Video Colorization Using Parallel Optimization in Feature Space

Bin Sheng, *Member, IEEE,* Hanqiu Sun, *Member, IEEE,* Marcus Magnor, *Member, IEEE,* and Ping Li

***Abstract*—We present a new scheme for video colorization using optimization in rotation-aware Gabor feature space. Most current methods of video colorization incur temporal artifacts and prohibitive processing costs, while this approach is designed in a spatiotemporal manner to preserve temporal coherence. The parallel implementation on graphics hardware is also facilitated to achieve realtime performance of color optimization. By adaptively clustering video frames and extending Gabor filtering to optical flow computation, we can achieve real-time color propagation within and between frames. Temporal coherence is further refined through user scribbles in video frames. The experimental results demonstrate that our proposed approach is efficient in producing high-quality colorized videos.**

***Index Terms*—Gabor feature space, parallel optimization, user strokes, video colorization.**

I. Introduction

MODERN SYSTEMS for digital video editing allow nonprofessional multimedia producers to restore or adjust the color of original monochromatic images [1]. Although substantial effort has been devoted to color propagation within a single image, comparatively little research has focused on videos. This is probably because of the increasing complexity of video colorization, which must balance the constraints between spatio-temporal color coherence and processing costs [1], [2]. Optimization over an entire video will be very slow because video data appear in a 3-D space (2-D frames arranged in linear time). The existing video colorization methods [1], [3] make the color propagation from keyframes to subsequent frames easier by relying on optical flow. However, optical flow will cause artifacts in certain frames, which are usually reflected in the colorization results. Such errors cannot be easily eliminated through user input or postprocessing approaches [1] because the optimization used by most of these methods is based on a one-pass linear system solver.

In this paper, a more efficient video colorization optimization method under parallel color propagation is proposed. Our key observation is that, since color propagation assigns similar pixels with similar colors, video colorization constitutes a smooth function in a higher-dimensional pixel feature space rather than in a video space. This function is approximated with spatio-temporal pixel subgraphs in the feature space. Thus, unlike previous methods that solve color optimization in a very large linear system, this method iteratively propagates the colors among the pixels in the graphs, which results in greater refinement and parallel acceleration. It was observed that this approximation maintains the visual fidelity of previous optimization methods while achieving a high level of parallel color propagation.

Special attention should be paid to the temporal coherence when static image colorization methods are applied to moving images. Colors may shift erratically or show no consistency between frames, causing visual fatigue for viewers. To solve these problems, we propose to leverage user scribbles through a new feature space formed by rotation-aware Gabor filtering. In this feature space, color similarity is established and Gabor flow is then used to compute the temporal connectivity of the pixel graphs. By optimizing colors on a per-pixel basis, color mismatch even between disjoint regions with similar textures is minimized. In comparison with previous work on stroke-based colorization, two notable contributions are made in this paper:

1) a method for maintaining temporal color coherence using optimized Gabor flow;
2) a parallelized strategy for fully solving the optimization of video colorization.

This paper is organized as follows. Previous work related to our approach is briefly reviewed in Section II and the Gabor feature space is presented in Section III, including rotation-aware Gabor filtering and feature graph generation. Color propagation over the video space and parallel optimization of this process are described in Section IV, a variety of experimental colorization results are presented in Section V, and concluding remarks are provided in Section VI.

Manuscript received January 27, 2013; revised April 27, 2013 and June 8, 2013; accepted June 19, 2013. Date of publication August 6, 2013; date of current version March 4, 2014. This work was supported in part by the National Natural Science Foundation of China under Grants 61202154 and 61133009, in part by the National Basic Research Project of China under Grant 2011CB302203, in part by Shanghai Pujiang Program under Grant 13PJ1404500, in part by RGC Grant 416311, in part by UGC Direct Grants for Research under Grants 2050454 and 2050485, in part by the European Union's Seventh Framework Program FP7/2007-2013 under Grant Agreement 256941, in part by Reality CG, in part by the Open Projects Program of the National Laboratory of Pattern Recognition, and in part by the Open Project Program of the State Key Laboratory of CAD&CG under Grant A1206, Zhejiang University. This paper was recommended by Associate Editor S. Battiato.

B. Sheng is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: shengbin@cs.sjtu.edu.cn).

H. Sun and P. Li are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong.

M. Magnor is with the Computer Graphics Laboratory, Braunschweig University of Technology, Braunschweig 38106, Germany.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2013.2276702

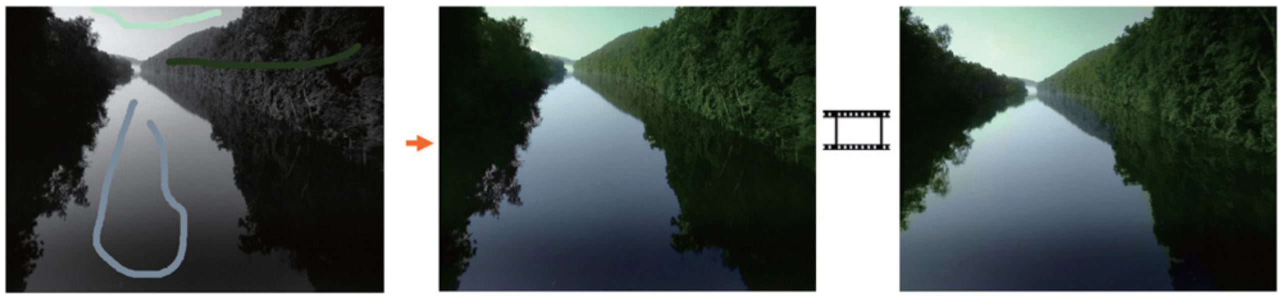


Fig. 1. Colorization results of the *River-Bank* video using parallel optimization in the Gabor feature space.

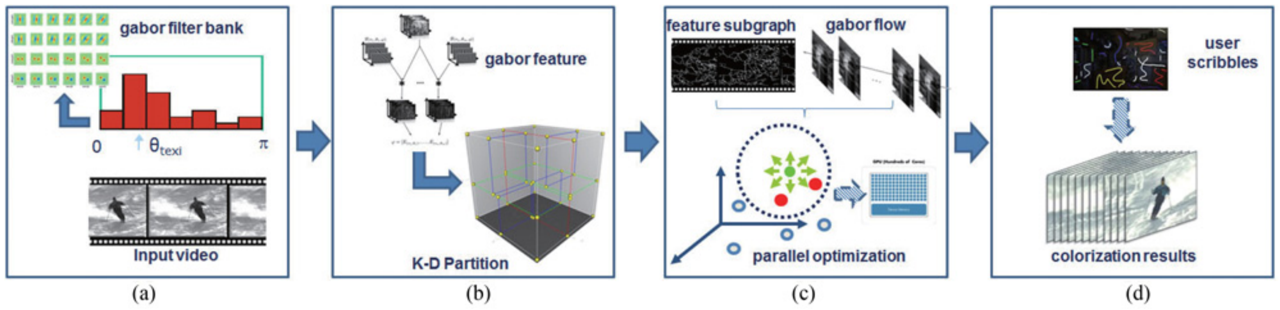


Fig. 2. Pipeline for our video colorization approach. (a) Given a set of source video frames, we set up rotation-aware Gabor filtering for texture discrimination and resolution detection. (b) Feature space generated by Gabor filtering is then adaptively partitioned into K-D tree subgraphs. (c) Gabor flow is constructed to represent temporal correspondence among different subgraphs. (d) We then propagate the colors directly to the pixels of these subgraphs in parallel.

II. Related Work

For image colorization techniques, colorization is usually achieved on the basis of the color scribbles input by users onto target grayscale images. In the technique in [1], the colors of all remaining image pixels are optimized by using these scribbles as constraints, while in other colorization approaches, learning techniques and grayscale intensities are utilized [2], [4], [5]. The relation between a grayscale image and its colored version is learned by the examples in [5] and [6], while a grayscale image is colorized on the basis of single and multiple images in [7] and [8], respectively. Color scribbles drawn by the user over a given image are utilized in other colorization techniques [3], [9]–[11]. The colors of these scribbles are then distributed algorithmically across the whole image. For example, a weighted average of scribble colors is used to colorize pixels [3], where a weight is in proportion to the geodesic distance between a pixel and its scribble. Bayesian texture classification for colorization of grayscale aerial and space imagery, and prototype matching can be used [12] to overcome some of the shortcomings mentioned in [4].

Recently, an image colorization technique based on the Gabor feature space was proposed, but only suitable for the single-image domain [13]. Since the pixels are still in the 2-D image space, color distribution can be optimized across the entire image. But this strategy does not scale well to video data and will induce cross-frame flickering artifacts. The key to handling these problems is color propagation in temporal video space, not merely in the 2-D space of unrelated frame images. A neural network when used to complete this temporal propagation [14] also increases computational costs and hinders parallelization.

Optical flow algorithms [15]–[22] are some of the most efficient methods for calculating the inconsistency of stereo imaging and motion in video sequences. However, these algorithms are faced with severe challenges that are caused by the occlusions resulting from scene structure and/or object motion and will lead to undependable intensity matching and optical flow smoothing across object boundaries (near the occlusion area). In lack of clearly detecting occlusions, some methods are put forward to handle oversmoothing, such as anisotropic diffusion with image or flow adaptiveness [20], [23]–[25], and isotropic diffusion with nonquadratic regularization [19], [21]. These methods can constitute vector fields that preserve discontinuity but not account for occlusions. A further step was taken by calculating both forward and backward optical-flow fields with flow-field regularization [26], [27]. The resulting flow diffusion in [26] is isotropic but adaptive to the difference between forward and backward optical-flow calculations (a large difference indicates occlusion), while the flow diffusion in [27] is anisotropic and dependent on image gradients. In the recent VideoSnapCut [28] high-quality results are obtained by using a series of local classifiers combined with a matting method, while reserving the capability for local refinement. Another recent technique, SIFT [29], is based on image registration through tracking cross-frame robust feature descriptors.

III. Approach Overview

The essential task of video colorization is to assign colors to pixels based on the intuition that pixels with high similarity should have similar colors. The key to our approach is to establish pixel similarity in the video's grayscale channel. To accomplish this, a novel pipeline is introduced that measures pixel similarity across video frames and allows parallel color optimization among pixels. Fig. 2 provides an overview of our approach. First, a rotation-aware feature space is set

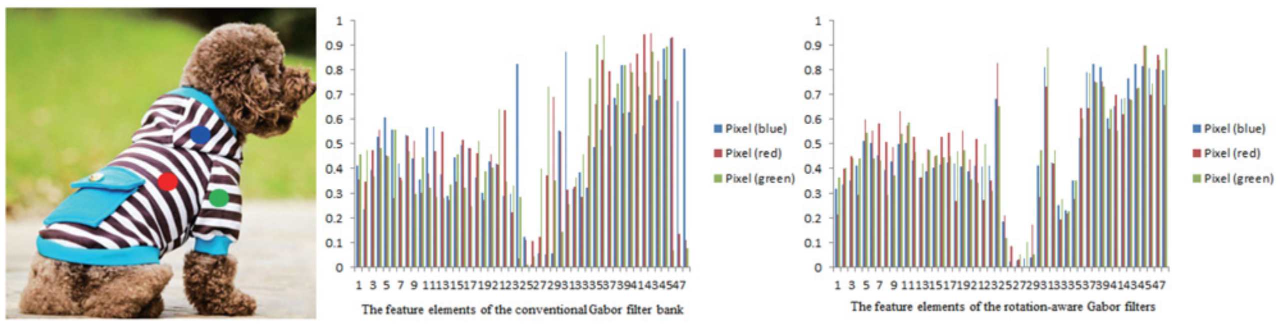


Fig. 3. Rotation-aware Gabor filter responses. Three sample pixels from the filtered video frame on the left. On the right, the 48 outputs of Gabor filter banks are drawn sequentially (the filter banks for the three samples are marked in red, green, and blue, respectively), demonstrating that the rotation-aware Gabor filters track textures more accurately than conventional Gabor filters.

up through fast Gabor filtering (Section III-A). Then, the pixels in this feature space are adaptively clustered to form subgraphs for local color optimization (Section III-B). Gabor flow is also used to establish dense temporal correspondences between subgraphs (Section III-C). Finally, the color optimization problem is approximated through per-pixel linear iteration in the feature space, thereby satisfying the user-specified color indications (Section IV). The general idea is to avoid unnecessary computations by constructing the Gabor flow to represent most pixel correspondences, and then by propagating colors directly to corresponding pixels, without color refinement. Fig. 1 illustrates this use of Gabor flow in color propagation for an example video.

A. Rotation-Aware Gabor Features

Pixel similarity is expressed in the feature space by Gabor filters. The outputted bank of 24 Gabor filters (four scales, six orientations) on a $k \times k$ neighborhood around a pixel is used to produce feature vectors. These filters locally represent the scale and orientation of a texture. The Gabor functions are mathematically characterized in [10] and [30].

The Gabor filters are built according to [30], and the individual feature components are normalized to balance their effects throughout the feature space [10], [30]. This normalization was performed pair-wise together with the calculation of feature distance between two pixels [30]. Fig. 3 shows the normalized effects of three pixels on the feature components in a test image. The mean $\mu_{m,n}$ and SDs$\sigma_{m,n}$ are equivalently normalized to measure texture similarity.

This approach measuring pixel similarity works well sometimes, but quite poorly for rotated textures or objects because all filter banks are orientated in the same way. Therefore, two pixels lying within objects or textures with different orientations may be measured to be very different even they are in the same neighborhood, as it often occurs in natural images. To build feature vectors with invariant rotation, a function $R(x, y) = \theta_{tex_i}$ representing the texture direction of pixel i in position (x, y) is determined, so the feature distance calculation is matched in terms of orientation. Then, a set of rotation-aware Gabor filters for pixel i is generated by adjusting each filter's orientation θ_n according to texture direction θ_i

$$\theta_n = (n\pi + \theta_{tex_i})/R. \tag{1}$$

By substituting (1) into the Gabor filter formulations, the rotation-aware Gabor filters are constructed for generation of feature vectors based on Fourier transformation. Due to the orientation of slope θ in a video frame, energy will be primarily distributed along $\theta \pm \pi/2$ in the complex frequency plane, and based on this, an orientation-sensitive Fourier measurement for pixel i is performed to build the local direction

$$t_{V(x_i,y_i,t_i)}(\theta) = \int |Fourier(V(x_i, y_i, t_i))_{\rho,\theta}|^2 d\rho. \tag{2}$$

$Fourier(V(x_i, y_i, t_i))_{\rho,\theta}$ is the Fourier transformation. $Fourier(V(x_i, y_i, t_i))$ in polar coordinates, on the $k \times k$ sampling window surrounding the video pixel $V(x_i, y_i, t_i)$ so that the direction of pixel $V(x_i, y_i, t_i)$ is defined as the θ when $t_{V(x_i,y_i,t_i)}(\theta)$ is maximized.

B. Feature Space Clustering

It is supposed that in a higher-dimensional space where pixels in a video are represented as points, the points representing pixels with high similarity are located close to each one. This space is called a Gabor feature space. If color is to be propagated throughout the neighborhood in the whole feature space, the connectivity of neighboring pixels should be determined. The concept of neighboring pixels in the feature space relies on the selection of similarity measure. In this paper, the similarity measure given in [13] is adopted. Based on this, we can define the Gabor feature space by simply expressing each pixel with its Gabor feature vector, because for the highly similar pixels, the Euclidean distance between their feature vectors is smaller. However, computation of neighbors in this feature space is rather expensive. In response, the clustering approaches from [31] and [32] were used, hoping to find a more efficient method for neighbor computation in the Gabor feature space.

To build connectivity and neighborhood for the pixels, the feature space is further subdivided by a k-d tree at higher dimensions. The tree was built in a top-down way, which hierarchically reflects the pixel similarity. It is supposed that D is a set of all the pixels of the input video, and cell C_i is related to its k-d tree partition. All that is needed is a predicate κ, with which, the pixel set P_i in Cell C_i can be considered a

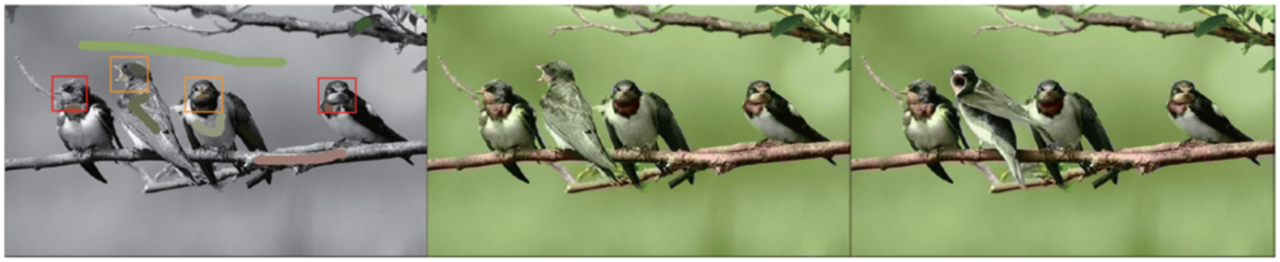


Fig. 4. Our video colorization (from left to right): the input scribbles of frame 1; the colorization result in Gabor space, in which the colors are transferred over the disjoint regions shown in the marked boxes; and the colorization propagation on frame 19.

cluster of pixels that neighbor each other in the feature space. $\kappa(C_i)$ is considered true when

$$|P_i| < \varepsilon_a \vee \max_k(dist(C_{i\,center}, C_{i\,corner_k})) < \varepsilon_b \tag{3}$$

where ε_a represents the threshold of pixel number in the subgraph, and ε_b represents the threshold of the subgraph's range in the feature space. The parameters are experimentally used by setting $\varepsilon_a = 10$ and $\varepsilon_a = 0.4$.

With the assistance of the predicate $\kappa(C_i)$, the subdivision becomes easy. The root cell D is tried first, which is the strongest bounding cube of points denoting all pixels in the Gabor feature space. If $\kappa(D)$ is false, then D is split into two cells and the splitting algorithm is performed again on each of the cells. Once this iterative clustering is finished, the resulting clusters will be highly similar to each other and well approximate the pixel neighborhood. The generated feature space is then transformed into a set of connected subgraphs.

C. Temporal Coherence With Gabor Flow

The connected subgraphs enable color propagation across the entire feature space. Compared with [1] and [7], to propagate color across all the pixels in the inherently interconnected image space, the pixels form the connected subgraphs as isolated islands in the feature space. If the feature space is not fully connected, additional user input is needed to ensure there is at least one color assignment per subgraph. In order to avoid complex user editing and to maintain temporal coherence, the temporal connectivity among the pixels in different subgraphs is preferred so that the color can automatically spread over the whole feature space. In a sense, temporal connectivity indicates to what degree our colonization is temporally coherent and therefore resistant to flickering artifacts.

A new flow field called Gabor flow is introduced to check pixel similarity over time. Based on the rotation-aware Gabor filters, a discrete parallel match algorithm is used to identify pixel color correspondences. The rotation-aware Gabor features make it possible to robust-match across texture/color appearances at different parts of the scene. The basic task is to find robust color correspondences among unconnected clusters in the Gabor feature space. A pixel's filter banks consist of p elements, $F_k, k = 1, \ldots, p$, each posing a local constraint on pixel movement, which is used to define Gabor flow energy as

$$E_{GF}(u, v) = \sum_{k=1}^{p}((F_k)_x u + (F_k)_y v + (F_k)_t)^2 \tag{4}$$

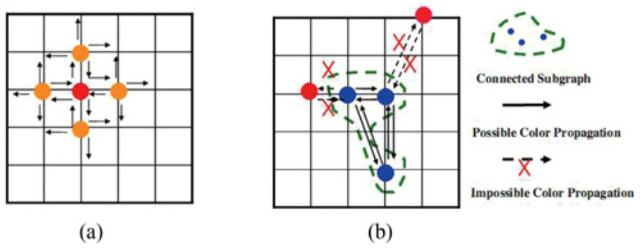


Fig. 5. Comparison of color propagation in 2-D image space with that in the Gabor feature space. (a) Color propagation in image spatial space. (b) Color propagation in our feature space.

$$\begin{pmatrix} (F_1)_x & (F_1)_y & (F_1)_t \\ (F_2)_x & (F_2)_y & (F_2)_t \\ \vdots & \vdots & \vdots \\ (F_p)_x & (F_p)_y & (F_p)_t \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{0} \tag{5}$$

where u and v are the x and y components of the gabor flow. The task is reduced to replacing the system entries in (5) with the components of vector-valued Gabor features. The resulting linear system can be solved by estimating Gabor flow with the least mean square method. Specifically, Gabor flow is defined in a rotation-aware manner by formulating the temporal connectivity of the pixels among subgraphs in the feature space.

As discussed earlier, our energy optimization can propagate colors to neighboring pixels regardless of spatial connectivity, as shown in Fig. 4. Fig. 5 shows the comparison of color propagation in 2-D image space and Gabor feature space.

IV. Efficient Color Propagation

In this section, we show how cluster-based pixel neighborhoods support efficient color propagation in the Gabor feature space. There are many energy formulations for color propagation [1], [2], [8], [10]. For the formulation in this paper, the energy minimization that determines the neighborhood in the Gabor feature space must be considered, in comparison with the well-known eight-neighbor area in the image space. For pixel i, its neighbors in the Gabor feature space, its neighbors $(N_F(i))$ within one subgraph, and its temporal correspondences determined by Gabor flow are found out.

After this neighborhood is formulated and as the intensity value Y and color hints scribbled by the user are known, the values of components U and V (in space YUV) should be estimated. As the methods to estimate V and U are

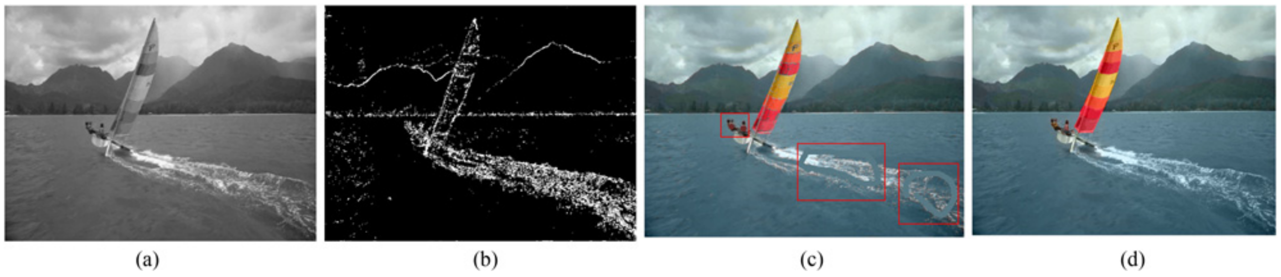


Fig. 6. Color propagation for a sample video. (a) Grayscale video frame at time t. (b) Masking map for this video frame, in which black pixels indicate regions of high accuracy in color propagation. (c) Further refinement is applied to the pixels which are marked in white in the masking map. (d) Final coloring result.

similar, only the calculation of U is described, referring to the notation in [1] for simplicity. We aim to minimize the difference between the weighted average of the colors at feature-neighboring pixels and the color component $U(i)$ at pixel i

$$\begin{aligned} E(U) &= \sum_i (U(i) - \sum_{k \in N_F(i)} W_{ki} U(k))^2 \\ W_{ki} &\propto e^{-dist(k,i)/2\sigma_F^2(i)}, \; k \in N_F(i) \qquad (6) \\ W_{ki} &= 0, \; \text{otherwise} \end{aligned}$$

where W_{ki} is a weighting function that sums to 1, and it is larger when its feature vector is identical to that of pixel i, $\sigma_F(i)$ is the variance of all feature distances between pixel i and its neighbors. The notation $k \in N_F(i)$ indicates that k and i are neighboring pixels in the feature space. Then, the weighting functions are computed on the basis of the feature distances between pixel i and its neighbors.

Given a set of locations i_{user} with user-specified colors $u(i_{user}) = u_{scribble}$; $v(i_{user}) = v_{scribble}$, $E(U)$ and $E(V)$ subjected to the specified color constraints are minimized. With the quadratic cost functions and the linear constraints, this optimization problem brings up a large set of linear equations, which can be solved by standard methods such as generalized minimal residual algorithm (GMRES) [33]. However, the construction of coefficients in the linear system is still very time-consuming because all pair-wise terms have to be enumerated.

A. Parallel Color Propagation

One way to utilize our Gabor feature clustering in this optimization is to express the variable $U(i)$ in (6) as a linear combination of its neighbors $N_F(i)$ in the Gabor feature space, which will result in highly parallelized iterations with fewer variables to solve. The user-specified color U_{user} is propagated to feature neighbors by using

$$U^0(i) = \begin{cases} U_{user}, & \text{if } i \in stroke_{user} \\ U(i), & \text{otherwise} \end{cases} \qquad (7)$$

$$U^{l+1}(i) = \sum_{k \in N_F(i)} W_{ki} U^l(k) \qquad (8)$$

where $stroke_{user}$ is the color of the user-input scribbles. Noticeably, the propagation can be significantly accelerated by assigning it to a GPU and/or multicore processor. Interestingly, for pixels with coloration artifacts, (8) makes it possible for the user to edit colorized video frames continuously, and thus simplifies their colorization (as shown in Fig. 6).

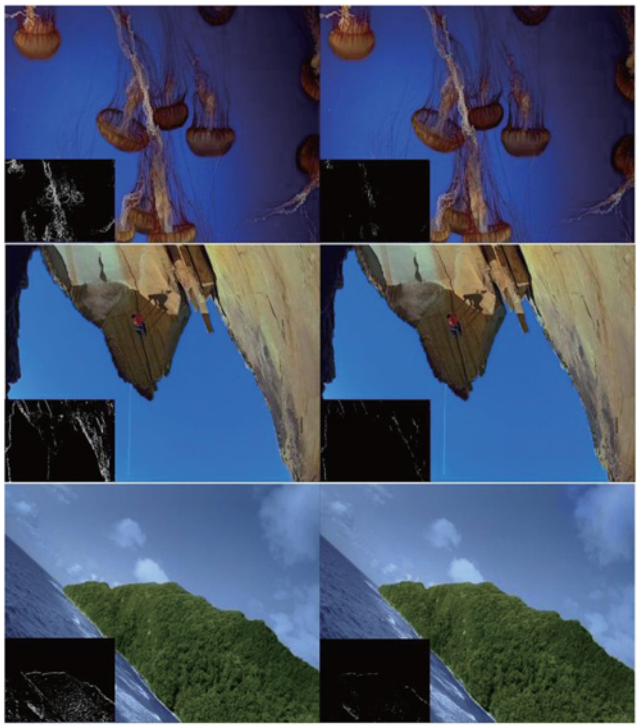


Fig. 7. Color propagation comparisons (a) using optical flow [21] and (b) using Gabor feature space. Note that the high-error region (indicated by white pixels) produced using optical flow is much larger than that produced using Gabor feature space.

B. Analysis

Video colorization can be optimized by establishing that linear iterations on per-pixel basis, so numerical analysis was performed. The reason is that color preparation indicated by (8) is, in fact, an alternative solver of (6), in which the linear system is sparse but unnecessarily symmetrical. Mathematically, we need to obtain the least-squares solution of $AU = 0$, where $A = E - (W_{kj})_{n \times n}$, E is the identity matrix. Since the pixel's neighborhood in Gabor feature space might be nonsymmetrical (there are instances when a pixel $j \in N_F(i)$ but the pixel $i \notin N_F(j)$), A might be a nonsymmetrical matrix. Therefore, we cannot use the conjugate gradient solver, and GMRES is time consuming for video data.

Fortunately, our linear system $AU = 0$ is also particular, since it can be rewritten as $U = F'U$ $(F' = E - A)$. $F'U$ is therefore proved to be a contraction mapping [34]. It is known that a contraction mapping has at most one fixed point, and

TABLE I
TIMING STATISTICS OF OUR VIDEO COLORIZATION IN GABOR FEATURE SPACE

Video	Resolution	#Frames	# Keyframes	Per-frame Color Optimzation (CPU) [1]	Inter-frame Color Propagation(GPU)
Fig. 1	550×416	50	1	$\sim 66s$ (per frame)	$\sim$ 21ms (per frame)
Fig. 9	600×453	426	3	$\sim 73s$ (per frame)	$\sim$ 21ms (per frame)
Fig. 12	550×416	145	4	$\sim 81s$ (per frame)	$\sim$ 30ms (per frame)
Fig. 16	550×416	300	3	$\sim 75s$ (per frame)	$\sim$ 19ms (per frame)
Fig. 18(a-c)	600×453	682	5	$\sim 79s$ (per frame)	$\sim$ 20ms (per frame)
Fig. 18(d-f)	550×416	153	2	$\sim 71s$ (per frame)	$\sim$ 19ms (per frame)

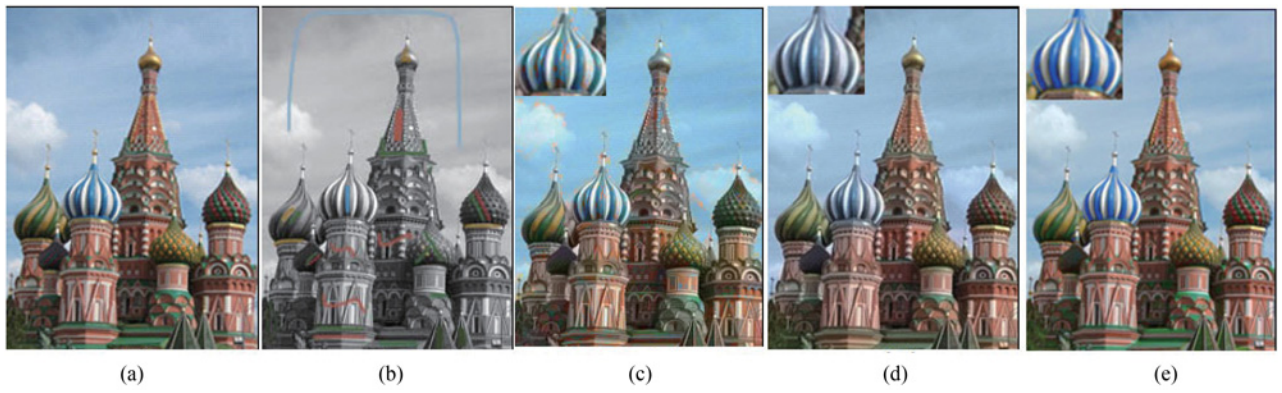


Fig. 8. Comparison of colorization results using the methods by Liu *et al.* [8] and Irony *et al.* [7]. (a) Ground-truth image of St. Basil's Cathedral for comparison. (b) Target grayscale image with user's strokes. (c) Irony *et al.*'s colorization result. (d) Liu *et al.*'s colorization result [29]. (e) Our colorization result. The insets in the top left corner of (c), (d) and (e) provide details of the different colorizations of the blue and white pinnacle. Our method shows a more realistic result.

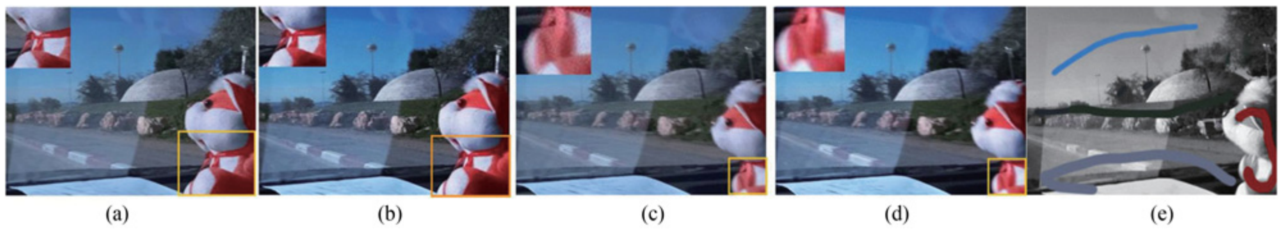


Fig. 9. Colorization comparison of the sample video (33 frames). (a) and (c) Colorization by Irony *et al.* [7]. (b) and (d) Results by our approach. (e) Input color scribbles.

we can use the iterated function $U_{n+1} = F'U_n$ to converge to the fixed point. We have derived (8) so as to provide a parallel, iterated function for solving (6). Therefore, parallel propagation is an approximation of solving the optimization in the Gabor space of the input video. Our proposed technique, when used for color propagation among video frames, yields equivalent optimization effects without the time-consuming calculations of optimization.

V. RESULTS AND DISCUSSION

Several experiments were performed on an Intel(R) Core TM(2) 2.3 GHz PC with nVidia GeForce 8800 GPU and 3 GB RAM. The resolution of our target videos range from 400×300 to 800×600. Preliminary tests show that only three to five keyframes of a 300-frame video are needed to achieve high-quality colorization.

Color propagation was GPU-accelerated using nVidia CUDA language for parallel computation, where the Gabor feature space [30] is extended to the k-d clusters of video pixels. In general, full propagation for each video frame was achieved within 3000 iterations, which equated to real-time performance in our environment. Table I shows the processing times for colorization of the test videos. In contrast to previous video colorization methods that relied on the spatial connectivity of image pixels [1], our Gabor feature-based colorization was able to handle most of the rotated and disjoint textures commonly seen in videos of natural settings.

Fig. 7 shows the comparison of the coloration of various videos (of *Jellyfish*, *Mountain Climbing*, and *Ocean Cruising*) using optical flow and Gabor feature space. Noticeably, the Gabor feature space is able to detect texture regions more accurately, leading to fewer errors in region matching. The k-d tree clustering of the Gabor feature space appears to be particularly well suited for propagating color for optimization. Because it is designed to measure pixel similarity, parallel iteration can optimize colors in both the feature space

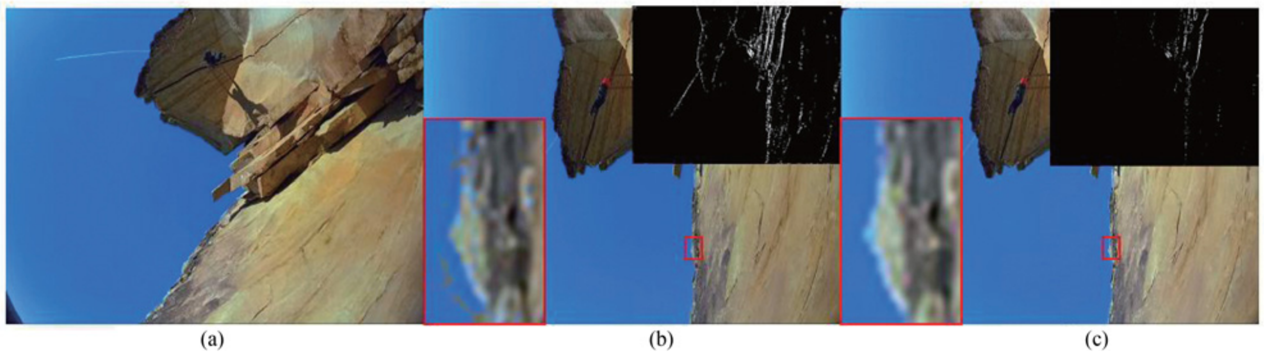


Fig. 10. Colorization using Gabor filters (b) without and (c) with rotation-aware properties, and the (a) colorization of another keyframe.

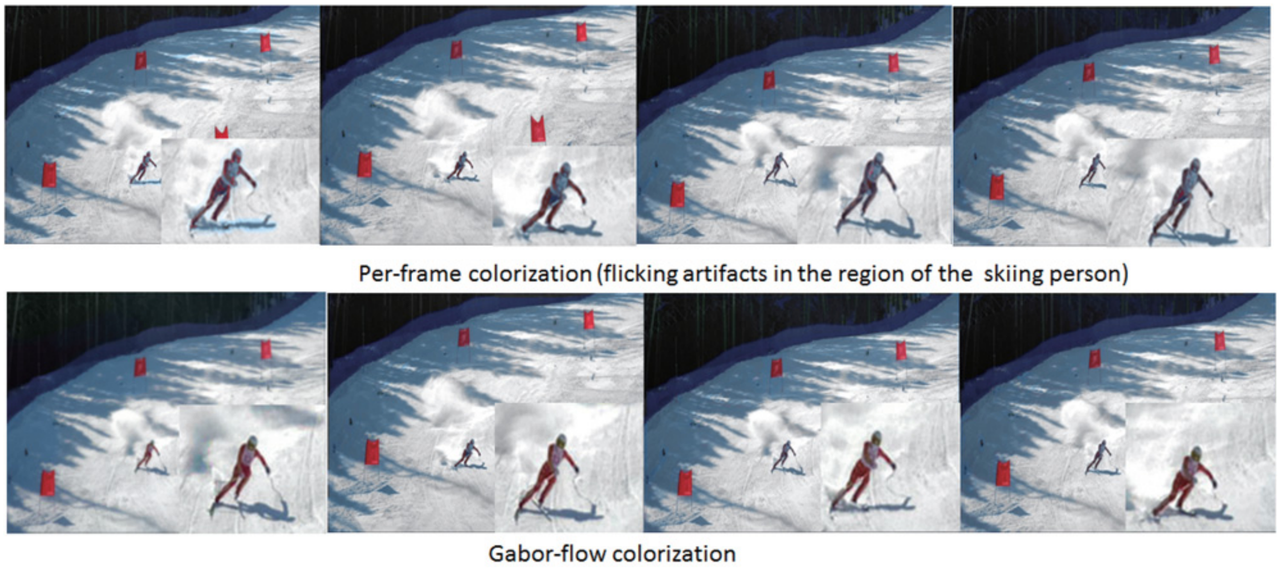

Per-frame colorization (flicking artifacts in the region of the skiing person)
Gabor-flow colorization

Fig. 11. Visual comparison of the per-frame colorization and our approach.

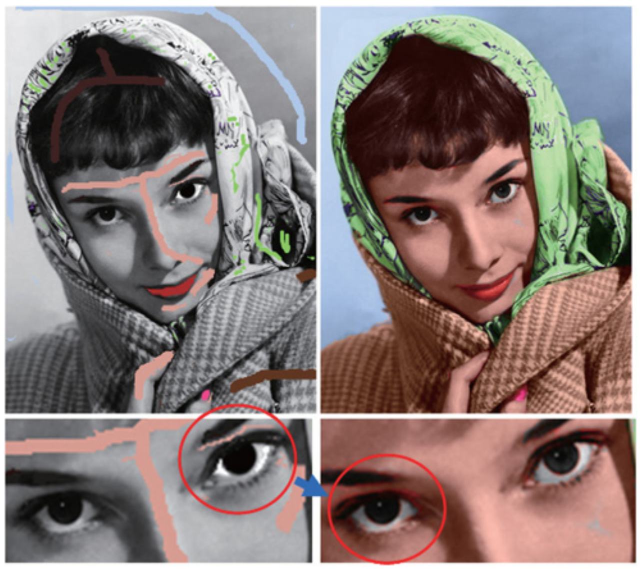


Fig. 12. Colorization using optimization in feature space. (a) Input strokes. (b) Colorization result. Please note that the input colors only existing in the right eye are propagated to the left side.

and the temporal pixel neighborhood of target video frames. Fig. 8 shows that though the reference images are unclear, our method works well given an adequate set of user-input color scribbles. Such input is crucial when reference images

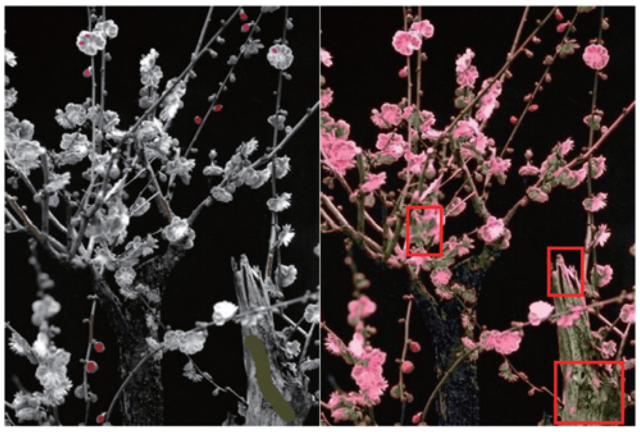


Fig. 13. Colorizing texture with multiple colors. (a) Input scribbles. (b) Our colorization result. Note that the red boxes indicate where colors have been improperly blended in texture-similar regions.

are scarce or no coherent color/illuminance match can be found.

Compared to the colorization methods in [7] and [8], our approach takes advantage of parallel color propagation in Gabor feature space to approximate the optimization in the video domain. As we know that the scribble-based colorization for video sequence may cause some flicking artifacts especially for occluded pixels, discontinuous motion boundaries, and

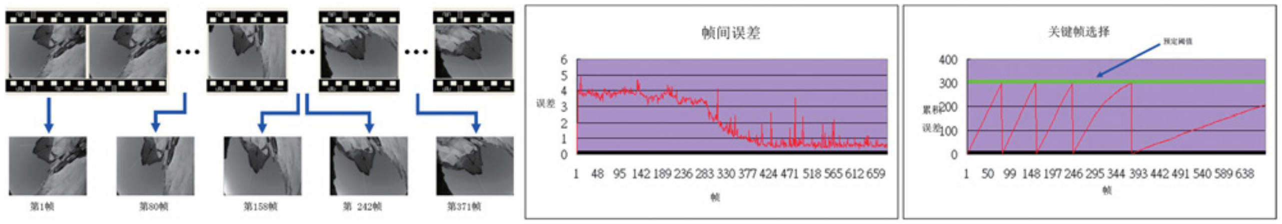


Fig. 14. Keyframe selection based on Gabor flow (from left to right): the keyframes selected from a *Climbing* video of 682 frames; the interframe accuracy error evaluated by Gabor flow; and the keyframe selection based on thresholding the accumulated accuracy error.

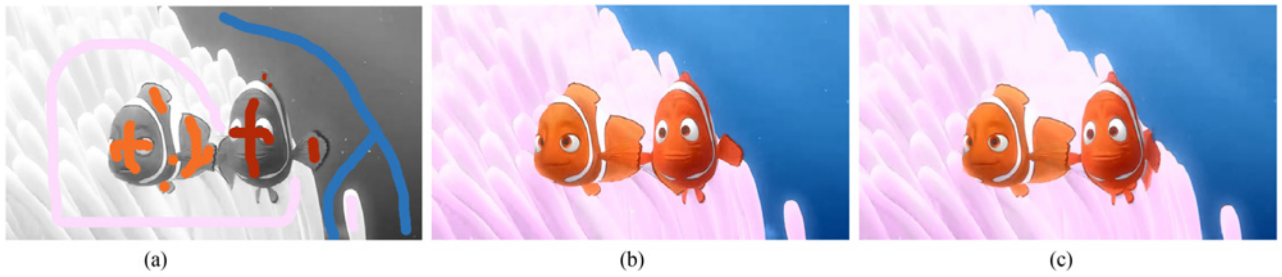


Fig. 15. Colorization results of the *Fish* video (ten frames). (a) User inputs on the grayscale frame 1. (b) Colorized result of the frame 1. (c) Colorized frame 10.

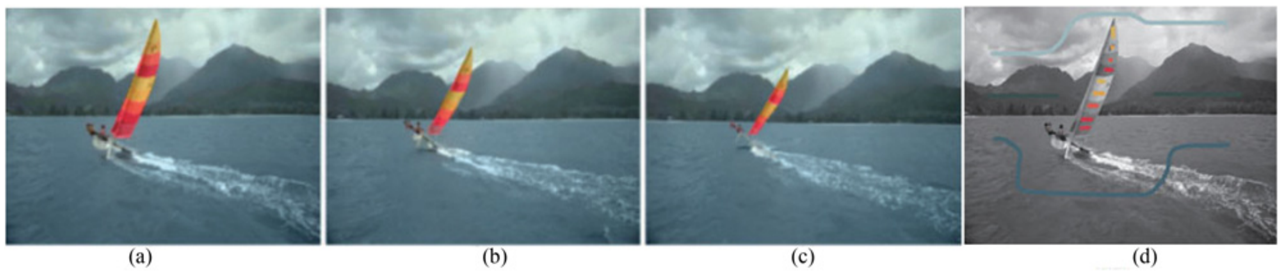


Fig. 16. Colorization results of the *Sailing* video (145 frames). (a) Colorized frame 1. (b) Frame 16. (c) Frame 44. (d) Color scribbles.

textureless regions. Inspired by the color-segmentation-based approaches, we apply temporally coherent color propagation based on the Gabor flow, and the flicking artifacts over video frames are suppressed, as shown in Fig. 11. Fig. 9 compares our colorization results with [7] given only four input color scribbles. Note that the Gabor feature-based colorization achieves comparable quality, and accurately handles rotated and disjoint regions. Fig. 12 demonstrates our colorization propagation based on the texture similarity. Fig. 10 shows the difference using Gabor filters without [Fig. 10(b)] and with [Fig. 10(c)] rotation-aware properties. Note that coherent colorization may require refinement of colors in some detailed regions of the video frames.

Figs. 15–20 show our video colorization results for various video sequences: *Fishes at Sea* (ten frames), *Sailing* (145 frames), *Ocean Cruising* (300 frames), *Mountain Climbing* (682 frames), *Skiing* (153 frames), *Tiger* (548 frames), and *Dolphin* (832 frames) videos. Note that in all these examples, colors are accurately assigned based on a small number of input color scribbles.

A. Video Length

In most cases, the colorization distortion of video data increases with time. We solve this problem by using Gabor flow to extract regular keyframes for user color scribbling. The test described in [35] is adopted to determine the accuracy of Gabor flow, based on forward/backward error maps. The keyframes at which the accumulated error reached a preset threshold were selected. Fig. 14 shows the selection of keyframes for *Climbing* (5), with interframe accuracy error and accumulated error.

B. User Scribbles

Scribble-based colorization was first introduced for static images [1], [10]. With these algorithms, colors are specified by users at several sparse positions in an image by plotting colored strokes, based on which colors are spread to the rest of the image. The basic principle of these methods is that spatially close regions with identical appearances should be assigned with similar colors. When an entire video is to be colorized, at least one color should be provided for each subgraph. Since the number of user scribbles is also decided by the user's expectation of video content in practice, it is not the main reason for selecting user inputs. The reason behind the input scribbles is to allow users to show the needed coloring effects by using scribbles. With the color specified at each scribble position, Gabor flow can propagate color to regions with similar appearances.

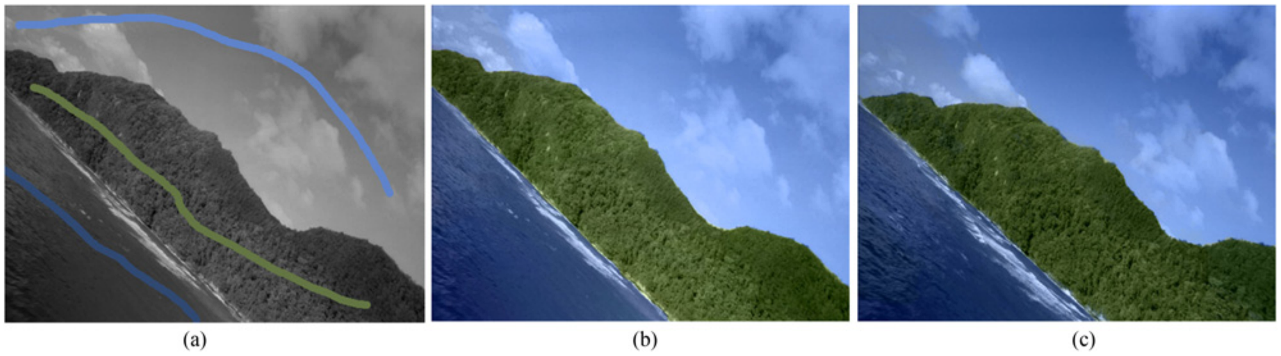


Fig. 17. Colorization results of the *Cruising* video (300 frames). (a) Color strokes on the grayscale frame 1. (b) Colorized result of frame 1. (c) Color propagated result of frame 35.

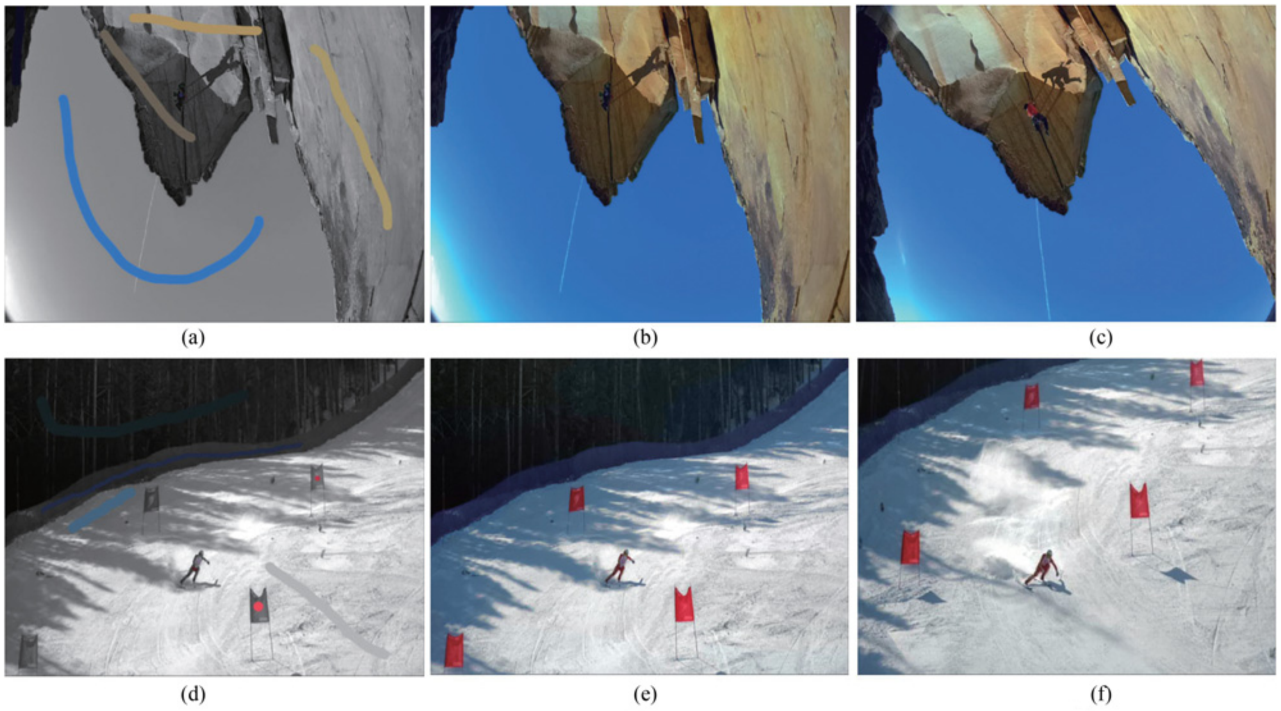


Fig. 18. Colorization results of the *Climbing* (682 frames) and *Skiing* (153 frames) videos. (a) and (d) Color strokes on the grayscale frames. (b) and (e) Colorized results of the corresponding frames. (c) and (f) Color propagated results of other video frames.

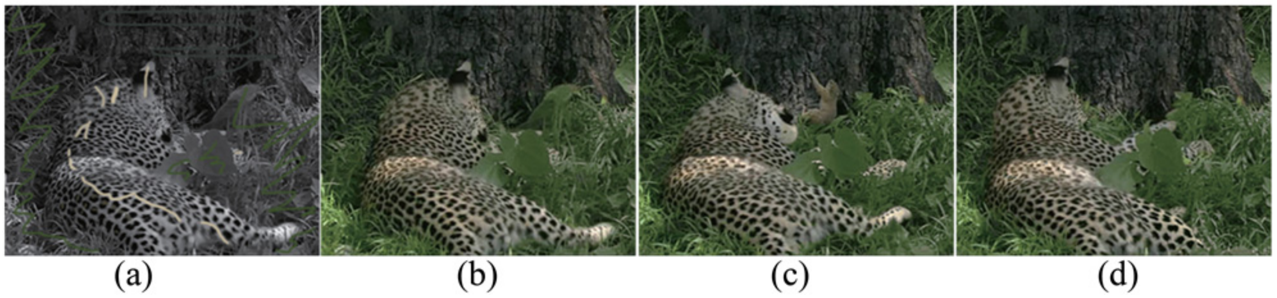


Fig. 19. Colorization results of the *Tiger* video. (a) Input user strokes for frame 1. (b) Colorized result of frame 1. (c) and (d) Color propagation results of other frames.

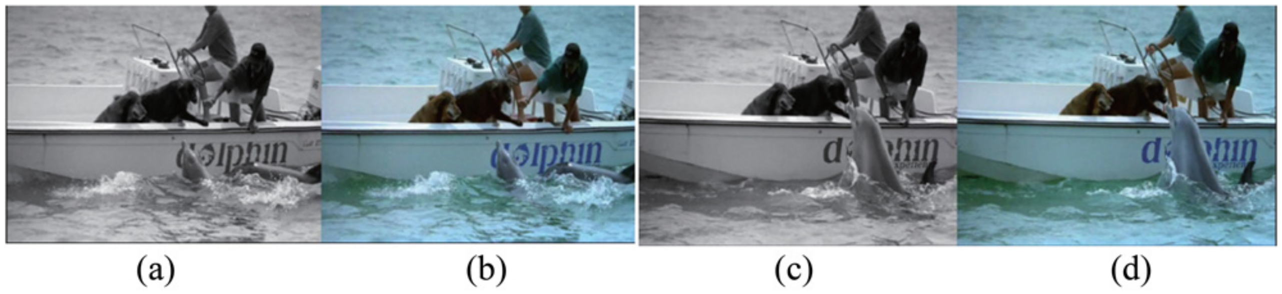


Fig. 20. Colorization results of the *Dolphin* video. (a) and (b) Input grayscale video frame and its colorized result. (c) and (d) Another input grayscale video frame and its colorized result.

C. Performance

The key to our improved colorization performance is the use of GPU-based parallelization of per-pixel color propagation. The time required for CPU-based or GPU-based colorization of test videos is provided in Table I. Performance varies with the number of color scribbles and the frame resolution. For a 550×416 video frame, the total time required for CPU-based implementation was 75 s on average, whereas that for our GPU-based implementation was about 50 ms. Colorization qualities were roughly the same for the two methods.

D. Limitation

Although our colorization approach worked well for most video examples, there were cases in which it failed to properly colorize within scribbles. This occurred when pixels with high texture similarity did not have similar colors. The plum blossom example shown in Fig. 13 is such a case: regions that share numerically similar features but are semantically distinct were not properly colorized. This limit is due to an inherent restriction of Gabor filters, resulting in overpropagation during optimization. Trusting to measured feature similarity as we do, certain spatial features (e.g., lines/boundaries) may be blurred by color propagation across object boundaries. A possible solution to this problem would be to compute a larger feature space by combining texture features and image spatial positions. Spatially proximal regions of similar appearance would then receive similar colors. Nonetheless, introducing the spatial relation into the pixel similarity seems to face the challenges of noise-removing and edge-preserving for a wide range of videos.

VI. Conclusion

The core challenge of colorization is to assign similar colors to texture-similar regions. In this paper, a novel approach to video colorization is presented, which uses the Gabor feature space to achieve good matching results, despite significant differences in appearance and spatial layout of video frames. This method is highly parallelizable. It is applicable to various video data, especially the videos of natural scenes. In the future, we will explore the use of more sophisticated monochrome texture descriptors in video sequences, and further improve the color propagation capabilities of this approach.

Acknowledgment

The authors would like to thank all reviewers for their helpful suggestions and constructive comments.

References

[1] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 689–694, 2004.
[2] Q. Luan, F. Wen, D. Cohen-Or, L. Liang, Y.-Q. Xu, and H.-Y. Shum, "Natural image colorization," in *Proc. 18th Eurograph. Conf. Rendering Tech.*, 2007, pp. 309–320.
[3] L. Yatziv and G. Sapiro, "Fast image and video colorization using chrominance blending," *IEEE Trans. Image Process.*, vol. 15, no. 5, pp. 1120–1129, May 2006.
[4] T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to greyscale images," in *Proc. SIGGRAPH*, 2002, pp. 277–280.
[5] Y.-W. Tai, J. Jia, and C.-K. Tang, "Local color transfer via probabilistic segmentation by expectation-maximization," in *Proc. IEEE Comput. Soc. Conf. CVPR*, vol. 1. 2005, pp. 747–754.
[6] E. Reinhard, M. Ashikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Comput. Graph. Appl.*, vol. 21, no. 5, pp. 34–41, Sep. 2001.
[7] R. Ironi, D. Cohen-Or, and D. Lischinski, "Colorization by example," in *Proc. Rendering Techniques*. Eurographics, Konstanz, Germany, 2005, pp. 201–210.
[8] X. Liu, L. Wan, Y. Qu, T.-T. Wong, S. Lin, C.-S. Leung, and P.-A. Heng, "Intrinsic colorization," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 152:1–152:9, 2008.
[9] Y.-C. Huang, Y.-S. Tung, J.-C. Chen, S.-W. Wang, and J.-L. Wu, "An adaptive edge detection based colorization algorithm and its applications," in *Proc. ACM Multimedia*, 2005, pp. 351–354.
[10] Y. Qu, T.-T. Wong, and P.-A. Heng, "Manga colorization," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 1214–1220, 2006.
[11] V. Konushin and V. Vezhnevets, "Interactive image colorization and recoloring based on coupled map lattices," in *Proc. Graphicon*, vol. 4. 2006, pp. 231–234.
[12] U. Lipowezky, "Grayscale aerial and space image colorization using texture classification," *Pattern Recognit. Lett.*, vol. 27, no. 4, pp. 275–286, 2006.
[13] B. Sheng, H. Sun, S. Chen, X. Liu, and E. Wu, "Colorization using the rotation-invariant feature space," *IEEE Comput. Graph. Appl.*, vol. 31, no. 2, pp. 24–35, Mar. 2011.
[14] M. Koleini, S. Monadjemi, and P. Moallem, "Film colorization using texture feature coding and artificial neural networks," *J. Multimedia*, vol. 4, no. 4, p. 240, 2009.
[15] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, nos. 1–3, pp. 185–203, 1981.
[16] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. Int. Joint Conf. Artif. Intell.*, vol. 81. 1981, pp. 674–679.
[17] D. Heeger, "Optical flow using spatiotemporal filters," *Int. J. Comput. Vision*, vol. 1, no. 4, pp. 279–302, 1988.
[18] R. March, "Computation of stereo disparity using regularization," *Pattern recognit. Lett.*, vol. 8, no. 3, pp. 181–187, 1988.

[19] M. Black and P. Anandan, “The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields,” *Comput. Vision Image Understanding*, vol. 63, no. 1, pp. 75–104, 1996.

[20] L. Alvarez, R. Deriche, J. Sánchez, and J. Weickert, “Dense disparity map estimation respecting image discontinuities: A PDE and scale-space based approach,” *J. Visual Commun. Image Representation*, vol. 13, nos. 1–2, pp. 3–21, 2002.

[21] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, “High accuracy optical flow estimation based on a theory for warping,” *Lecture Notes Comput. Sci.*, vol. 3024, pp. 25–36, May 2004.

[22] A. Bruhn, J. Weickert, and C. Schnörr, “Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods,” *Int. J. Comput. Vision*, vol. 61, no. 3, pp. 211–231, 2005.

[23] H. Nagel and W. Enkelmann, “An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 5, pp. 565–593, May 1986.

[24] A. Mansouri, A. Mitiche, and J. Konrad, “Selective image diffusion: Application to disparity estimation,” in *Proc. IEEE Int. Conf. Image Process.*, vol. 3. 1998, pp. 284–288.

[25] N. Grammalidis and M. Strintzis, “Disparity and occlusion estimation in multiocular systems and their coding for the communication of multiview image sequences,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 3, pp. 328–344, Mar. 1998.

[26] M. Proesmans, L. Van Gool, and A. Oosterlinck, “Determination of optical flow and its discontinuities using a nonlinear diffusion,” in *Proc. 3rd Eur. Conf. Comput. Vision ECCV*, 1994, pp. 294–304.

[27] L. Alvarez, R. Deriche, T. Papadopoulo, and J. Sánchez, “Symmetrical dense optical flow estimation with occlusions detection,” *Int. J. Comput. Vision*, vol. 75, no. 3, pp. 371–385, 2007.

[28] X. Bai, J. Wang, D. Simons, and G. Sapiro, “Video SnapCut: Robust video object cutout using localized classifiers,” in *Proc. ACM SIGGRAPH Papers*, 2009, p. 70.

[29] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. Freeman, “SIFT flow: Dense correspondence across different scenes,” in *Proc. ECCV*, 2008, pp. 28–42.

[30] B. S. Manjunath and W.-Y. Ma, “Texture features for browsing and retrieval of image data,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 8, pp. 837–842, Aug. 1996.

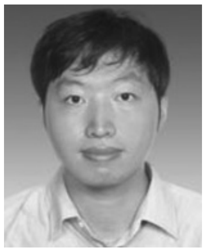
[31] A. Agarwala, “Efficient gradient-domain compositing using quadtrees,” *ACM Trans. Graph.*, vol. 26, no. 3, pp. 94:1–94:5, 2007.

[32] K. Xu, Y. Li, T. Ju, S.-M. Hu, and T.-Q. Liu, “Efficient affinity-based edit propagation using k-d tree,” *ACM Trans. Graph.*, vol. 28, no. 5, pp. 118:1–118:6, 2009.

[33] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. Philadelphia, PA, USA: SIAM, 2003.

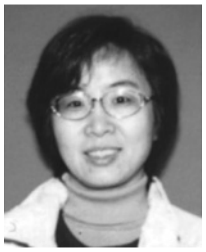
[34] A. Granas and J. Dugundji, *Fixed Point Theory*. Berlin, Germany: Springer, 2003.

[35] Y. Chuang, A. Agarwala, B. Curless, D. Salesin, and R. Szeliski, “Video matting of complex scenes,” in *Proc. 29th Annu. Conf. Comput. Graph. Interactive Tech.*, 2002, pp. 243–248.



Bin Sheng (M’13) received the BA degree in English and the B.E. degree in computer science from Huazhong University of Science and Technology, Wuhan, China, in 2004, the M.S. degree in software engineering from University of Macau, Macau, in 2007, and the Ph.D. degree in computer science from Chinese University of Hong Kong, Shatin, Hong Kong, in 2011.

He is an Assistant Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include virtual reality, computer graphics, and image-based techniques.



Hanqiu Sun (M’98) received the M.S. degree in electrical engineering from University of British Columbia, Vancouver, BC, Canada, and the Ph.D. degree in computer science from University of Alberta, Alberta, ON, Canada.

She is an Associate Professor with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, Hong Kong. Her research interests include virtual reality, interactive graphics/animation, real-time hypermedia, virtual surgery, mobile image/video synopsis and navigation, and touch-enhanced simulations.

Marcus Magnor (M’04) received the Ph.D. degree in electrical engineering from Erlangen University, Erlangen, Germany.

He is a Full Professor of computer science and the Director of the Computer Graphics Laboratory, Braunschweig University of Technology, Braunschweig, Germany. He is also an Adjunct Professor with the Department of Physics and Astronomy, University of New Mexico, Albuquerque, NM, USA. His research interests include visual computing, from image formation, acquisition, and analysis to image synthesis, display, perception, and cognition. His areas of research include, but are not limited to, computer graphics, computer vision, visual perception, image processing, computational photography, astrophysics, imaging, optics, visual analytics, and visualization.

Ping Li received the B.Eng. degree in computer science and technology from China Jinan University, Guangzhou, China. He is currently pursuing the Ph.D. degree at the Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, Hong Kong.

His research interests include image/video processing and creative media, including image/video retexturing, stylization, colorization, video summarization, and GPU acceleration.