. (c), (d), (e) and (f) are the results obtained according to [Xu et al. 2009], [Farbman et al. 2010], [Li et al. 2010] and our method respectively. The authors of these three papers kindly share their code with us. See text for more explanation.*

artifacts as shown in Figure 2 (c). Farbman et al. [2010] used the *diffusion distance* to measure affinity, which has the potential to correctly relate blended pixels with faraway points in the feature space. However, the week affinity between $A$ and the marked pixels leads to similar 'halo' artifacts as shown in Figure 2 (d). Li et al. [2010] solved the edit propagation by function interpolation with a similar affinity measure as [Xu et al. 2009]. As shown in Figure 2 (e), the result has similar artifacts as that in (c). In comparison, our method maintains the manifold structure in the feature space. For example, if the color at $A$ is a linear combination of that at $B$ and $C$ in the input image, our algorithm enforces this linear relationship in edit propagation, and requires $A$ to be the same combination of $B$ and $C$ in the result image. Our algorithm generates more pleasing results as shown in Figure 2 (f).

We apply our edit propagation framework for various image and video editing applications including color manipulations and matting as shown in Figure 1. Since we employ LLE to represent the manifold structure, our method only relates every pixel to a few of its nearest neighbors. In other words, we do not need to compute an all pixel-pairs affinity matrix like most previous affinity-based propagation methods [An and Pellacini 2008; Xu et al. 2009; Farbman et al. 2010]. As a result, our method has advantage on runtime efficiency, and can be easily applied to high resolution images and videos.

## 2 Related Work

Early methods propagate user specified edits locally with the guidance of image gradients. This framework was first introduced by Levin et al. [2004] for colorization, and later extended by Lischinski et al. [2006] for image tonal manipulations. However,methods based on local propagation have difficulties in dealing with fragmented image regions. These methods often require a large number of user scribbles to achieve desired results. In comparison, our method maintains the global manifold structure and generates better results with a few scribbles as demonstrated in experiments. Li et al. [2008] introduced pixel clustering to address this problem. An and Pellacini [2008] advocated for interaction among all pixel-pairs to enable propagation across disconnected regions. However, the exhaustive computation of all-pairs affinity is inefficient in computation and storage. Late, Xu et al. [2009] applied clustering and computed affinity among cluster centers to accelerate computation. Farbman et al. [2010] further employed the diffusion distance [Coifman and Lafon 2006] to faithfully measure affinity between pixels in a feature space. These methods in principal require a dense affinity matrix for edit propagation, which is inefficient. Unlike these optimization based edit propagation methods, Li et al. [2010] formulated the edit propagation as a Radial Basis Functions(RBF) interpolation problem for better computation and memory efficiency. However, all these methods cannot well constrain pixels dissimilar from the marked ones, which often causes halo artifacts at regions with color blending. In comparison, our method is free from such artifacts, though it is slower than [Li et al. 2010].

Our work is also related to a large body of edge preserving filters such as bilateral filter [Tomasi and Manduchi 1998; Paris and Durand 2009]. Here, we only briefly refer to some of the most recent methods. Chen et al. [2007] introduced the *bilateral grid* to speedup and generalize the bilateral filter. Fattal [2009] introduced the edge avoiding wavelets, and applied it for edge-aware editing. Fattal et al. [2009] further derived a coarse image representation to facilitate editing. Ma and Xue [2011] and Yang et al. [2011] enhanced edited images by preserving antialiased edges.

## 3 Manifold Preserving Propagation

We propagate user specified edits from some sparsely marked samples to the whole image or video. In previous works such as [An and Pellacini 2008; Xu et al. 2009; Farbman et al. 2010], this propagation is typically performed with two considerations. First, the results at marked samples should be close to the user specified input. Second, similar pixels should have similar results. This approach cannot handle dissimilar pixels in the image, such as those with significant color blending as shown in Figure 2. This causes halo artifacts of the edited object. The user might add additional strokes on the object boundary to address this problem. However, it is difficult to specify the desired color, since these strokes often cover blended pixels.

We also require the results at marked samples to be close to the user input. However, instead of requiring similar pixels to have similar results, we seek to maintain the manifold structure formed by the pixels in some feature space, which could be the RGB color space. For example, suppose the color at pixel $A$ can be obtained by linearly combining those at the pixel $B$ and $C$ in the original image as shown in Figure 2 (a). We seek to keep this relationship in the result image to propagate user edits.

Our algorithm is inspired by the Locally Linear Embedding (LLE) [Roweis and Saul 2000], which projects data from a high dimensional space to a low dimensional manifold based on the simple intuition that each sample can be represented by a linear combination of its neighbors. Suppose we use a vector $\mathbf{X}_i$ to represent a pixel $i$ in some feature space. Given a data set $\mathbf{X}_1, ..., \mathbf{X}_N$, for each $\mathbf{X}_i$, we find its $K$ nearest neighbors, namely $\mathbf{X}_{i1}, ..., \mathbf{X}_{ik}$. We compute a set of weights $w_{ij}$ that can best reconstruct $\mathbf{X}_i$ from these $K$ neighbors. Specifically, we compute $w_{ij}$ by minimizing

$$\sum_{i=1}^{N} \left\| \mathbf{X}_i - \sum_{j=1}^{K} w_{ij} \mathbf{X}_{ij} \right\|^2, \qquad (1)$$

subject to the constraint $\sum_{j=1}^{K} w_{ij} = 1$. These coefficients can be computed according to the method described in [Roweis and Saul

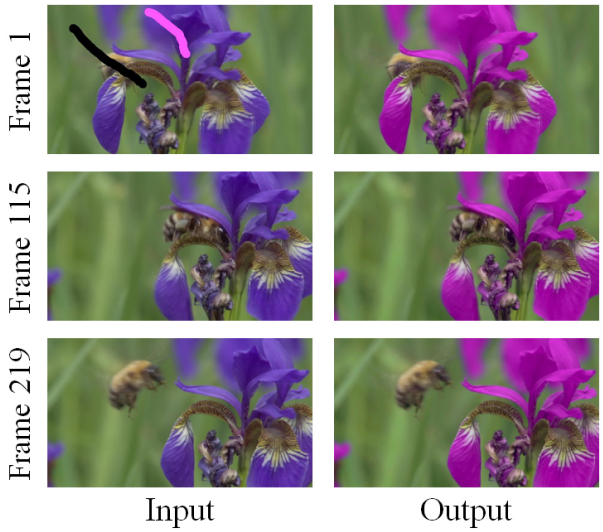


**Figure 3:** *Video object recoloring. Shown on the upper left corner are the user scribbles. Note the user only marks the first frame. We provide some sample frames below. Left and right are the original and recolored videos.*

2000]. [1] The result matrix $W$ (the $ij$-th elements of $W$ is $w_{ij}$) captures the manifold structure of this data set in the feature space. In our edit propagation algorithm, we seek to maintain this manifold structure by requiring $\mathbf{z}_i = \sum_{j=1}^{K} w_{ij}\mathbf{z}_{ij}$ in the result image. Here, $\mathbf{z}_i$ is the edited result at pixel $i$. It can be a pixel color, or some abstract editing parameters such as transparency or tonal values.

Suppose the user specifies the results $\mathbf{g}_i$ for a subset of pixels $\mathcal{S}$. We can propagate this editing to the whole image by inferring a value $\mathbf{z}_i$ at every pixel by minimizing the following energy,

$$E = \lambda \sum_{i \in S} (\mathbf{z}_i - \mathbf{g}_i)^2 + \sum_{i=1}^{N} \left( \mathbf{z}_i - \sum_{z_j \in N_i} w_{ij}\mathbf{z}_j \right)^2 . \qquad (2)$$

The first term ensures the final results to be close to the user specified values $\mathbf{g}_i$ on the subset $\mathcal{S}$. The second term maintains the manifold structure in the feature space.

This energy can be further written in a matrix form as

$$E = (Z_r - G)^T \Lambda (Z_r - G) + Z_r^T (I - W)^T (I - W) Z_r. \qquad (3)$$

Here, $Z_r$ is a vector formed by concatenating all $\mathbf{z}_i$, $I$ is the identity matrix. $\Lambda$ is a diagonal matrix, and $G$ is a vector where

$$\Lambda_{ii} = \begin{cases} \lambda & i \in \mathcal{S} \\ 0 & otherwise \end{cases} \qquad G_i = \begin{cases} \mathbf{g}_i & i \in \mathcal{S} \\ 0 & otherwise \end{cases}$$

Equation (3) is a quadratic function about $Z_r$, which can be minimized by solving the linear equation

$$\left[ (I - W)^T (I - W) + \Lambda \right] Z_r = \Lambda G. \qquad (4)$$

Equation (4) is a sparse linear system and can be solved efficiently.

[1] Note that some of the coefficients could be negative. However, our algorithm is generally robust to these negative coefficients.

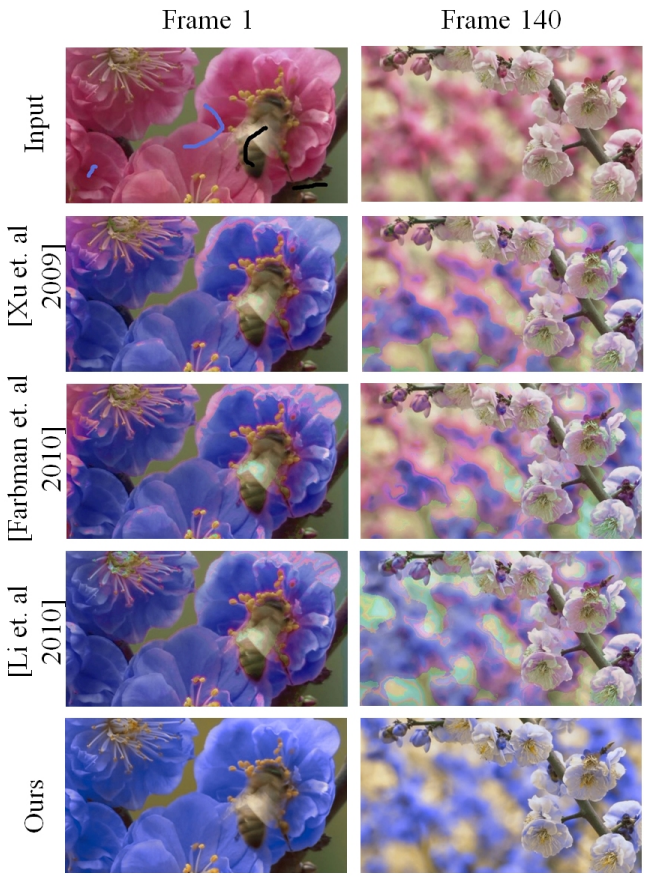


**Figure 4:** *Comparisons on video recolor. The user marks only the first frame (shown at the upper left corner). Our method achieves good results, while the methods of Xu et al. [2009], Farbman et al.[2010] and Li et al. [2010] generate halo artifacts at image boundaries and blending regions.*

## 4 Applications

We apply our manifold preserving edit propagation framework for various image and video editing applications in this section. We also compare them with recent affinity-based propagation such as [Xu et al. 2009; Farbman et al. 2010] to demonstrate the strength of our propagation method. We define a feature vector $\mathbf{X}_i$ as the concatenated RGB color and spatial temporal coordinate $(x, y, t)$ at a pixel $i$ in the recoloring and matting applications. Here, $t$ is the video frame index. In the colorization application, we define the feature vector $\mathbf{X}_i$ as the concatenated texture feature, SIFT descriptor, intensity and spatial coordinate. Of course, additional features such as motion might be used depending on the application.

### 4.1 Video Objects Recoloring

Our manifold preserving propagation can be applied for interactive video objects recoloring. In this application, the user draws color scribbles to specify the desired colors at some sample pixels. (We use black strokes to enforce pixel colors to be unchanged.) The manifold preserving propagation will propagate this edit to the whole video. Here, we set $\mathcal{S}$ as the set of pixels covered by the input scribbles, and $\mathbf{g}_i$ as the user specified color. $\mathbf{z}_i$ is the final RGB color vector optimized at each pixel.

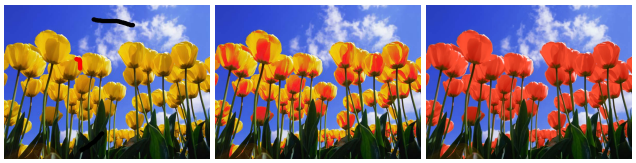

**Figure 5:** *Comparison on image recolor. Left: the input image with user scribbles. Middle: the result from Photoshop CS5 ('Replace Color'). Right: our result.*

### 4.2 Video Color Theme Editing

We also apply our edit propagation to change the color theme of a video. This consists of three steps, namely source color theme extraction, theme mapping and color optimization. The source color theme extraction computes the original color theme $T$ with colors $\{t_1, t_2, ..., t_m\}$ of the input video. Here, we set $m = 11$ and follow the psychophysical study [Chang et al. 2003] to compute the colors. Specifically, we map each RGB color in the original video to a basic color type according to the probabilistic mapping provided in [Chang et al. 2003]. We then compute the mean color for each type to generate $T$. Once $T$ is estimated, the user can select a target color theme $R$ with the colors $\{r_1, r_2, ..., r_m\}$. $R$ can be chosen from the Adobe Kuler, or generated by automatic color enhancement algorithms such as [Wang et al. 2010]. [2] To establish an one-to-one mapping between $T$ and $R$, we minimize $\sum_{i=1}^{m} (t_i - r_{k_i})^2$ by exhaustively checking all possible mappings. Here, $1 \leq k_i \leq m$ and $r_{k_i}$ is the corresponding color of $t_i$. We compute $(t_i - r_{k_i})^2$ as the Euclidean distance in the color mood space [Ou et al. 2004]. We then apply our edit propagation for color theme conversion. We set $\mathcal{S}$ as the pixels whose RGB values exactly match with the colors in $T$. For a pixel $i \in \mathcal{S}$ with original color $t_i$, we set $\mathbf{g}_i = r_{k_i}$. $\mathbf{z}_i$ is the final color RGB of a pixel.

### 4.3 Grayscale Image Colorization

Our method is applicable to colorize grayscale images with a scribble user interface like that in [Levin et al. 2004]. We first apply an over-segmentation method [Achanta et al. 2010] to divide a grayscale image into superpixels. Each superpixel is considered as a point in the feature space. We formulate $\mathcal{S}$ to contain superpixels that intersect with the user strokes. We set $\mathbf{g}_i$ and $\mathbf{z}_i$ as the user specified color and the optimized color at the $i$-th superpixel respectively. We concatenate the texture and SIFT features (both computed in the same way as [Chia et al. 2011]) with average pixel intensity and coordinate in a superpixel as the feature vector. After our edit propagation, all pixels within the same superpixel have the same RGB color. Hence, we smooth this intermediate result by the guided image filter [He et al. 2010].

### 4.4 Image Matting

We can solve image matting with our edit propagation algorithm. In this application, $\mathcal{S}$ is the set of definite foreground and definite background pixels marked by the user. We set $\mathbf{g}_i$ as 1 (or 0) for pixels in the definite foreground (or background) region. We denote the alpha value in the $i$th pixel as $\mathbf{z}_i$. Our manifold preserving propagation will compute the alpha value at unknown pixels.

[2] In case the number of colors in $T$ and $R$ are different, we iteratively merge the two nearest colors in the larger theme to ensure both color themes have the same number of colors.


**Figure 6:** *Video color theme editing. The left and right are sample frames from the original and edited videos. The color themes of both videos are visualized in the bottom.*

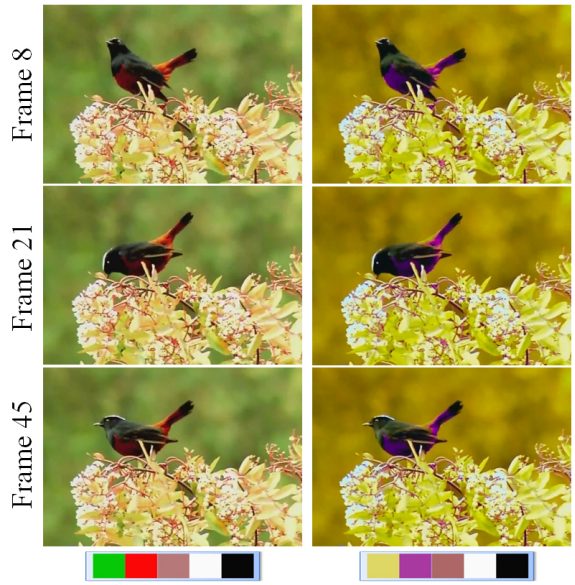


**Figure 7:** *An additional example on video color theme editing.*

## 5 Experiments

We evaluated our methods in the various proposed applications. More results and applications can be found in our supplement files.

### 5.1 Video Objects Recoloring

Figure 3 shows our video object recoloring result. The user only marked a few strokes in the first frame of the video. All the 242 frames were successfully recolored. We show some sample frames in Figure 3, where the left and right images are the original and recolored video frames. More examples are included in the supplementary files.

We compare our method with [Xu et al. 2009], [Farbman et al. 2010] [Li et al. 2010] in Figure 4. The foreground flower was not

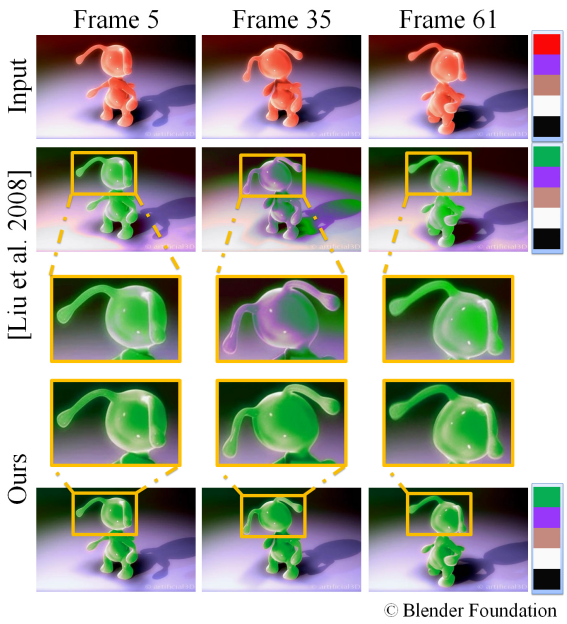


**Figure 8:** *Comparison on color them editing with optical flow [Liu et al. 2008] based propagation. The imprecise flow computation causes color fluctuation, while our method produces stable results.*

recolored appropriated in the results of [Xu et al. 2009], [Farbman et al. 2010] and [Li et al. 2010], where some boundary pixels with color blending kept the original hue. In comparison, our method generated more nature results. We also compare our method with the 'Replace Color' feature in Photoshop CS5 in Figure 5. Our method produced a satisfactory result as shown on the right. In comparison, the 'Replace Color' feature could not propagate over some of the image edges as shown in the middle. Additional scribbles will be required to achieve results with similar quality as ours.

### 5.2 Video Color Theme Editing

Figure 6 shows some results from our video color theme editing method. Once the user specified the target color theme, all the 466 frames were automatically processed by our method in 21 seconds. Our method generated visually appealing results (please refer to the supplementary video). Figure 7 provides an additional example. User edits could also be propagated along optical flow in video sequences. However, optical flow algorithms tend to fail in complex videos with variable lighting, fast motion or motion blur. Figure 8 shows a comparison between our method and the optical flow [Liu et al. 2008] based propagation. Due to inaccurate flow computation, the color of the same object fluctuates back and forth as shown in the second row of Figure 8. In contrast, our method generated consistent results.

### 5.3 Grayscale Image Colorization

Figure 9 shows the comparison with [Levin et al. 2004] on colorization. The method described in [Levin et al. 2004] requires nearby pixels with similar intensity to be colored similarly. Therefore, pixels faraway from the user strokes may be colored incorrectly, as can be seen from the 'bunny' example in Figure 9 (we highlight some problems with red boxes in the picture). At the same time, it also has difficulties in dealing with pixels in blended regions (such as hair or unfocused regions), because these pixels are often less similar from marked ones. This problem is exemplified by the 'bear' and 'flower' examples in Figure 9. Generally speaking, to achieve results of similar quality, our method requires less user scribbles.

### 5.4 Image Matting

We compare our results on matting with [Levin et al. 2008] in Figure 10 on the dataset provided by [Rhemann et al. 2009]. For each example, we first show the input image, tri-map and ground truth matting on the top from left to right. The results obtained from [Levin et al. 2008] and our method are provided below. Our method works well in most of the examples for its advantages in dealing with color blending which is common in translucent objects. The example in the lower right corner shows the limitation of our method in dealing with large translucent objects. More matting results are provided in the supplementary materials.

## 6 Evaluation

We evaluated our algorithm on a variety of images and videos on a PC with an Intel Core 4 Duo 2.6GHz processor and 4GB RAM. Typically, for 30M pixels and $K = 30$, our method takes about 200 seconds and 350M RAM.

We evaluate our method in Figure 11 with different number of neighbors (varying $K$). Typically, $K$ should be large enough to construct and maintain the manifold structure. When $K$ is too smaller, as in Figure 11(a) ($K = 1$) and (b) ($K = 3$), the propagation will fail because we cannot construct the whole manifold from very limited neighborhood information. In practice, we use the error $\varepsilon = \sum_{i=1}^{N} \left\| \mathbf{z}_i - \sum_{j=1}^{K} w_{ij}\mathbf{z}_{ij} \right\|^2$ to evaluate our choice of $K$. This error is large when $K$ is too small and the manifold cannot be constructed. We set $K = 30$ in all the experiments shown in this work.

**Limitation** One limitation of our method is that the edit propagation does not take semantic information into consideration. The first row of Figure 12 shows such an example, where the skin of the image character is changed to red when we try to match the color theme of the image (b) to that of (a). This is a common drawback in edit propagation methods where semantic information is not included. Our method is also limited when dealing with image regions with indistinct features. As shown in the second row of Figure 12, part of the cat was incorrectly colored in green because there is insufficient feature for the algorithm to differentiate the cat and foliage.

## 7 Conclusion

We presented a novel algorithm based on locally linear embedding for edit propagation. While previous methods propagate edits by evaluating affinities among pixels, our method preserves the overall manifold structure formed by all pixels in a feature space. We demonstrated the effectiveness of our method in various applications including color editing, gray image colorization and matting.

## Acknowledgements

We would like to thank the anonymous reviewers for their help in improving the paper. We also want to thank Song-Chun Zhu for helpful suggestions, Jianwei Li, Feng Ding and Zhihong Fang for data processing, and Michael Brown for advices on English. This work was partially supported by NSFC (60933006), 863 Program (2012AA011504). Ping Tan was supported by the Singapore ASTAR PSF grant R-263-000-698-305.

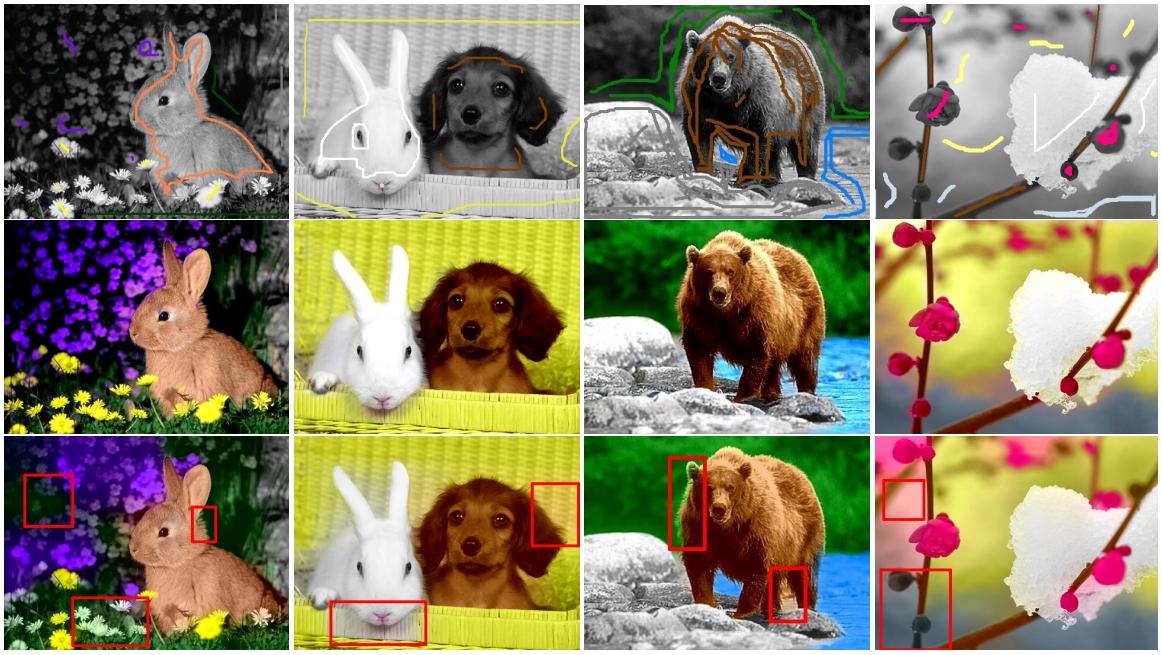

**Figure 9:** *Comparison on colorization. The first row shows the user scribbles. The second and the third rows are the results obtained with our method and the method in [Levin et al. 2004]. We highlight some problems with red boxes.*

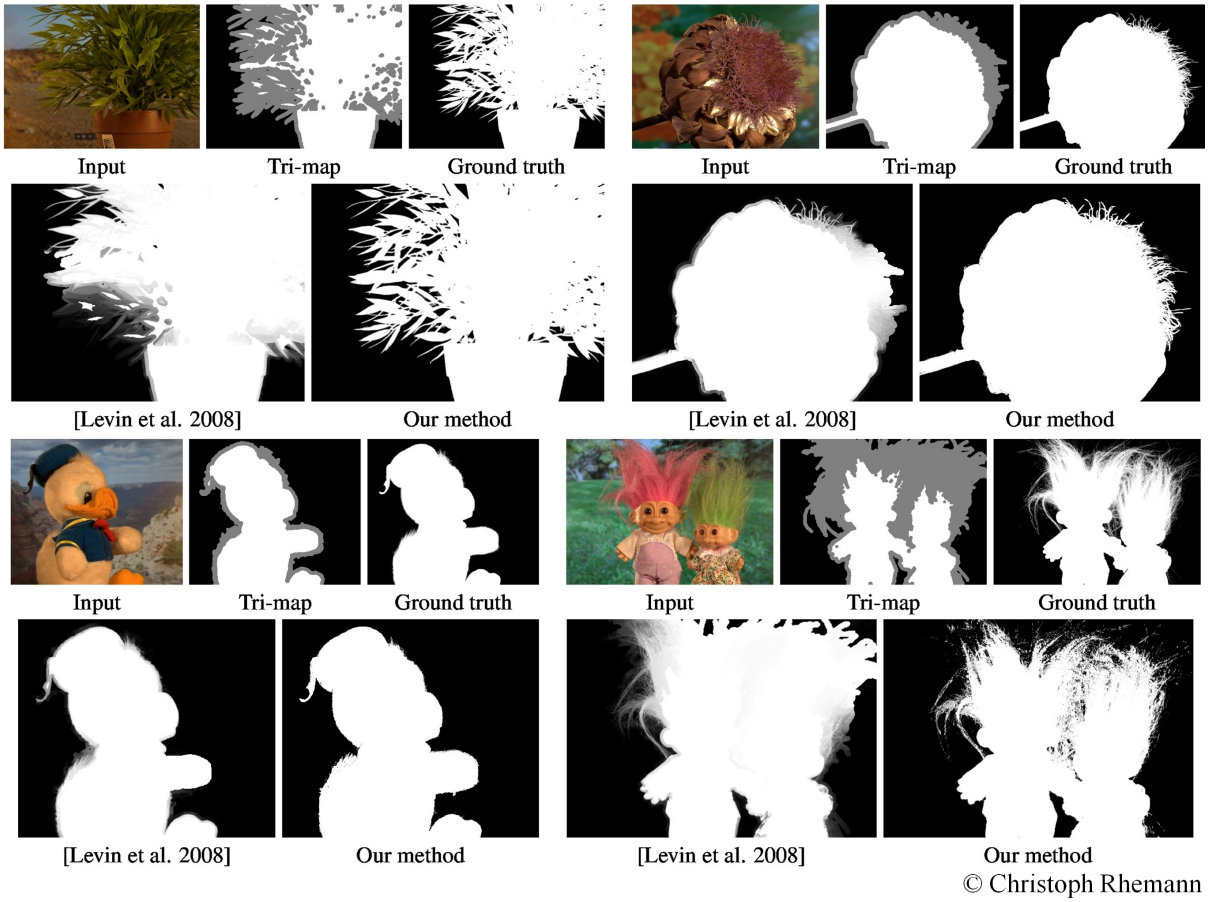


**Figure 10:** *Comparisons on matting. For each example, we show the input image, tri-map and ground truth on the top, and show the results generated by [Levin et al. 2008] and our method on the bottom.*

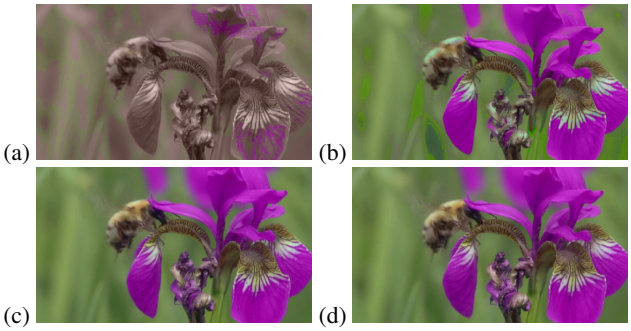


**Figure 11:** *Color theme editing with different K. Please refer to the upper left corner of Figure 1 for the input. (a)-(d) are results obtained by setting K as 1, 3, 20, 50.*

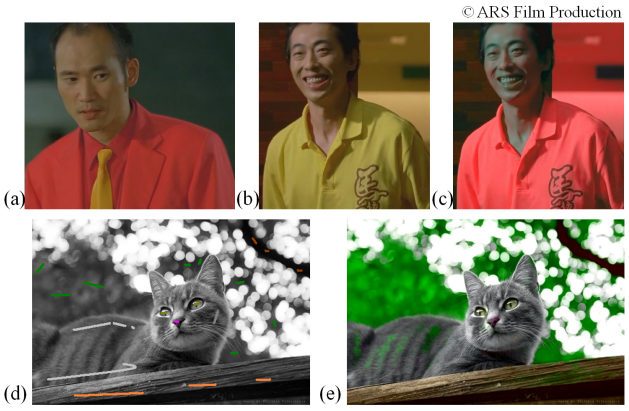


**Figure 12:** *Failed examples of our method. The first row shows a failure case for lacking of semantic constraints. The second row shows problems caused by indistinct features.*

## References


ACHANTA, R., SHAJI, A., SMITH, K., LUCCHI, A., FUA, P., AND SÜSSTRUNK, S. 2010. SLIC Superpixels. Tech. Rep. no. 149300, EPFL.

AN, X., AND PELLACINI, F. 2008. Appprop: all-pairs appearance-space edit propagation. In *ACM Trans. Graph. (Proc. of Siggraph)*, vol. 27.

CHANG, Y., SAITO, S., AND NAKAJIMA, M. 2003. A framework for transfer colors based on the basic color categories. In *Proc. of Computer Graphics International*, 176–183.

CHEN, J., PARIS, S., AND DURAND, F. 2007. Real-time edge-aware image processing with the bilateral grid. *ACM Trans. Graph. (Proc. of Siggraph) 26*, 3.

CHIA, A. Y.-S., ZHUO, S., GUPTA, R. K., TAI, Y.-W., CHO, S.-Y., TAN, P., AND LIN, S. 2011. Semantic colorization with internet images. *ACM Trans. Graph. (Proc. of Siggraph Asia) 30*, 6.

COIFMAN, R., AND LAFON, S. 2006. Diffusion maps. *Appl. Comput. Harmon. A. 21*, 1, 5–30.

DE RIDDER, D., AND DUIN, R. P. 2002. Locally linear embedding for classification. Tech. Rep. PH-2002-01.

FARBMAN, Z., FATTAL, R., AND LISCHINSKI, D. 2010. Diffusion maps for edge-aware image editing. *ACM Trans. Graph. (Proc. of Siggraph Asia) 29*, 6.

FATTAL, R. 2009. Edge-avoiding wavelets and their applications. *ACM Trans. Graph. 28*, 3.

HE, K., SUN, J., AND TANG, X. 2010. Guided image filtering. In *Proc. of ECCV*, 1–14.

LEVIN, A., LISCHINSKI, D., AND WEISS, Y. 2004. Colorization using optimization. *ACM Trans. Graph. (Proc. of Siggraph) 23*, 3.

LEVIN, A., RAV-ACHA, A., AND LISCHINSKI, D. 2008. Spectral matting. *IEEE Trans. Pattern Anal. Mach. Intell. 30*, 10, 1699–1712.

LI, Y., ADELSON, E. H., AND AGARWALA, A. 2008. Scribbleboost: Adding classification to edge-aware interpolation of local image and video adjustments. *Comput. Graph. Forum 27*, 2, 1255–1264.

LI, Y., JU, T., AND HU, S.-M. 2010. Instant propagation of sparse edits on images and videos. *Comput. Graph. Forum 29*, 7, 2049–2054.

LISCHINSKI, D., FARBMAN, Z., UYTTENDAELE, M., AND SZELISKI, R. 2006. Interactive local adjustment of tonal values. *ACM Trans. Graph. (Proc. of Siggraph) 25*, 3.

LIU, C., FREEMAN, W. T., ADELSON, E. H., AND WEISS, Y. 2008. Human-assisted motion annotation. In *Proc. of CVPR*.

MA, L.-Q., AND XU, K. 2011. Antialiasing recovery for edit propagation. In *Proc. of Virtual Reality Continuum and Its Applications in Industry (VRCAI)*, 125–130.

OU, L., LUO, M., WOODCOCK, A., AND WRIGHT, A. 2004. A study of colour emotion and colour preference. part i: Colour emotions for single colours. In *Proc. of Color Res. Appl.*, 232–240.

PARIS, S., AND DURAND, F. 2009. A fast approximation of the bilateral filter using a signal processing approach. *Int. J. Comput. Vision 81*, 1, 24–52.

PELLACINI, F., AND LAWRENCE, J. 2007. Appwand: editing measured materials using appearance-driven optimization. *ACM Trans. Graph. (Proc. of Siggraph) 26*, 3.

RHEMANN, C., ROTHER, C., WANG, J., GELAUTZ, M., KOHLI, P., AND ROTT, P. 2009. A perceptually motivated online benchmark for image matting. In *Proc. of CVPR*, 1826–1833.

ROWEIS, S., AND SAUL, L. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 2323–2326.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. In *Proc. of ICCV*, 839.

WANG, B., YU, Y., WONG, T.-T., CHEN, C., AND XU, Y.-Q. 2010. Data-driven image color theme enhancement. *ACM Trans. Graph. (Proc. Siggraph Asia) 29*, 6 (December), 146:1–146:10.

XU, K., LI, Y., JU, T., HU, S.-M., AND LIU, T.-Q. 2009. Efficient affinity-based edit propagation using k-d tree. *ACM Trans. Graph. (Proc. of Siggraph Asia) 28*, 5.

YANG, L., SANDER, P. V., LAWRENCE, J., AND HOPPE, H. 2011. Antialiasing recovery. *ACM Trans. Graph. 30*, 3.