Fast Video Recoloring via Curve-Based Palettes

Zheng-Jun Du, *Member, IEEE*, Jia-Wei Zhou, Kang Li, Jian-Yu Hao, Zi-Kang Huang, and Kun Xu

***Abstract*—Color grading, as a crucial step in film post-production, plays an important role in emotional expression and artistic enhancement. Recently, a geometric palette-based approach to video recoloring has been introduced with impressive results. It offers an intuitive interface that allows users to alter the color of a video by manipulating a limited set of representative colors. However, this method has two primary limitations. Firstly, palette extraction is computationally expensive, often taking more than one hour to generate palettes even for medium-length videos, which significantly limits the practical application of color editing for longer videos. Secondly, the palette colors are less representative, and some primary colors may be omitted from the resulting palettes during topological simplification, making it less intuitive in color editing. To overcome these limitations, in this paper, we propose a novel approach to video recoloring. The core of our method is a set of Bézier curves that connect the dominant colors throughout the input video. By slicing these Bézier curves in RGBT space, per-frame palette can be naturally derived. During recoloring, users can select several frames of interest and modify their corresponding palettes to change the color of the video. Our method is simple and intuitive, enabling compelling time-varying recoloring results. Compared to existing methods, our approach is more efficient in palette extraction and can effectively capture the dominant colors of the video. Extensive experiments demonstrate the effectiveness of our method.**

***Index Terms*—Video recoloring, palette, Bézier curves, color editing.**

I. Introduction

VIDEO recoloring aims to adjust or enhance the colors of a video to achieve a particular visual effect or feeling. It is widely used in film production, advertising, art creation, short video editing, etc. While some commercial software (e.g., Adobe Premiere, DaVinci Resolve, and Lightworks) provides strong support for video color editing, they usually require the user to learn a great deal of specialized knowledge and are therefore less user-friendly. In recent years, palette-based approaches [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14] have made significant progress in image recoloring, and have been successfully extended to video scenarios by Du et al. [15] with impressive results. Compared to commercial tools, palette-based video recoloring is simpler and more intuitive, allowing users to adjust the colors of a video by manipulating a small set of representative colors.

Du et al. [15] projected the input video into the RGBT space and treated its pixels as a 4D point set. They then generated a geometric structure, referred to as a skew polytope, to represent the color palette of the video. Finally, Mean Value Coordinates (MVC) interpolation was employed to transfer palette colors' variations to the entire video. As a pioneering work in palette-based video recoloring, their method offers an intuitive interface and enables natural, artifact-free time-varying recoloring via simple interactions. However, this approach has two primary limitations. First, palette extraction is a time-consuming process. The initial skew polytope generated for a video has a complex topological structure with thousands of vertices and edges, making it unsuitable for direct use as a palette. Thus, significant time is required to iteratively refine its topology. Second, the vertices, i.e., the palette colors, are far away from the video pixels after simplifying the skew polytope, making the palette colors lack representativeness. As a result, color editing becomes less intuitive and unpredictable.

To overcome the aforementioned limitations, we propose a novel curve-based approach for video recoloring. Our method consists of two primary steps: palette extraction and video recoloring. In the first step, we initially extract frame-wise representative colors through clustering. Next, we fit a set of Bézier curves [16] in RGBT space to connect these dominant colors across video frames, to represent the palette of the input video. In the second step, to achieve better local control and more nuanced color editing, we allow users to recolor the input video by directly adjusting local frame palettes. To this end, we first slice these Bézier curves in each frame time to obtain per-frame color palettes. After users modify the color palettes of specific frames, we refit these Bézier curves to propagate those color changes to the entire video.

We have demonstrated the effectiveness of our method on a wide range of examples. In comparison with existing approaches, we made the following contributions:

- We propose a novel geometric structure, i.e., a set of Bézier curves, to represent the color palette of a video. This structure effectively captures the dominant colors of each frame and connects them to form a set of smooth curves, resulting in better temporal consistency and natural, time-varying recoloring results.
- Our curve-based video palette can be extracted much more efficiently. Compared to existing methods, our palette extraction algorithm achieves a 400× speedup due to avoiding a large number of iterations.

Received 7 November 2024; revised 19 June 2025; accepted 13 August 2025. Date of publication 16 September 2025; date of current version 19 September 2025. This work was supported in part by the National Natural Science Foundation of China under Project 62562052 and in part by the Youth Program of the Natural Science Foundation of Qinghai Province under Project 2023-ZJ-951Q. The associate editor coordinating the review of this article and approving it for publication was Prof. Rafal K. Mantiuk. *(Zheng-Jun Du, Jia-Wei Zhou, and Kang Li contributed equally to this work.) (Corresponding author: Kun Xu.)*

Zheng-Jun Du, Jia-Wei Zhou, Kang Li, Jian-Yu Hao, and Zi-Kang Huang are with the School of Computer Technology and Application, Qinghai University, Xining 810016, China.

Kun Xu is with the Key Laboratory of Pervasive Computing, Ministry of Education, and the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: xukun@tsinghua.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TIP.2025.3607584, provided by the authors.

Digital Object Identifier 10.1109/TIP.2025.3607584

- We offer a new interaction style that allows users to recolor videos by directly modifying the color palettes of specific frames, thus significantly improving local control during video recoloring.

The remainder of the paper is structured as follows. First, we review some relevant work. Then, we describe the proposed algorithm in detail. Next, we provide extensive experiments to demonstrate the effectiveness of our method. Finally, we offer a concise conclusion and discuss the limitations of the proposed method.

II. Related Work

In this section, we first review some related works on palette-based image recoloring, followed by a brief introduction of the palette-based video recoloring approach introduced by Du et al. [15].

Palette-based approaches for image recoloring have become increasingly popular in recent years. In these approaches, a color palette [17], [18], [19], [20], [21] is typically defined as a small set of dominant colors that reflect the primary color distribution of the input image. Once the color palette of an image is obtained, a predefined color transfer function is employed to map the palette colors' variation to the whole image. After years of advancement, many palette-based image recoloring techniques have been proposed, achieving remarkable progress in color editing.

The first palette-based image recoloring method was proposed by Chang et al. [1]. They first extract the color palette using K-means clustering. Subsequently, a Radial Basis Function (RBF) based algorithm is applied to propagate the variations of the palette colors to the whole image. Zhang et al. [2] adopted a similar method for palette extraction, but encoded the color of any pixel in the input image as a weighted sum of the palette colors with an energy optimization method. In recoloring, the weights are fixed, and users modify the palette colors to adjust the appearance of the input image. Tan et al. [3] first proposed a geometric method to build color palettes. Specifically, they treat the input image as a point set in RGB space, calculate its convex hull, and use whose vertices as the color palette of the input image. Next, they decompose the input image into a set of ordered layers that can reconstruct the input by alpha blending. Finally, users adjust the color of the image by modifying the palette or the corresponding layers. Later, to improve computational efficiency and achieve better spatial consistency in recoloring, Tan et al. [4], [6] extended the approach of Tan et al. [3] to the RGBXY space. Although convex-hull-based methods are intuitive and efficient for image recoloring, the palette colors are less representative because the convex hull vertices are distant from image pixels, making recoloring less predictable. To address this problem, Wang et al. [7] proposed a geometric approach to refine the positions of convex hull vertices, improving the representativeness of the palette colors. Another drawback of convex hull-based methods is that they often introduce reconstruction errors. To address this issue, Grogan and Smolic [8] proposed a novel geometric technique for palette extraction and layer decomposition that produced layers with uniform color and faithfully maintained the colors in the input image, and achieved accurate reconstruction. Aksoy et al. [9] proposed a novel scheme to decompose the input image into a set of soft color segments, where the colors in each segment conform to a Gaussian distribution. In recoloring, users adjust the colors of the image by manipulating these soft segments. Zhang et al. [10] designed a novel blind color separation model to simultaneously extract the color palette and calculate the mixing weights, enabling more faithful palette-based recoloring. Unlike previous methods, Chao et al. [12] introduced "ColorfulCurves", which generates a hue-chroma palette and the corresponding tone curves to support sparse, per-palette color control of lightness over the image. To achieve better local control, Chao et al. [13] proposed a hierarchical palette-based approach to fulfill arbitrary image-space constraints in recoloring, and it automatically divides the image into semantic subregions to achieve color constraints using local palettes when constraints cannot be satisfied. In addition, some researchers proposed a new type of palette, the Playful Palette [22], [23], which provides artists with an interactive parametric color mixer.

In recent years, with advances in deep learning, some neural network-based approaches to image recoloring have been proposed. For example, Cho et al. [24] introduced "PaletteNet", which uses a content-aware method to match a user-selected color palette with an input image, but only supports a fixed palette size. Akimoto et al. [25] used a U-Net to calculate mixing weights of an image relative to its corresponding color palette, but often leads to unexpected global color changes during recoloring. Some other works [26], [27] extended palette-based methods to color editing in NeRF-represented scenes with impressive results.

Du et al. [15] extended geometric palette-based image recoloring techniques [3], [4], [7] to video scenario and proposed the first geometric palette-based approach to video recoloring. They first project the input video into RGBT space, and treat its pixels as a 4D (RGB color and a time) point set. Next, they build an initial 4D skew polytope by gluing adjacent frames' RGB convex hulls. Typically, the initial skew polytope has a complex topology with thousands of vertices and edges, making it unsuitable for direct use as a palette. So, they further perform block merging, redundant vertices removing, and vertex refining, respectively, to obtain a simplified skew polytope, and whose vertices serve as the video palette. In recoloring, Mean Value Coordinates [28] interpolation is employed to transfer the palette's color variation to the whole video. Their method provides an intuitive GUI for color editing, and achieves natural, time-varying recoloring results. However, this method has two primary limitations. First, it is time-consuming in palette construction, as it involves numerous iterations in block merging, vertex removal, and vertex refinement. This severely limits its practical use for longer videos. Second, the palette colors are less representative, as they are far from the video pixels after simplification, making color editing less intuitive.

III. Motivation

Our goal is to generate a palette from a given video that enables simple, natural, and time-varying recoloring. To

accomplish this, the extracted palette for a video should meet four criteria: (1) The palette should be capable of capturing the dominant colors in each frame, allowing users to easily select these colors to edit the video. (2) Users should be able to generate time-varying effects, for example, converting a video featuring a forest in the fall into a forest that fades from spring to fall. (3) The palette should maintain good temporal consistency between adjacent frames, ensuring a smooth gradient of colors over time. (4) Palette extraction and color editing should be as efficient as possible.

To meet the criteria mentioned above, Du et al. [15] proposed to use a 4D skew polytope to represent the color palette of a video. However, the construction process of the skew polytope is computationally expensive, necessitating numerous iterations to simplify the initial skew polytope with complex topological structures. Furthermore, the resulting palette colors may be less representative as the vertices gradually move away from the video pixels during the simplification process.

To address these issues, we seek a novel geometric structure, i.e., a set of Bézier curves, to represent the color palette of a video. As a widely used tool in Computer-Aided Design (CAD) and computer graphics, the Bézier curve is highly suitable for our task. Firstly, compared to the skew polytope, the Bézier curve has a simpler topology and can be fitted more efficiently. Secondly, each fitted Bézier curve connects the dominant colors that smoothly transition over time, making it more representative than those from existing methods. Lastly, the Bézier curve has excellent continuity, ensuring good temporal consistency between adjacent frames. This attribute is particularly beneficial for producing smooth, time-varying recoloring results.

IV. Method

A. Curve-Based Video Palettes

In this subsection, we define our curve-based video palette, the slicing operator, and introduce the interaction style of our approach to video recoloring.

1) Definition: For a video I with n frames, in this paper, we define its color palette Z as a set of Bézier curves in RGBT space. Specifically, the i-th Bézier curve is defined as:

$$Z_i(\tau) = \sum_{j=0}^{k-1} V_i^j B_{j,k}(\tau). \tag{1}$$

Where k denotes the number of control points, V_i^j denotes the j-th control point of Z_i, $\tau \in [0, 1]$ is a parameter, and $B_{j,k}(\tau)$ is the Bernstein basis function [29]:

$$B_{j,k}(\tau) = \binom{k}{j} \tau^j (1-\tau)^{k-j}. \tag{2}$$

2) Slicing: Once the Bézier curve-based video palette is generated, any frame j's color palette P_j can be naturally obtained by slicing the video palette i.e., a set of Bézier curves with a hyper-plane $t = \frac{j}{n-1}$ in RGBT space:

$$P_j = \text{Slice}(Z, j) = \bigcup_{i=0}^{m-1} \left\{ Z_i \left(\frac{j}{n-1} \right) \right\}. \tag{3}$$

Where m is the number of Bézier curves. For the sake of description, a frame's color palette is also referred to as the frame palette.

3) Interaction Style: In the approach of Du et al. [15], the user recolors a video by modifying its global palette colors. Although this approach is intuitive, it does not support user-specific editing on local frames. To achieve better local control during video recoloring, we allow the user to recolor a video by directly adjusting the color palettes of frames of interest.

To smoothly propagate local frames' color palette changes to the rest frames, we refit a set of Bézier curves according to the edited frame palettes, and then get the propagated frame palettes by slicing the refitted Bézier curves at each frame time. Finally, each frame is recolored with frame palettes before and after editing. Thanks to the superior continuity of Bézier curves, our method can easily generate videos with smooth color transition effects.

In general, our Bézier curve-based video palette offers more flexible interaction and can produce smoother recoloring results than existing methods.

B. Overview

Our goal is two-fold. First, we want to build a set of Bézier curves that connect the representative colors in each video frame, to represent the color palette of an input video. Second, to achieve better local control in color editing, we allow users to recolor a video by directly modifying the color palettes of specific frames, rather than altering the global video palette. To this end, we propose a two-stage approach to video recoloring.

(1) **Video palette extraction**. We first extract the representative colors from each frame to build several color-transition sequences. We then fit a set of Bézier curves in RGBT space, with each curve connecting a sequence of dominant colors that smoothly transition over time, forming the color palette of the input video. See Sec. IV-C for more details.

(2) **Video recoloring**. After users make edits to the color palettes of specific frames, we first refit a set of Bézier curves to propagate the variations of local frame palettes to the remaining frames. We then recolor all frames based on their color palettes before and after editing. See Sec. IV-D for more details.

The full pipeline of our method is illustrated in Fig. 1.

C. Video Palette Extraction

Our video palette extraction algorithm takes a video I with n frames as input and outputs m Bézier curves $\{Z_0, Z_1, \cdots, Z_{m-1}\}$ as the color palette of the input video. To build such a video palette, we first estimate the number m. We then extract dominant colors from each frame to build m time-varying color sequences. Finally, we fit m Bézier curves over these color sequences to generate the color palette.

1) Estimating the Number of Bézier Curves: Here, we adopt a simple scheme to automatically estimate the value of m. Specifically, we first uniformly sample a frame sequence by selecting one frame every 5 frames in the video, and take the average number of dominant colors across these sampled

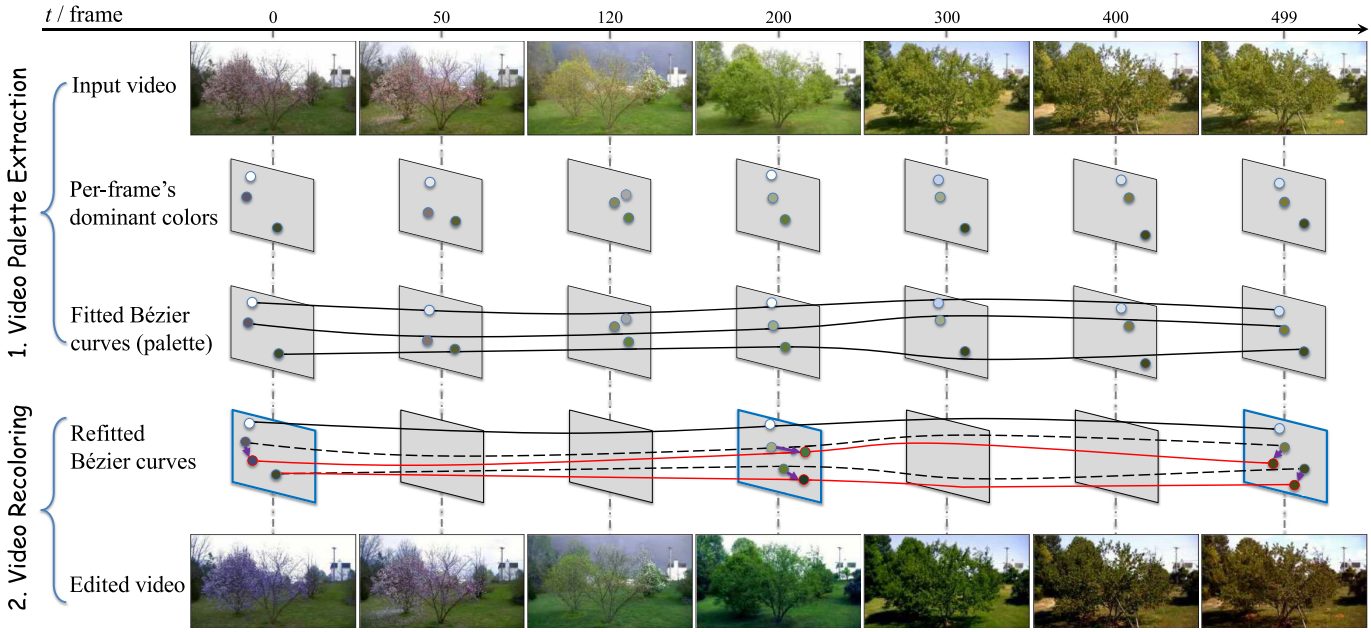


Fig. 1. The pipeline of our approach. Our method consists of two primary stages: video palette extraction and video recoloring. In stage 1, we first estimate the number of Bézier curves. Then, we extract representative colors in each frame to create a set of color transition sequences. Finally, we fit a Bézier curve for each sequence, which collectively forms the color palette of the input video. Meanwhile, each frame's color palette can be naturally obtained by slicing the Bézier curves. In stage 2, we begin by refitting a new set of Bézier curves according to users edits on specific frames' color palettes, and then recolor each frame according to its frame palettes before and after editing. Our method is efficient and can generate natural, smooth, and time-varying recoloring results. In this example, the user modified the color palettes of the 1st, 200th, and last frames, resulting in the flowers changing to purple at the beginning, the trees turning lush green, and eventually fading to yellow towards the end.

frames as the value of m. For each sampled frame, the dominant colors are obtained as follows.

(1) Segment the frame into superpixels (denoted as $\mathcal{S}$) using SLIC [30].

(2) For any superpixel $\mathcal{S}_i$, we use the centroid C_i as its representative color, and assign C_i a saliency weight θ_i, which is initialized to the pixel count in $\mathcal{S}_i$.

(3) Select the color C_i with the maximum saliency weight. To attenuate the saliency weights of other similar colors, the saliency weight θ_j of any candidate color C_j is updated by:

$$\theta_j = (1 - \exp(-d_{i,j}^4/\sigma^4)) \cdot \theta_j. \tag{4}$$

Where $d_{i,j}$ denotes the Euclidean distance between C_i and C_j in LAB space, and σ is a falloff (default 80).

(4) Repeat step (3) until the current maximum saliency weight falls below a predefined threshold ε (default 4).

2) Building Time-Varying Color Sequences: After determining the number of Bézier curves, we build m time-varying color sequences for fitting the Bézier curves.

A straightforward approach to build these sequences is to extract the dominant color per frame with k-means clustering independently and then perform one-to-one matching between adjacent frames. Although intuitive, k-means clustering is sensitive to initial values setting, it may lead to matched colors of adjacent frames being completely different. As a result, the fitted Bézier curves may not accurately capture the time-varying dominant colors.

We observe that colors in adjacent frames are generally similar and change smoothly, which implies that their clustering results should also be similar. Based on this observation, instead of performing clustering on each frame independently, we design a progressive method to simultaneously capture per-frame dominant colors and perform inter-frame matching.

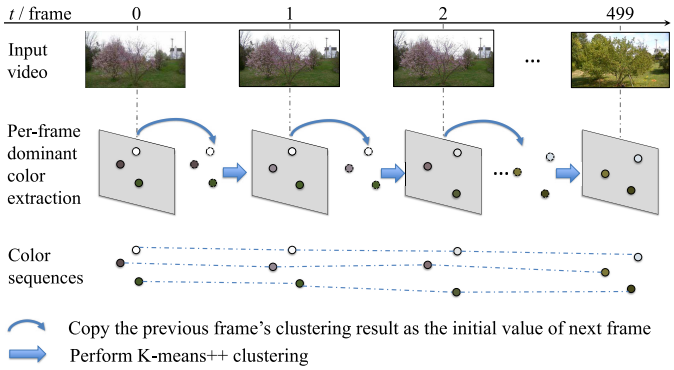


Fig. 2. Our progressive method for extracting per-frame dominant colors and building color sequences.

Specifically, as shown in Fig. 2, we first perform k-means++ clustering [31] on frame 0, to generate its m dominant colors $\{C_0^0, C_0^1, \cdots, C_0^{m-1}\}$ (subscript and superscript denote frame and color indices, respectively). Next, we take $\{C_0^0, C_0^1, \cdots, C_0^{m-1}\}$ as the initial values of K-means++ to generate frame 1's dominant colors $\{C_1^0, C_1^1, \cdots, C_1^{m-1}\}$. We repeat this process until all frames' dominant colors are obtained. As a result, the dominant color C_j^i ($i \in [0, m)$) in frame j naturally corresponds to the most similar color C_{j+1}^i in frame $j+1$.

The resulting m sequences of time-varying dominant colors can be expressed as:

$$C = \{C_{i\in[0,m)}\} \text{ and } C_i = \{C_0^i, C_1^i, \cdots, C_{n-1}^i\}. \tag{5}$$

3) Fitting Bézier Curves: Now we fit a Bézier curve Z_i for each color sequence C_i. To encourage the fitted Bézier curve to accurately capture the dominant colors in each sequence, we define an energy function to assess its quality. Specifically, the energy function is defined as:

$$E_{\text{fit}} = \frac{1}{n}\sum_{j=0}^{n-1}\left\| Z_i\left(\frac{j}{n-1}\right) - C_j^i \right\|^2. \tag{6}$$

Here, $\frac{j}{n-1}$ denotes the normalized time of frame j, $Z_i(\cdot)$ denotes the value of Z_i at that time, C_j^i denotes the extracted i-th representative color of frame j. The purpose of this energy function is to encourage the fitted and extracted dominant colors in each frame to be as close as possible.

Algorithm 1 Video Palette Extraction

Input:
a video I with n frames
Output:
m Bézier curves to represent the color palette of I

uniformly sample $\kappa = n/5$ frames in the video
for $i \leftarrow 0$ to $\kappa - 1$ **do**
  extract m_i dominant colors
end for
$m \leftarrow (\sum_i^{\kappa} m_i)/\kappa$
perform K-means++ clustering on frame 0 to generate its m dominant colors: $\{C_0^0, C_0^1, \cdots, C_0^{m-1}\}$
for $i \leftarrow 1$ to $n - 1$ **do**
  perform K-means++ clustering on frame i using $\{C_{i-1}^0, C_{i-1}^1, \cdots, C_{i-1}^{m-1}\}$ as the initial value, to generate its m dominant colors: $\{C_i^0, C_i^1, \cdots, C_i^{m-1}\}$
end for
for $i \leftarrow 0$ to $m - 1$ **do**
  build a sequence of time-varying colors across all frames: $C_i = \{C_0^i, C_1^i, \cdots, C_{n-1}^i\}$ (Eq. 5)
  fit a Bézier curve Z_i on C_i (Eq. 6)
end for
return $Z = \{Z_0, Z_1, \cdots, Z_{m-1}\}$ as the video palette

Now we obtain the optimal Bézier curve Z_i characterized by k control points $\{V_i^0, V_i^1, \cdots, V_i^{k-1}\}$ (we empirically set $k = \max(6, \lfloor n/30 \rfloor)$) by minimizing Eq. 6. By fitting Bézier curves on all m time-varying color sequences, we construct the color palette of the input video:

$$Z = \{Z_0, Z_1, \cdots, Z_{m-1}\}. \tag{7}$$

The pseudo code of our video palette extraction algorithm is given in Algorithm 1.

D. Video Recoloring

During video recoloring, we allow users to alter local frame palettes to recolor the input video. To smoothly propagate user edits to other frames, we first refit the Bézier curves based on edits to specific frame palettes. The edited frame palettes are then obtained by slicing the refitted curves at each frame time. At last, we employ a color transfer function to map the changes in each frame's palette to all pixels in that frame, achieving time-varying video recoloring.

1) Refitting the Bézier Curves: We denote the edited frame palettes as $\widetilde{P} = \{\widetilde{P}_{i\in F}\}$, where F is the index set of edited frames. Our goal is to refit a set of Bézier curves according to user edits, propagating the palette variations from local frames to the rest of the video. Once the Bézier curves are refitted, each frame's edited color palette can be obtained by slicing the refitted curves. Benefiting from the excellent continuity of Bézier curve, user edits on local frames can smoothly propagate to other frames.

We design an energy function to measure the quality of the refitted Bézier curves Z'. A suitable energy function should take into account both the edit intention and the color trends in the original video. Specifically, our energy function is defined as the weighted sum of an edit intention term E_{edit} and a color trends term E_{trend}:

$$E_{\text{refit}} = E_{\text{edit}} + \lambda E_{\text{trend}}. \tag{8}$$

Where λ is a weight parameter to balance the relative contribution of these two terms, we empirically set $\lambda = 0.1$ for all examples.

The edit term forces the refitted Bézier curves to pass through all modified frame palettes' colors, to satisfy users' edit intentions. It is defined as:

$$E_{\text{edit}} = \frac{1}{|F|}\sum_{i\in F}\sum_{j=0}^{m-1}\left\| Z_j'\left(\frac{i}{n-1}\right) - \widetilde{P}_i^j \right\|^2. \tag{9}$$

Where Z_j' is the j-th Bézier curve to be fitted, $Z_j'(\cdot)$ denotes the value of Z_i' at the normalized time of frame i, $\widetilde{P}_i^j$ is the j-th edited palette color of frame i.

The color trend term encourages the color trends (or the shapes) of refitted Bézier curves to be as similar to the original ones as possible. It is defined as:

$$E_{\text{trend}} = \frac{1}{n}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}\left\| \nabla Z_j'\left(\frac{i}{n-1}\right) - \nabla Z_j\left(\frac{i}{n-1}\right) \right\|^2. \tag{10}$$

Here, $\nabla Z_j(\cdot)$ and $\nabla Z_j'(\cdot)$ denote the first derivatives of the original curve Z_j and the refitted curve Z_j' in frame i, respectively.

By minimizing Eq. 8, we get the refitted Bézier curves Z'. Any frame i's edited color palette can be obtained via the slicing operator i.e., $P_i' = \text{Slice}(Z', i)$. Next, we describe per-frame recoloring using the edited frame palettes.

2) Recoloring Per-Frame: Given an input video I, the original frame palettes $P = \{P_i\}$ and the edited frame palettes $P' = \{P_i'\}$, our goal is to recolor each frame to obtain the recolored video I'. To this end, we design a color transfer function, similar to Chang et al. [1], to map the variation of each frame palette to its all pixels. Specifically, for any pixel $I_{i,j}$ (where i is the frame index and j is the pixel index), its edited color $I_{i,j}'$ is defined as:

$$I_{i,j}' = I_{i,j} + \sum_{k=0}^{m-1} w_{i,j}^k \left(P_{i,k}' - P_{i,k}\right) \text{ and } \sum_{k=0}^{m-1} w_{i,j}^k = 1. \tag{11}$$

Here, $P_{i,k}$ and $P'_{i,k}$ denote the k-th color in the original and edited frame palettes P_i and P'_i, $w^k_{i,j}$ denotes the normalized similarity weight of $I_{i,j}$ to the palette color $P_{i,k}$, defined as:

$$w^k_{i,j} = \frac{S_k\left(I_{i,j}\right)}{\sum_{r=0}^{m-1} S_k\left(I_{i,j}\right)}. \tag{12}$$

Here, $S_k\left(I_{i,j}\right)$ is a function to measure the similarity between $I_{i,j}$ and the palette color $P_{i,k}$, it is defined as:

$$S_k(x) = \sum_{i=0}^{m-1} \delta_{k,i}\phi\left(x, P_{i,k}\right). \tag{13}$$

Here, $\delta_{k,i}$ are weight coefficients to be determined, $\phi\left(x, P_{i,k}\right)$ is a radial basis function which is defined as:

$$\phi\left(x, P_{i,k}\right) = \exp(-(x - P_{i,k})^2/2\sigma^2). \tag{14}$$

Here, σ is determined by calculating the mean distance of all pairs of colors in P_i.

Eq. 13 depicts the similarity between a pixel and a palette color. We impose the constraints: $S_k(x) = 1$ if $x = P_{i,k}$, and $S_k(x) = 0$ if x is identical to any other color in P_i. This enforces that each palette color is most similar to itself (similarity 1) and completely dissimilar to other palette colors (similarity 0). Based on this constraint, we can construct a linear system with m^2 equations to solve for weight coefficients $\delta_{k,i}$. Once these coefficients are determined, the similarity weights (Eq. 12) and the recolored frame (Eq. 11) can be derived accordingly.

Algorithm 2 Video Recoloring

Input:
the input video I with n frames
the video palette $Z = \{Z_0, Z_1, \cdots, Z_{m-1}\}$
the edited frame palettes $\widetilde{P}$

Output:
the recolored video I'

generate the original frame palette set P by slicing the fitted Bézier curves Z (Eq. 3)
refit a set of Bézier curves Z' based on the edited frames palettes $\widetilde{P}$ (Eq. 8)
generate the edited frame palette set P' by slicing the refitted Bézier curves Z' (Eq. 3)
for each frame I_i in the input video I **do**
  for each pixel $I_{i,j}$ in the frame I_i **do**
    calculate its edited color $I'_{i,j}$ (Eq. 11)
  end for
end for
return the recolored video I'

The pseudo code of the full process of video recoloring is given in Algorithm 2.

Note that while our video palette is essentially a set of high-degree polynomials, it does not lead to high-degree problems. For example, modifying a local part of the Bézier curve will not cause unpredictable global changes in its shape. This is primarily achieved through our designed energy function, which ensures that the fitted Bezier curve remains faithful to the user's editing intent while also preventing drastic changes in its shape. Consequently, edits made to specific frames can be smoothly, as expected, propagated to other video frames.

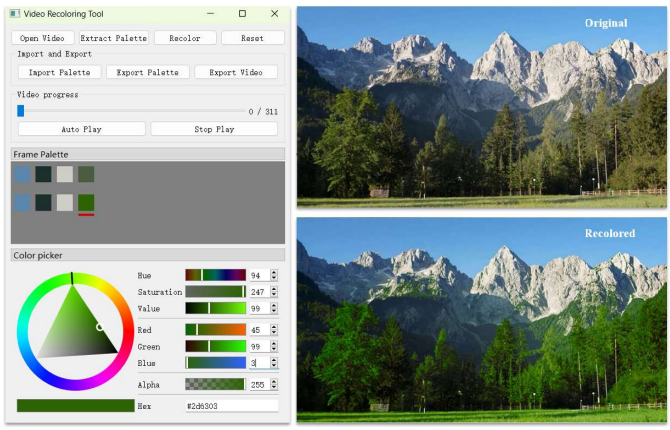


Fig. 3. Our video recoloring GUI. In this tool, the primary function panel (including opening and saving videos, palette extraction, a preview slider, and palette adjustment) is located on the left side. The input and recolored videos are displayed side by side on the right, allowing for intuitive comparison.

V. Experiments

We perform all experiments on a desktop computer equipped with an Intel Core i7-11700 2.5 GHz CPU and 16 GB RAM. Our algorithm is implemented in C++ language, without GPU acceleration.

We also developed a graphical user interface using the Qt library to help users recolor videos. As shown in Fig. 3, users can drag a progress bar to preview both the input and recolored video sequences. Concurrently, the color palette of the current frame is displayed below, allowing users to make edits to specific frames to recolor the input video as desired.

A. Evaluation

1) Parameter ε: We first evaluate the threshold ε used to determine the number of Bézier curves (Sec. IV-C1). We provide two examples in Fig. 4 to demonstrate the effectiveness of our parameter settings.

Smaller values of ε typically yield more colors, but might reduce their distinctiveness. In contrast, larger values tend to produce fewer distinct colors, but may ignore representative colors. For example, in the *Smoke* example, using a smaller value ($\varepsilon = 1$) results in similar yellow colors in frame 0 and comparable pink colors in frame 60, while a larger value ($\varepsilon = 10$) neglects the brown colors in frame 0 and frame 30. In the *Sea* example, a smaller ε produces a similar light beige in frame 0, whereas a larger ε fails to capture the light blue.

Overall, our default setting ($\varepsilon = 4$) provides a good trade-off between the distinctiveness and representativeness of the extracted dominant colors. We used the same setting in all experiments and consistently achieved good results.

λ]Parameter λ When refitting the Bézier curve-based video palette (Eq. 8), the parameter λ is used to balance the relative contribution of the edit and the trend losses. Typically, higher values of λ encourage the refitted curves to preserve their original shapes but might ignore user adjustments to local frames. In contrast, lower values adhere more closely to local edits, but may alter the rate of color transitions.

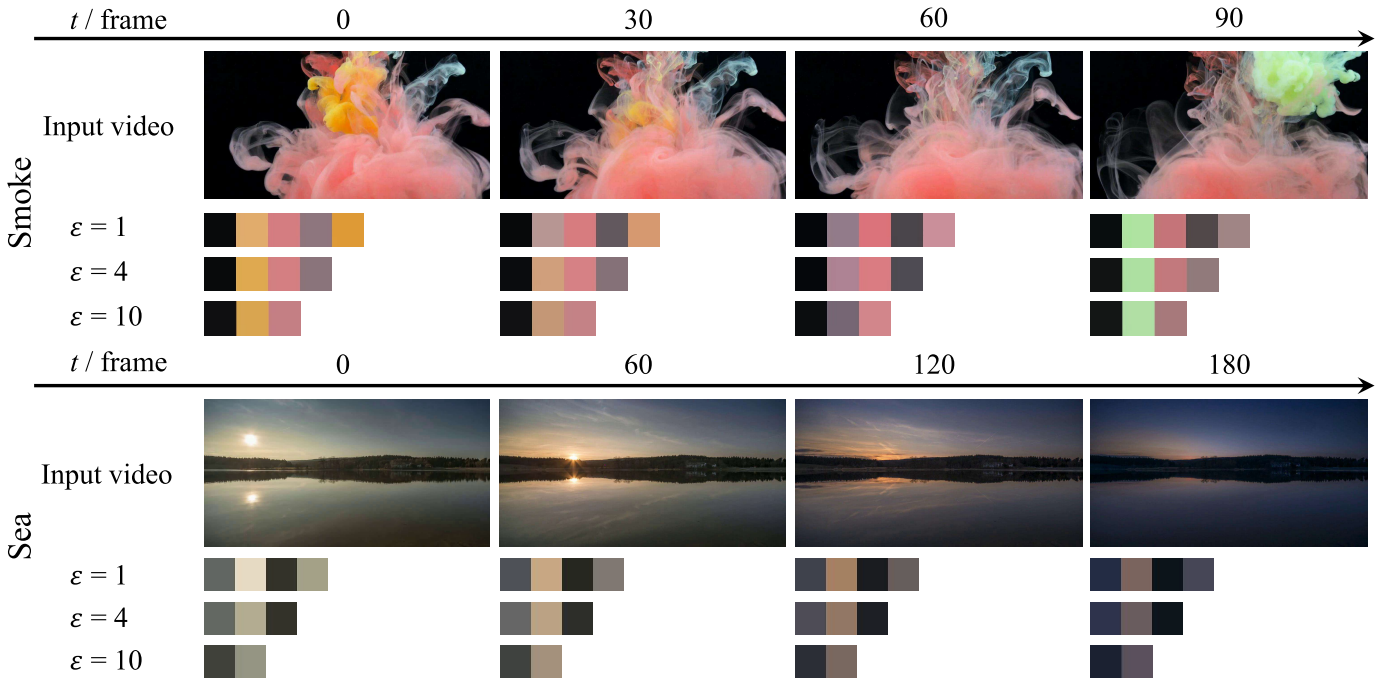


Fig. 4. Frame palettes generated with different values of ε. For each example, from top to bottom: a timeline, the input video sequence, the sliced frame palettes when the threshold ε is set to 1, 4 and 10.

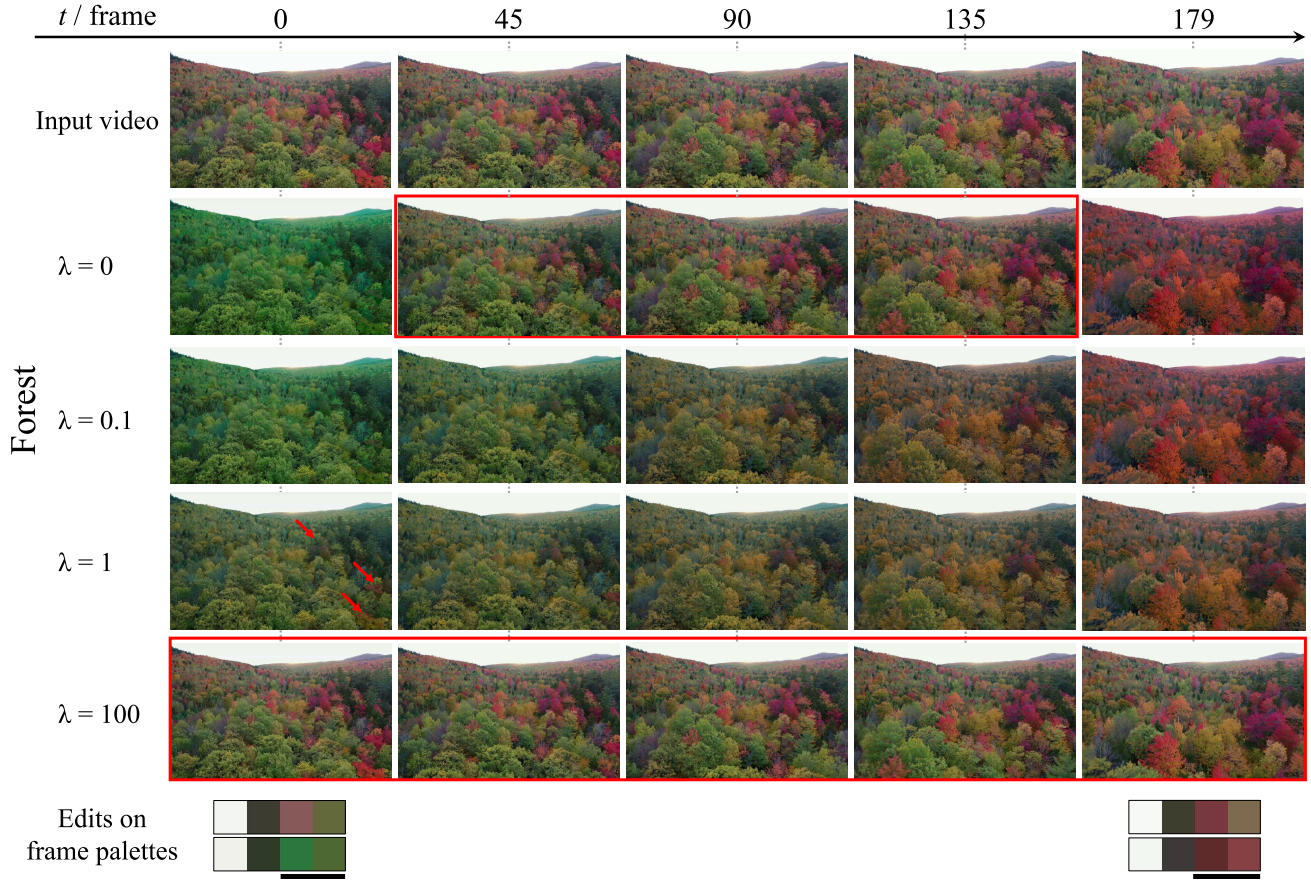


Fig. 5. Video recoloring results with different values of λ. For each example, from top to bottom: a timeline, the input video sequence, the recoloring results for $\lambda = 0, 0.1, 1, 100$, and the corresponding edits applied to the frame palettes.

We test our Bézier curve refitting algorithm with different values of λ in Fig. 5. The input video showcases the exquisite scenery of a forest painted in the rich hues of autumn. Our goal is to create a time-lapse transition from spring to fall.

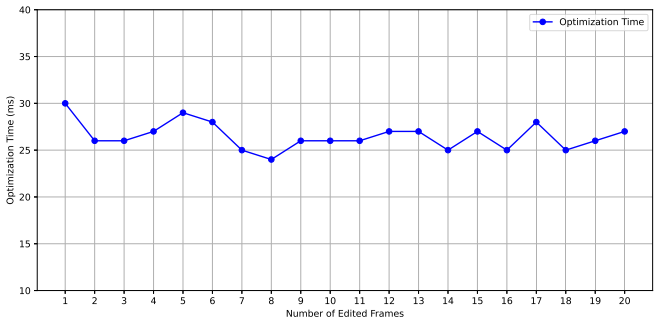


Fig. 6. Optimization times required for editing different numbers of frames.

To achieve this, we change the palette colors of the first frame to green and the last frame to red. With $\lambda = 0$, the colors of the first and last frames change as expected, but a smooth transition across all frames is not achieved (see the three frames in the middle). With $\lambda = 1$ or a higher value like 100, it tends to preserve the original autumn colors but ignores user edits. To make a good trade-off, we set $\lambda = 0.1$ for all examples.

3) Optimization Performance: To evaluate the optimization performance (Eq. 8), we measured the running times for editing 1, 2, up to 20 frames in the *Forest* example, as shown in Fig. 6. The results demonstrate that our optimization is rather efficient, with most execution times ranging between 25 and 30 milliseconds, thus fully supporting real-time video recoloring.

B. Comparisons

To fully evaluate the effectiveness of our algorithm, we provide multiple examples in Figs. 5 and 6 to compare our algorithm with Du et al. [15] in terms of recoloring results and computational efficiency.

In the field of palette-based recoloring, most works focus on static images. Du et al. [15] pioneered the extension of palette-based color editing algorithms to video scenes. They employ a 4D geometry, i.e., a skew polytope, to represent the palette of a video, achieving natural recoloring results. However, this method still has some limitations.

Firstly, their method progressively built the skew polytope, involving numerous iterations in block merging and vertex removal, requiring significant time for video palette generation. Our method employs a simpler geometry, i.e., a set of Bézier curves in RGBT space, to represent the video palette, avoiding numerous iterations and resulting in more efficient palette extraction. The running times for video palette extraction of both methods are presented in Table I. It is evident that our method is over 400 times faster than Du et al.'s method.

Secondly, the palette colors in Du et al.'s method lack representativeness because they are derived from the skew polytope vertices rather than directly from video pixels. Furthermore, the simplification process (block merging and vertex removal) moves the vertices far away from the video pixels, further reducing the representativeness of the palette. This will result in unnecessary global color changes when editing the color palette. We generate the video palette by first performing per-frame clustering and then fitting a set of Bézier curves. This ensures better palette representativeness, enabling more intuitive editing of local object colors. As shown in the *Bird* example of Fig. 7, the user intends to change the bottom of the feeder from dark-cyan to violet. Du et al.'s method causes unexpected global color changes in regions such as the background, a bird's head, and the grain container. Our results better align with the user's editing intent. Similarly, in the *Lake* example of Fig. 8, Du et al.'s method also produces undesired color changes in the hill and the sky.

TABLE I

COMPARISON OF TIME COST BETWEEN DU ET AL. [15] AND OUR METHOD FOR VIDEO PALETTE EXTRACTION. ON AVERAGE, OUR ALGORITHM FOR EXTRACTING VIDEO PALETTE IS MORE THAN 400 TIMES FASTER THAN DU ET AL. [15]

Example	Resolution	#Frame	Du et al. [15] (s)	Ours (s)
Bird	1280×720	120	1,865	**7**
Starry	640×338	120	1,065	**2**
Lake	640×360	181	1,315	**2**
Ink	1280×720	93	3,963	**6**

Lastly, for videos with complex color variations, Du et al.'s method tends to produce merging or splitting vertices during block merging. This hinders the accurate capture of continuous color transitions for specific objects throughout the video. In contrast, our method yields a set of independent yet smooth Bézier curves, making it easier to capture the color variations of certain objects. Consequently, our method offers greater convenience for fine-tuning the gradient effect of such objects. For example, in the *Starry* example of Fig. 7, the user wants to adjust the color of the sky from yellow to blue and then smoothly back to yellow. Du et al.'s approach struggles to remove yellow hues from the sky, while our color palette adeptly captures the nuances of the sky's color shifts, thereby accomplishing a more seamless gradient transition. Similarly, in the *Ink* example of Fig. 8, to produce a similar effect, Du et al.'s method requires more complex editing and produces an unexpected purple during the transition.

In general, our method effectively overcomes the limitations mentioned above. Quantitative and qualitative experimental comparisons demonstrate the efficiency and effectiveness of the proposed method.

C. Results

We generated 10 video recoloring results, shown in Figs. 1, 9 and 10. The examples shown in Fig. 1 and Fig. 9 are time-lapse videos, and Fig. 10 presents some videos with time-varying lighting. Our method consistently generates high-quality recoloring results in all examples. For example, in the *Tower* example, the user successfully adjusted the tone of the night sky by modifying the color palettes of three frames, achieving a smooth gradient effect from light to dark. In the *Sunset* example, the user converted the original sunset scene into a sunrise video by modifying the start and end frame palettes. In the *Seasons* example, the user fine-tuned the palettes to make leaves more tender in spring, richer green

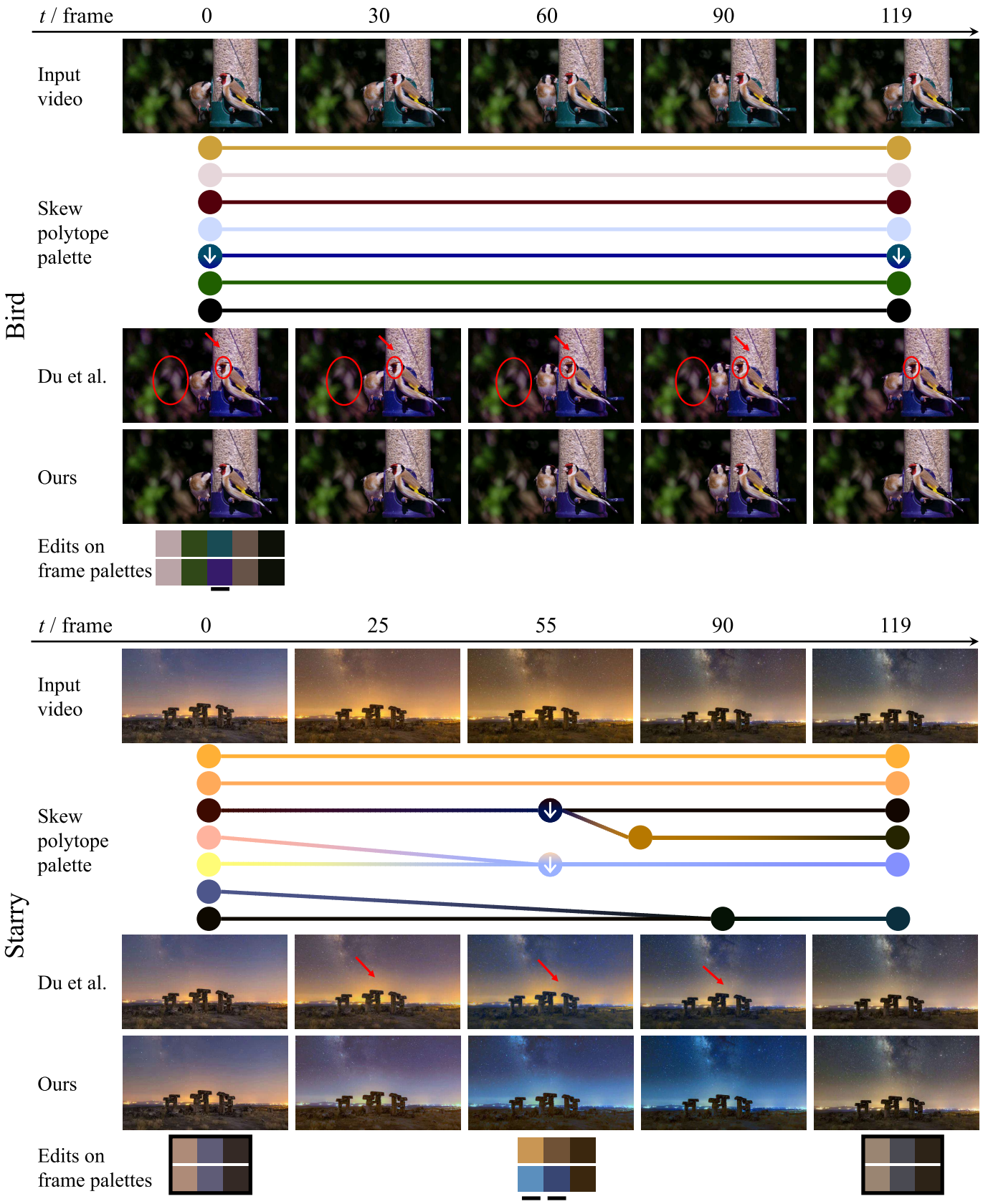


Fig. 7. Comparison of our method with Du et al. [15]. For each example, from top to bottom, we provide the input video, the skew polytope palette generated by Du et al. [15] and edits made on it, the corresponding recoloring results, our recoloring results and edits on frame palettes. Please note the marked areas and arrows to compare the differences of both results.

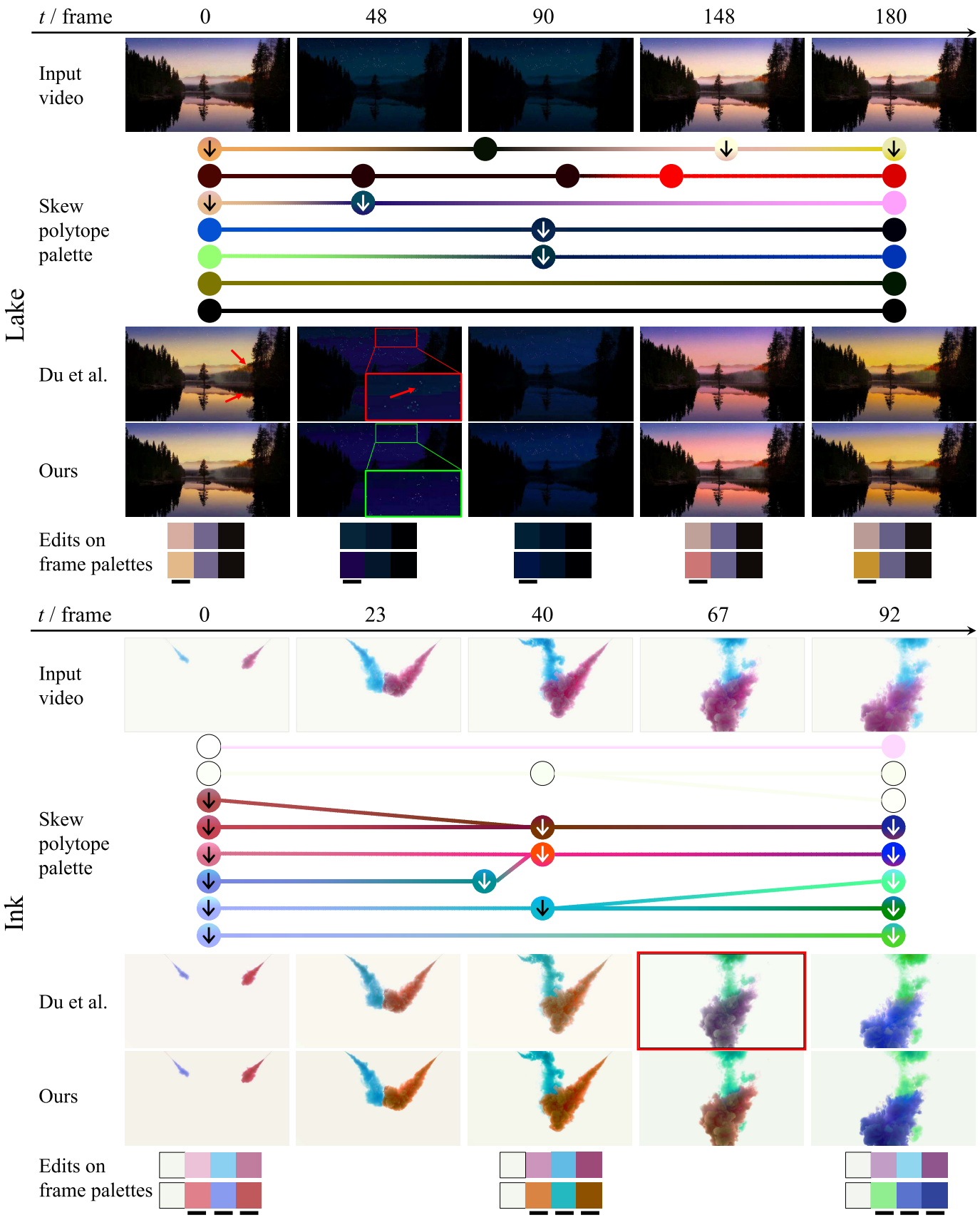


Fig. 8. Comparison of our method with Du et al. [15]. For each example, from top to bottom, we provide the input video, the skew polytope palette generated by Du et al. [15] and edits made on it, the corresponding recoloring results, our recoloring results and edits on frame palettes. Please note the marked areas to compare the differences of both results.

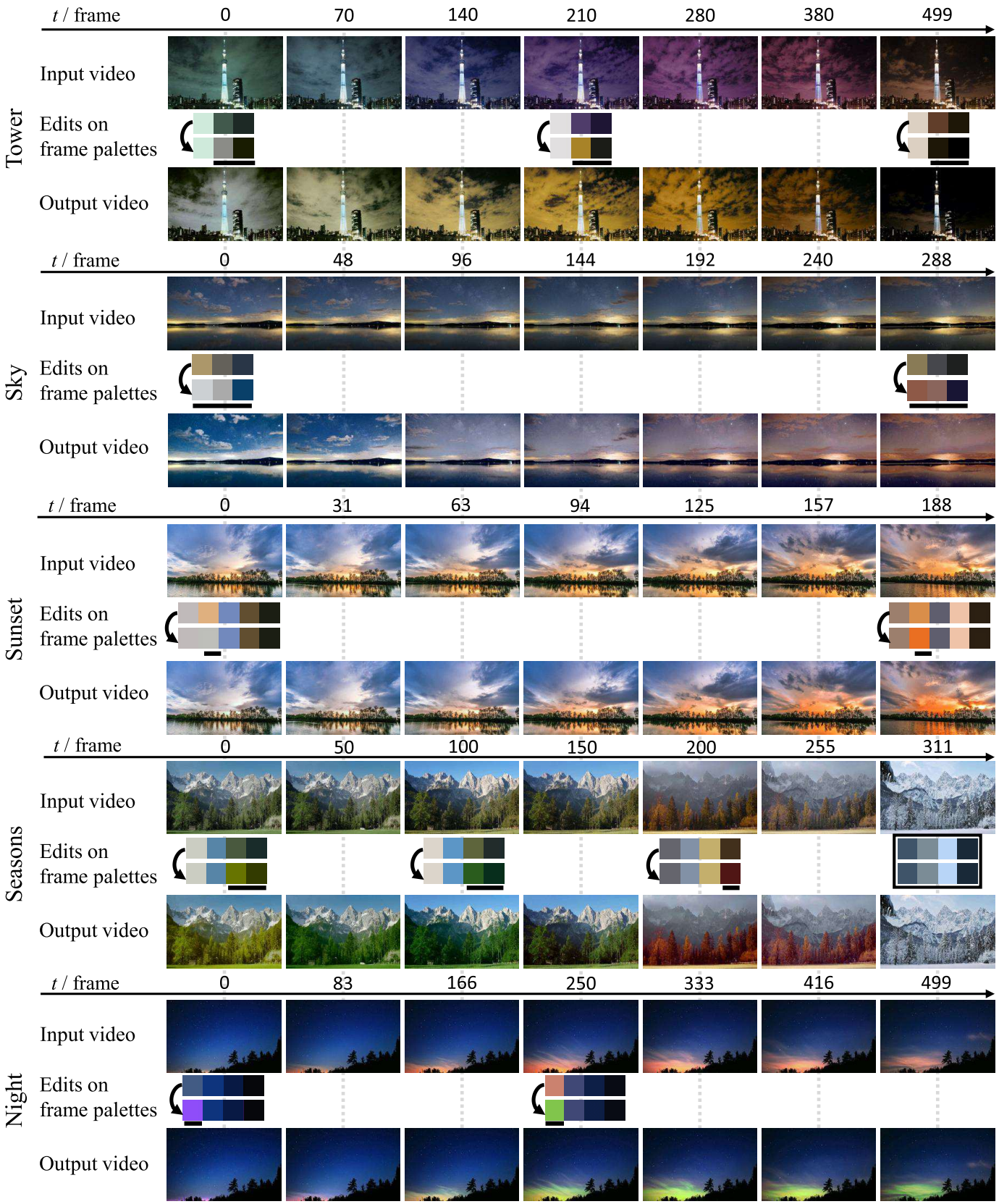


Fig. 9. Results generated by our method. For each example (top to bottom): a timeline, input video, user edits on specific frames (top row: original palettes, bottom row: edited palettes), and recolored video.

in summer, and more vibrant red in autumn, enhancing the seasonal transformation. In the *Man* example, the user changed the color of the light on the face, making the skin tone look more natural and the overall lighting feel softer.

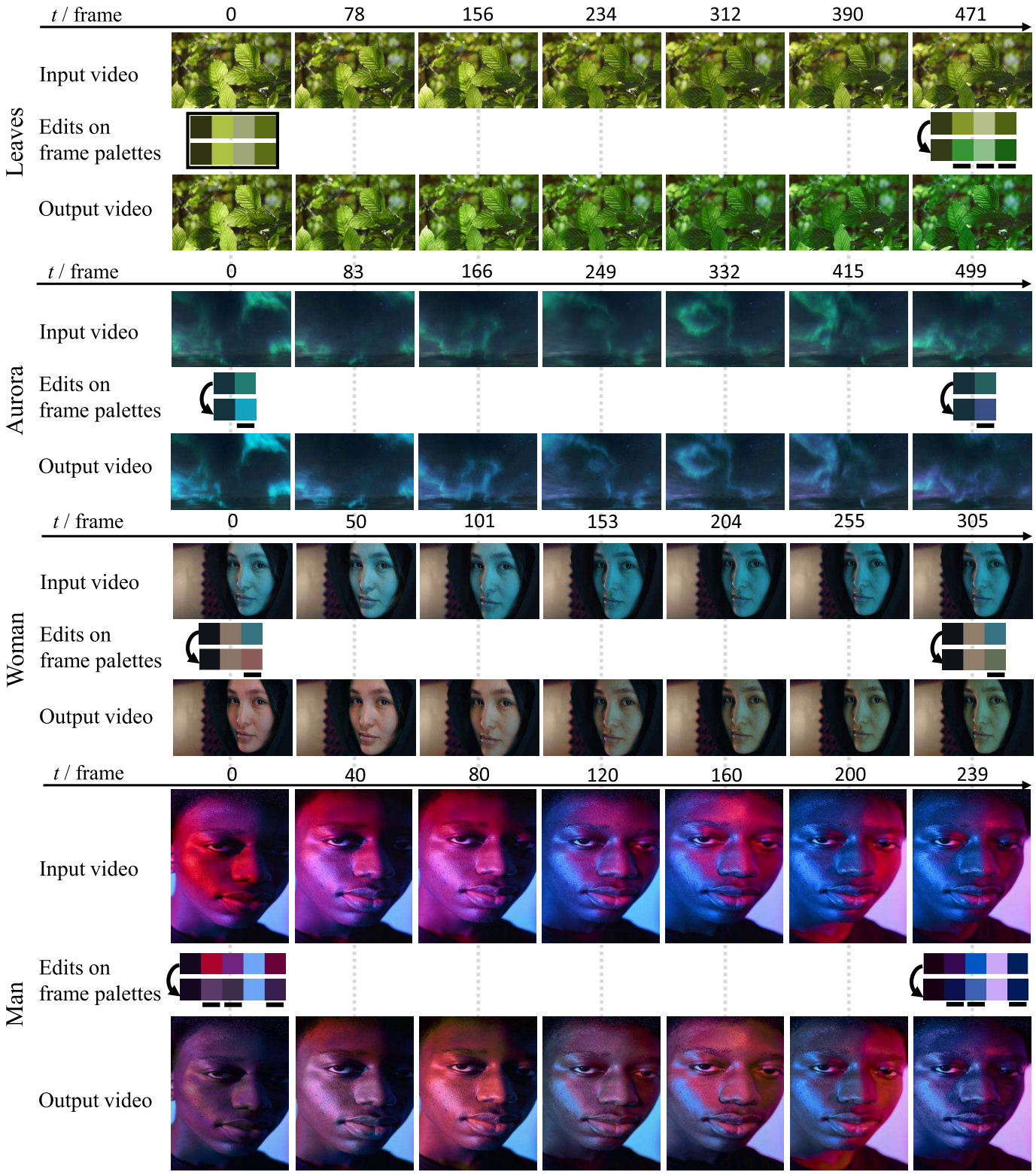


Fig. 10. More results generated by our method. For each example (top to bottom): a timeline, input video, user edits on specific frames (top row: original palettes, bottom row: edited palettes), and recolored video.

Table II presents the running times for palette extraction in all examples, including details on resolution, frame count, and processing time.

In general, our method is efficient and robust for different examples and could produce natural, artifact-free, and smooth time-varying recoloring results.

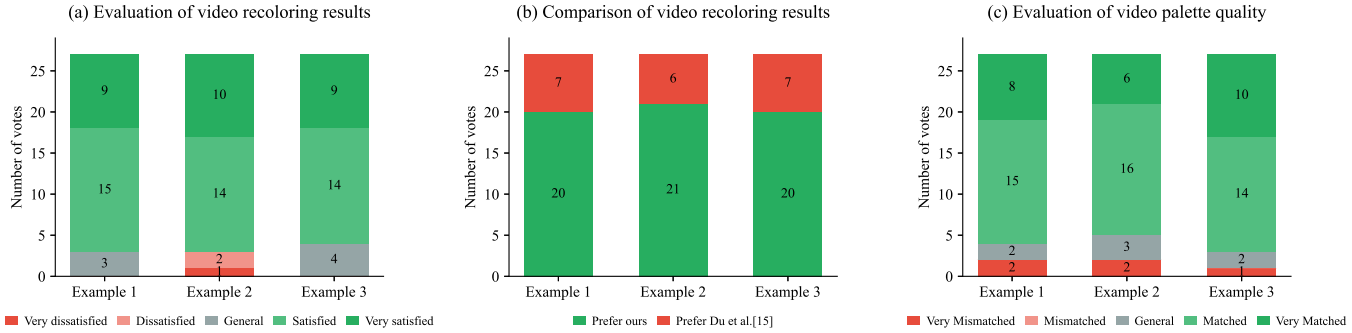


Fig. 11. Statistics of user study. We use stacked bar charts to show the user votes for each task. For Task 1, 71 of 81 votes (87.65%), for Satisfied/Very Satisfied. For Task 2, 61 of 81 votes (75.30%) preferred our method. For Task 3, 69 of 81 votes (85.18%) for Satisfied/Very Satisfied.

TABLE II
RUNNING TIMES REQUIRED FOR VIDEO PALETTE EXTRACTION

Example	Resolution	#Frame	Palette Extract. (s)
Tree (Fig. 1)	1280×1280	500	45
Smoke (Fig. 4)	1920×1080	106	20
Sea (Fig. 4)	1280×720	248	19
Forest (Fig. 5)	1280×720	180	12
Bird (Fig. 7)	1280×720	120	7
Starry (Fig. 7)	640×338	120	2
Lake (Fig. 8)	640×360	181	2
Ink (Fig. 8)	1280×720	93	6
Tower (Fig. 9)	1280×720	500	39
Sky (Fig. 9)	1280×720	289	24
Sunset (Fig. 9)	1920×1080	189	33
Seasons (Fig. 9)	1280×720	312	27
Night (Fig. 9)	1280×720	500	32
Leaves (Fig. 10)	1280×720	472	35
Aurora (Fig. 10)	1280×720	500	36
Woman (Fig. 10)	1280×720	306	23
Man (Fig. 10)	608×1080	240	11

D. User Study

To further validate the effectiveness of our method, we conducted a user study to evaluate our palette extraction and video recoloring results. We invited 27 users aged between 18 and 30 to participate in this experiment. This study contains the following three tasks:

(1) **Task 1: Evaluation of video recoloring results.** We asked each participant to evaluate three examples. For each example, we provided the input video, the edit intention, and the recolored video. We then asked them to rate the results according to the recoloring quality and how well the results align with the intentions. The rating range is divided into five levels, ranging from "very dissatisfied" to "very satisfied".

(2) **Task 2: Comparison of video recoloring results.** We asked each participant to evaluate three examples. For each example, we provided the input video, the edit intention, and the recolored videos produced by Du et al. [15] and our method (these two results are anonymous and are arranged in random order). The participants were then asked to choose the best result.

(3) **Task 3: Evaluation of video palette quality.** We asked each participant to evaluate three examples. For each example, we provided the input video and the frame palettes generated by our method. We then asked them to judge whether the frame palettes capture the dominant color changes in the video. Similarly, the rating scale is divided into 5 levels from "very dissatisfied" to "very satisfied".

Per-task voting results are detailed in Fig. 11. Overall, we received very positive feedback. For Task 1, about 88% of the users rated our recoloring results as satisfied or higher. For Task 2, approximately 75% of the participants preferred the results generated with our method over that of DiVerdi et al. [23]. For Task 3, around 85% of the participants were satisfied or higher with the generated frame palettes, indicating that our approach can successfully capture the main color changes in the input videos.

VI. CONCLUSION AND FUTURE WORK

In this paper, we introduced a novel video recoloring approach that employs a set of Bézier curves to represent the palette of the input video, achieving efficient palette extraction yet natural recoloring. Compared to existing methods, our approach offers three main advantages. 1) We extract video palettes by fitting a set of Bézier curves, which avoids numerous of iterative operations and therefore greatly improves the efficiency of palette extraction. 2) Our video palette exhibits better representativeness, as the data points for fitting the Bézier curves come directly from the video frames. 3) Our approach provides better local control, allowing users to directly edit specific frames to recolor a video.

However, our method still has several limitations that need to be further addressed.

- Our video palette extraction relies solely on color information, without considering semantic features. Thus, it cannot perform content-aware recoloring, that is, distinct objects or regions sharing similar colors cannot be recolored independently.
- Although our palette effectively captures gradual color transitions, it struggles with abrupt changes, particularly rapid color merging and separation events.
- During video palette extraction, all Bézier curves utilize a fixed number of control points, lacking adaptability based on the intensity of color variations over time.

- Since we apply k-means clustering to capture dominant colors per frame, it may fail to capture color variations within smaller or less prominent objects and regions.

To overcome these challenges, we propose the following solutions and aim to refine them in future works.

- Firstly, we plan to extract the video palette in a high-dimensional space that integrates semantic and color information, to enable content-aware video recoloring.
- Secondly, to handle videos with abrupt color changes, we consider using the scene segmentation method (Zhu and Zhou [32]) to divide the video into segments with smooth color transitions. A dedicated sub-palette will be extracted per segment, enabling users to recolor complex sequences by modifying local frame palettes within targeted segments.
- Thirdly, we will develop an improved geometric method to adaptively determine the number of control points for each Bézier curve. This ensures that color sequences with subtle variations require fewer control points, while those with pronounced variations necessitate more points.
- Lastly, to enable the video palette to capture color changes in small or visually subtle regions, we can employ saliency detection or manual interaction to select specific regions, ensuring that their color variations are reflected in the palette.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments and constructive suggestions.

REFERENCES

[1] H. Chang, O. Fried, Y. Liu, S. DiVerdi, and A. Finkelstein, "Palette-based photo recoloring," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 1–139, Jul. 2015.

[2] Q. Zhang, C. Xiao, H. Sun, and F. Tang, "Palette-based image recoloring using color decomposition optimization," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 1952–1964, Apr. 2017.

[3] J. Tan, J.-M. Lien, and Y. Gingold, "Decomposing images into layers via RGB-space geometry," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–14, Jul. 2017.

[4] J. Tan, J. Echevarria, and Y. Gingold, "Efficient palette-based decomposition and recoloring of images via RGBXY-space geometry," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–10, Dec. 2018.

[5] J. Tan, S. DiVerdi, J. Lu, and Y. Gingold, "Pigmento: Pigment-based image analysis and editing," 2017, *arXiv:1707.08323*.

[6] J. Tan, J. Echevarria, and Y. Gingold, "Palette-based image decomposition, harmonization, and color transfer," 2018, *arXiv:1804.01225*.

[7] Y. Wang, Y. Liu, and K. Xu, "An improved geometric approach for Palette-based image decomposition and recoloring," in *Proc. Comput. Graph. Forum*, vol. 38, 2019, pp. 11–22.

[8] M. Grogan and A. Smolic, "Image decomposition using geometric region colour unmixing," in *Proc. Eur. Conf. Vis. Media Prod.*, Dec. 2020, pp. 1–10.

[9] Y. Aksoy, T. O. Aydin, A. Smoli, and M. Pollefeys, "Unmixing-based soft color segmentation for image manipulation," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–19, Jul. 2017.

[10] Q. Zhang, Y. Nie, L. Zhu, C. Xiao, and W.-S. Zheng, "A blind color separation model for faithful palette-based image recoloring," *IEEE Trans. Multimedia*, vol. 24, pp. 1545–1557, 2022.

[11] M.-Y. Cui, Z. Zhu, Y. Yang, and S.-P. Lu, "Towards natural object-based image recoloring," *Comput. Vis. Media*, vol. 8, no. 2, pp. 317–328, Jun. 2022.

[12] C.-K.-T. Chao, J. Klein, J. Tan, J. Echevarria, and Y. Gingold, "ColorfulCurves: Palette-aware lightness control and color editing via sparse optimization," *ACM Trans. Graph.*, vol. 42, no. 4, pp. 1–12, Jul. 2023, doi: 10.1145/3592405.

[13] C.-K.-T. Chao, J. Klein, J. Tan, J. Echevarria, and Y. Gingold, "LoCoPalettes: Local control for Palette-based image editing," *Comput. Graph. Forum*, vol. 42, no. 4, Jul. 2023, doi: 10.1111/cgf.14892.

[14] Q. Sun, Y. Nie, Q. Zhang, and G. Li, "Building coarse to fine convex hulls with auxiliary vertices for palette-based image recoloring," *IEEE Trans. Vis. Comput. Graphics*, vol. 30, no. 8, pp. 5581–5595, Aug. 2024.

[15] Z.-J. Du, K.-X. Lei, K. Xu, J. Tan, and Y. Gingold, "Video recoloring via spatial–temporal geometric palettes," *ACM Trans. Graph.*, vol. 40, no. 4, pp. 1–16, Aug. 2021.

[16] H. Prautzsch, W. Boehm, and M. Paluszny, *Bézier and B-Spline Techniques*, vol. 6. Cham, Switzerland: Springer, 2002.

[17] B. J. Meier, A. M. Spalter, and D. B. Karelitz, "Interactive color palette tools," *IEEE Comput. Graph. Appl.*, vol. 24, no. 3, pp. 64–72, May 2004.

[18] J. Delon, A. Desolneux, J. L. Lisani, and A. B. Petro, "Automatic color palette," in *Proc. IEEE Int. Conf. Image Process.*, Sep. 2005, pp. II–706.

[19] P. O'Donovan, A. Agarwala, and A. Hertzmann, "Color compatibility from large datasets," in *Proc. ACM SIGGRAPH papers*, Jul. 2011, pp. 1–12.

[20] S. Lin and P. Hanrahan, "Modeling how people extract color themes from images," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, Apr. 2013, pp. 3101–3110.

[21] Y. Cao, A. B. Chan, and R. W. H. Lau, "Mining probabilistic color palettes for summarizing color use in artwork collections," in *Proc. SIGGRAPH Asia Symp. Visualizat.*, Nov. 2017, pp. 1–8.

[22] M. Shugrina, J. Lu, and S. Diverdi, "Playful palette: An interactive parametric color mixer for artists," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–10, Aug. 2017.

[23] S. DiVerdi, J. Lu, J. Echevarria, and M. Shugrina, "Generating playful palettes from images," *Expressive*, vol. 19, pp. 69–78, May 2019.

[24] J. Cho, S. Yun, K. Lee, and J. Y. Choi, "PaletteNet: Image recolorization with given color palette," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1058–1066.

[25] N. Akimoto, H. Zhu, Y. Jin, and Y. Aoki, "Fast soft color segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8274–8283.

[26] Z. Kuang, F. Luan, S. Bi, Z. Shu, G. Wetzstein, and K. Sunkavalli, "PaletteNeRF: Palette-based appearance editing of neural radiance fields," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 20691–20700.

[27] Q. Wu, J. Tan, and K. Xu, "PaletteNeRF: Palette-based color editing for NeRFs," 2022, *arXiv:2212.12871*.

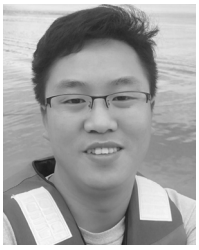
[28] T. Ju, S. Schaefer, and J. Warren, "Mean value coordinates for closed triangular meshes," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 561–566, Jul. 2005.

[29] G. M. Phillips and G. M. Phillips, "Bernstein polynomials," in *Interpolation Approximation by Polynomials*. New York, NY, USA: Springer, 2003, pp. 247–290.

[30] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

[31] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2007, pp. 1027–1035.

[32] Y. Zhu and D. Zhou, "Scene change detection based on audio and video content analysis," in *Proc. 5th Int. Conf. Comput. Intell. Multimedia Appl. (ICCIMA)*, Sep. 2003, pp. 229–234.



Zheng-Jun Du (Member, IEEE) received the Ph.D. degree in computer science from Tsinghua University in 2023. He is an Associate Professor with the School of Computer Technology and Application, Qinghai University. His research interests include computer graphics, image and video processing, computer vision, and dynamic SLAM.

Jia-Wei Zhou received the master's degree from Qinghai University in 2024. His research interests include computer graphics, image and video processing, and computer vision.

Kang Li is currently pursuing the bachelor's degree with Qinghai University. His research interests include computer graphics, image and video processing, and computer vision.

Jian-Yu Hao is currently pursuing the bachelor's degree with Qinghai University. His research interests include computer graphics, image and video processing, and computer vision.

Zi-Kang Huang received the bachelor's degree from Qinghai University in 2023. He is currently pursuing the master's degree at Tianjin University. His research interests include computer graphics, image and video processing, and computer vision.

Kun Xu received the bachelor's and Ph.D. degrees from the Department of Computer Science and Technology, Tsinghua University, in 2005 and in 2009, respectively. He is an Associate Professor with the Department of Computer Science and Technology, Tsinghua University. His research interests include real-time rendering, image/video editing, and 3D scene synthesis.