Computers & Graphics 119 (2024) 103875

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Edit propagation via color palettes[☆]

Zi-Xun Xia [a], Jian-Yu Hao [a], Kang Li [a], Ao-Xiang Tian [a], Zheng-Jun Du [a,b,*]

[a] *Department of Computer Technology and Application, Qinghai University, Xining, 810016, China*
[b] *Qinghai Provincial Key Laboratory of Media Integration Technology and Communication, Xining, 810016, China*

ARTICLE INFO

Keywords:
Edit propagation
Color palette
Optimization
Color editing

ABSTRACT

The ability to ease operation and real-time feedback to image editing has attracted increasing attention in our everyday lives. Existing edit propagation approaches typically formulate the color editing task as a quadratic energy optimization problem, which is computationally expensive and requires a lot of user interactions. To address this problem, in this paper, we propose a novel edit propagation method based on color palette. We first extract a color palette from the input image, and then calculate the mixing weights of the image pixels with respect to the extracted palette. Finally, we solve for an edited palette according to user edits, to propagate local edits to the whole image. Our main contributions include formulating the edit propagation as a palette-based optimization problem that can be solved efficiently, and requiring only fewer user interactions than existing methods. We have demonstrated our method for color editing on a wide range of examples. Compared to existing methods, our approach is more efficient and friendly for novice users due to its ease of interaction.

1. Introduction

Color editing is an active research topic in the fields of image and video processing, computer vision, and computer graphics, which has been widely used in film production, advertising design, digital art creation, etc. Edit propagation is a well-known technique in color editing, which provides the user with an intuitive interface and automatically propagates sparse user edits to the whole image. After over a decade of development, edit propagation has made significant progress in color editing.

Traditional approaches [1–6] typically formulate edit propagation as a quadratic energy optimization problem. It needs to solve a large-scale linear system. It is time-consuming and storage costly even for a moderate-sized input image, which limits the practical use for large-size input. Alternatively, several recent methods [7–9] speed up edit propagation with interpolation or data-drive learning methods. While it reduces the processing time significantly, it requires the user to provide more fine-tuned strokes to achieve desirable results, which is not friendly for novice users to learn and use.

To address these problems in edit propagation, we present in this paper a palette-based edit propagation method that effectively propagates user edits throughout the image and requires less interaction.

Palette-based color editing [10–15] is another well-studied technique in color editing. In this field, a color palette is typically defined as a small set of colors that express the main color distribution of an image. With the extracted color palette, pixels in the input image can be efficiently encoded as a linear combination with respect to the color palette. In color editing, the user adjusts the color of the input image by modifying the color palette. These methods have become increasingly popular in recent years due to their computational efficiency and ease of use. However, existing palette-based methods still have a primary limitation: it does not support direct editing of the image. Considering both salience and simplification, the extracted palette typically contains fewer colors. As a result, if one object's color is not present in the palette, the user has to seek and modify several palette colors that could affect that object, to recolor the image.

The intuitive nature of edit propagation, together with the computational efficiency of the palette-based approach, inspired us to propose a novel palette-based edit propagation method. We follow Tan et al. [11] to employ a simplified convex hull in RGB space to present the color palette of a given image. Then, we calculate the mixing weights of each pixel to the extracted palette by mean value coordinates interpolation, so that each pixel can be naturally expressed as a convex combination of the palette colors. Instead of modifying the color palette to recolor the input image in most existing methods, we allow the user to put sparse edits on pixels directly, and then propagate it to the whole image. We formulate the edit propagation as a simple optimization problem that can be solved efficiently. A wide range of examples have demonstrated the effectiveness of our method.

The main contributions of this paper can be summarized as follows:

[☆] This article was recommended for publication by H. Fu.
[*] Corresponding author at: Department of Computer Technology and Application, Qinghai University, Xining, 810016, China.
E-mail address: dzj@qhu.edu.cn (Z.-J. Du).

https://doi.org/10.1016/j.cag.2024.01.002
Received 26 September 2023; Received in revised form 5 January 2024; Accepted 5 January 2024
Available online 8 January 2024

- We present a palette-based edit propagation that enables direct user interaction with images for color editing, which can be solved more efficiently and requires less interaction than previous methods.
- We develop a graphical user interface (GUI) for color editing, enabling non-expert users to easily engage in real-time image editing without extensive learning or training.

The rest of the paper is organized as follows: First, we give a brief introduction to related works. Second, we provide a background of the proposed method. Next, we describe the details of the proposed algorithm. Then, we provide extensive experiments to illustrate the effectiveness of our approach. At last, we conclude and discuss the limitations of our paper.

2. Related works

2.1. Edit propagation

Propagating user-sparse edits to the whole image is a well-known technique with a wide range of applications, such as interactive color editing, colorization, material editing, tone mapping, etc. Traditional approaches automatically propagate user edits to the rest of the image, guided by the rule that similar pixels should receive similar edits. The pioneering work of edit propagation was proposed by Levin et al. [1] for grayscale image colorization. It propagates the colors of user strokes to adjacent pixels with the assumption that pixels of similar lightness should have similar colors. Later, Lischinski et al. [16] extended the idea of edit propagation for tonal adjustment. Pellacini et al. [2] proposed a novel approach to edit spatially-temporally varying measured materials through an optimization that enforces similar edits are applied to areas with a similar appearance. An et al. [3] first introduced edit propagation into interactive color editing. They measured the affinities in all pixel pairs to enable long-range propagation. While powerful, all these methods are computational and storage costly. Xu et al. [5] applied the K-D tree in edit propagation for acceleration to reduce time and memory costs. This approach speeds up the algorithm significantly, but real-time feedback was still challenging. Li et al. [7] formulated edit propagation as an interpolation problem in high-dimensional affinity space, and achieved real-time color editing. Alternatively, Bie et al. [17], Xiao et al. [18], and Li et al. [9] adopted clustering and sampling-like methods to reduce the computational costs of previous approaches further.

With the advance of deep learning, recently some approaches based on neural networks were proposed. Endo et al. [19] proposed the first deep learning-based edit propagation, "DeepProp", that extracts high-dimensional features with a convolution neural network (CNN) for edit propagation. Gui et al. [20] formulated edit propagation as a multi-class classification problem and employed a fully convolution network that can be trained end-to-end, to extract the visual and spatial features and predict the resulting image. Although these methods could achieve desirable results, they are sensitive to user edits, and cannot generate resulting images in real-time.

2.2. Palette-based image recoloring

Palette-based image recoloring is a recent paradigm for color editing. The first work was proposed by Chang et al. [10]. They employed a modified K-means clustering to extract the palette, and then transfer the color of the changed palette to the whole image by using the radial basis function (RBF) interpolation. Zhang et al. [21] generalizes this method with a decomposition optimization approach and shows a wide range of applications such as color transfer and image segmentation, etc. Tan et al. [11] proposed a geometric method for palette extraction. They compute a simplified convex hull of an image in RGB space and use its vertices as the color palette. Color editing is then achieved by calculating the mixing weights of each pixel to the extracted palette. Tan et al. [13] extended Tan et al. [11] with a simple and efficient method where they extract the color palette and compute the corresponding mixing weights in RGBXY space for better spatial consistency. However, the simplified convex hull obtained using the above methods often has vertices far from existing colors, resulting in poor color representation. To address this issue, Wang et al. [15] presented an optimization approach to improve the representativeness of the palette, and employ mean value coordinates (MVC) [22] interpolation to achieve efficient and smooth recoloring. Alternatively, Sun et al. [23] presented a novel coarse-to-fine convex hull construction technique involving auxiliary vertices. Initially, they create a coarse convex hull with direct image pixels, resulting in a compact shape yet not encompassing all pixels. Then, they integrate auxiliary vertices into the coarse hull, generating a fine convex hull that includes more image pixels. This approach ensures higher reconstruction accuracy and better representativeness in recoloring.

Recently, Du et al. [24] further extended this method to video editing by computing a color palette in RGBT space. It achieves natural and smooth recoloring over time. In addition, Aksoy et al. [12] treated each palette color as a Gaussian distribution, and decomposed the input into a set of layers for recoloring. Zhang et al. [25] proposed a novel blind color separation model to extract the color palette and calculate the mixing weights simultaneously. Chao et al. [26] introduced the "ColorfulCurve" for palette-aware lightness control and image space color editing. It extracts a hue-chroma palette and builds palette-based tone curves, allowing sparse, per-palette-color control of lightness across the image. Their method supports direct pixel color editing especially when the colors to be edited do not appear in the palette. Their approach also supports palette editing, and typically the palette size is limited for ease editing, making it difficult to ensure the sparsity of pixels' mixing weights. Although they devised a sparsity term to restrict more palette colors from changing, a single palette color can still influence extensive regions, leading to undesirable global color changes in recoloring. Our approach is similar to Chao et al. [26] in interaction, the main difference is that our method allows the palette to contain more colors while adding additional correlation constraints in the optimization process of the edited palette, thereby achieving better local control during recoloring.

Deep learning-based methods have made advancements in the field of palette-based image recoloring. For instance, Cho et al. [27] introduced "PaletteNet", which adopts a content-aware approach to adjust the color of an input image matching a user-defined palette. However, this algorithm only supports a fixed-size color palette. Akimoto et al. [28] leveraged a U-Net to estimate the mixing weights of a given image and the corresponding color palette. However, this approach often produces blending weights, and the mixing weights are not sparse enough, resulting in less localized color editing. Chao et al. [29] proposed a semantic segmentation-based method for image recoloring, which involves complex user interactions and heavily depends on the semantic segmentation model, often lacking sufficient accuracy. In contrast, our method primarily aims to enhance the interaction efficiency of classic edit propagation methods without taking semantic information into account, thereby making it more efficient and user-friendly.

Our work builds on top of Wang et al. [15] to extract the color palette of a given image. Unlike most existing approaches, we view the palette as a hidden tool, providing a more intuitive interface that allows users to recolor the input image by directly modifying the pixels. Besides, our method is more efficient and requires less interaction than existing approaches.

3. Background and motivation

Our method builds on convex hull-based image recoloring. Before introducing our approach, we begin with some relevant background in this section. It consists of two main aspects i.e., palette extraction and mixing weights calculation.

3.1. Palette extraction

Palette extraction aims to abstract a small set of representative colors from a given input image. Inspired by the traditional painting process, Tan et al. [11] argue that the colors in an image often result from blending multiple colors in a palette. Following this idea, they proposed the first convex hull-based approach for color editing. Specifically, they first project the input image into 3D RGB space, then compute the convex hull and simplify it. Finally, the vertices of the simplified convex hull act as the color palette. However, the palette colors (convex hull vertices) typically are far from the existing colors in the image due to the nature of the convex hull, making the color palette less representative. To address this problem, Wang et al. [15] proposed an optimized algorithm to enhance the representativeness of the color palette.

Given an input image I, Wang et al. [15] aim to find a polyhedral $P = (V, F)$ (V and F are sets of vertices and faces, respectively) to minimize the following energy function

$$E = \theta \cdot R(P, I) + S(V, I) \tag{1}$$

Which combines the reconstruction loss $R(\cdot)$ and the representation loss $S(\cdot)$, weighted by a balance parameter θ. We empirically set $\theta = 100$ for all of our examples.

The reconstruction loss is measured by the average distance of all pixels to the convex hull of V

$$R(P, I) = \frac{1}{N} \sum_i \left\| I_i - \mathrm{Proj}(I_i) \right\|_2 \tag{2}$$

Where N denotes the number of pixels in I, $\mathrm{Proj}(I_i)$ denotes the projection of I_i to the polyhedral P. We let $\mathrm{Proj}(I_i) = I_i$ if I_i is enclosed in P, because such pixel can be accurately reconstructed according to the property of a convex hull.

The representativeness loss is measured by the average distance of the palette colors V to the centers of its neighbors $\mathcal{N}$.

$$S(V, I) = \frac{1}{|V|} \sum_i \left\| V_i - \mathcal{N}_i \right\| \tag{3}$$

Where $\mathcal{N}_i$ is the center of V_i's neighbors. V_i's neighbors are defined as the k nearest pixels to V_i in RGB space.

In optimization, the polyhedral G is initialized as the simplified convex hull calculated using Tan et al. [11]. The resulting color palette is obtained by iteratively optimizing the energy function (Eq. (1)).

3.2. Mixing weights calculation

After obtaining the color palette of an image, Wang et al. [15] employ mean value coordinates (MVC) to calculate the mixing weights of pixels. That is to say, any pixel I_i is expressed as a convex combination of the color palette P

$$I_i = \sum_{j=1}^{M} w_{ij} P_j \tag{4}$$

Here M is the palette size, w_{ij} is referred to as the mixing weight of I_i respect to P_j, and it satisfies that $\sum_j w_{ij} = 1$ and $0 \le w_{ij} \le 1$. once the color palette along with the mixing weights is obtained, user could recolor the image by modifying the color palette, i.e. $I'_i = \sum_{j=1}^{M} w_{ij} P'_j$.

Motivation. The color palette is a powerful tool for color editing. In convex hull-based approaches [11,13,15], pixels can be naturally and efficiently encoded as a convex combination of the color palette by mixing weights calculation. Thanks to this expression, the user could recolor an image by modifying its corresponding color palette. However, it does not support direct editing of the input image. As a result, recoloring some objects or regions can be tedious, especially if the colors of those objects or regions do not appear in the color palette. In edit propagation, while it allows the user to make edits directly, and propagates them to the whole image, it is always time and space consuming, or requires the user to provide intensive interactions. Considering the advantages of palette-based approaches and the disadvantages of edit propagation, we present a novel palette-based approach for edit propagation. It enables the user to put edits to the image directly, and then efficiently propagates user edits to the whole image.

4. Method

Our goal is to propagate user edits to the whole image quickly. To this end, we first follow Tan et al. [11] and Wang et al. [15] to extract a color palette from the input image, and calculate the mixing weights of each pixel with respect to the color palette. We then formulate the propagation of user edits as an efficient optimization problem, to achieve intuitive and real-time color editing. Since Section 3 has detailed the palette extraction and mixing weights calculation, next, we focus on the proposed edit propagation algorithm.

4.1. Formulation

Given an input image $I \in \mathbb{R}^{N\times 3}$ (where N denotes the total number of pixels in the image), an extracted color palette $P \in \mathbb{R}^{M\times 3}$ (where M denotes the palette size), along with the corresponding mixing weights $W \in \mathbb{R}^{N\times M}$. Let $D = \{d_i\}$ be the indices of user modified pixels, $C = \{C_{d_i}\}$ and $C' = \{C'_{d_i}\}$ are the original and modified pixels, respectively. Our goal is to solve for the edited image I' that satisfies user intention. To this end, we formulate this task as an optimization problem: seeking a palette P' that is minimally changed from the original palette P. Typically, similar pixels have similar mixing weights, so that user local edits can be efficiently and naturally propagated to similar pixels in the rest of the input image with $I' = WP'$. The pipeline of this method is illustrated in Fig. 1.

To seek a suitable palette P', we define a loss function to measure its quality. A well-designed loss function should take into account the editing intention, locality and quality of the edited image. Therefore our loss function is defined as the weighted sum of an editing term L_{edit}, a correlation term L_{corr} and a sparsity term L_{sparse}, and our goal in searching for a palette P' is formulated as:

$$P' = \arg\min_{P'} L \quad \text{and} \quad L = \lambda_1 L_{\text{edit}} + \lambda_2 L_{\text{corr}} + \lambda_3 L_{\text{sparse}} \tag{5}$$

Where λ_1, λ_2 and λ_3 are the balancing parameters that control the relative contribution of these terms. We empirically set $\lambda_1 = 8$, $\lambda_2 = 4$ and $\lambda_3 = 1$.

Edit term. The edit term forces user-specific pixels to be accurately recolored to target colors with the modified color palette P'. We enumerate all user-specific pixels to measure the editing accuracy. It is defined as the average L_2 difference between the recolored and original user-specific pixels:

$$L_{\text{edit}} = \frac{1}{|D|} \sum_{i\in D} \left\| W_i P' - C'_i \right\|_2^2 \tag{6}$$

Where $W_i \in \mathbb{R}^{1\times M}$ denotes the mixing weights of any user-specific pixel C_i, and the recoloring result of C_i is calculated by $W_i P'$.

Correlation term. To meet user edit intention as much as possible, it is also essential to maintain the colors of pixels that are not similar to the specified ones. Its goal is to avoid unnecessary color change in the color propagation process. We calculate the pixel-wise L_2 difference between the recolored result and the input image to measure the correlation loss. It is defined as:

$$L_{\text{corr}} = \frac{1}{\sum_{i=1}^{N} \omega_i} \sum_{i=1}^{N} \omega_i \left\| I_i - W_i P' \right\|_2^2 \tag{7}$$

Where ω_i is a weight parameter, its purpose is to keep the colors of pixels that are not similar to user-specified pixels as unchanged as possible. So that pixels are similar (dissimilar) to those user-specified

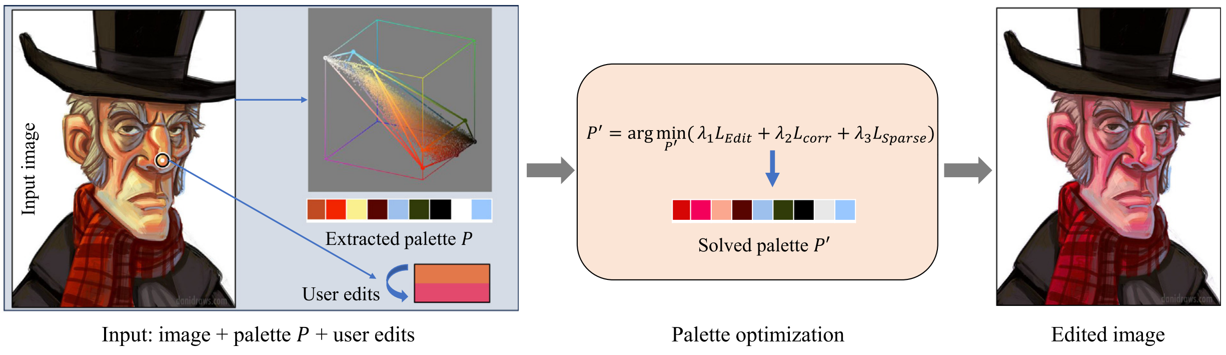

Input image
Extracted palette P
User edits
Input: image + palette P + user edits
$P' = \arg\min_{P'}(\lambda_1 L_{Edit} + \lambda_2 L_{corr} + \lambda_3 L_{Sparse})$
Solved palette P'
Palette optimization
Edited image

Fig. 1. Pipeline of our method. Our method takes the image I, the extracted palette P, and the user edits C' as input, and then seeks a color palette P' by minimizing an energy function to match the user editing intention, at last, outputs the edited image with $I' = WP'$. Image @Dani Jones.

pixels would receive smaller (larger) weights. So the weight ω_i of any pixel I_i is determined by the minimal distance between I_i and user-specific pixels C, it is defined as:

$$\omega_i = \exp(\min_{j\in D} \left\| I_i - C_j \right\|_2^2) \tag{8}$$

Sparsity term. Locality is an essential criterion in color editing. It facilitates the user to achieve targeted editing without undesired global color changes. To this end, we seek a color palette P' that has minimal change against the original palette P according to user edits. That is to say, we expect the variation between P and P' to be as sparse as possible. Specifically, the sparsity term is defined as:

$$L_{\text{sparse}} = \frac{1}{M}\sum_{i=1}^{M} \| P_i' - P_i \|_1 \tag{9}$$

4.2. Optimization

Directly solving the edit propagation problem in Eq. (5) is difficult. Two primary issues need to be resolved. Firstly, we calculate the correlation term (Eq. (7)) over all pixels in the input image, which is fairly time-consuming even for moderate-sized input images, and cannot achieve real-time feedback. Secondly, for better locality, we formulate the sparsity term as L_1 difference between changed palette P' and original palette P. Therefore, the loss function is not differentiable everywhere; it is hard to obtain stable optimal solutions, and it is easy to fall into local optima.

To improve computational efficiency, we evenly sample $\mathcal{S}$ (we set $\mathcal{S} = 256$ in default) pixels in the input image for correlation term calculation. To achieve stable palette P', we try to remove the L_1 difference from the sparsity term (Eq. (9)) and convert it into a common linear term. Specifically, let

$$\begin{cases} X_{i,j} = \dfrac{\left\| P_{i,j}' - P_{i,j} \right\|_1 + \left(P_{i,j}' - P_{i,j} \right)}{2} \\ Y_{i,j} = \dfrac{\left\| P_{i,j}' - P_{i,j} \right\|_1 - \left(P_{i,j}' - P_{i,j} \right)}{2} \end{cases} \tag{10}$$

Where $j \in \{1,2,3\}$ is used to traverse all three RGB channels of P_i and P_i' (1: Red, 2: Green, 3: Blue). Let $X_i = (X_{i,1}, X_{i,2}, X_{i,3})$, $Y_i = (Y_{i,1}, Y_{i,2}, Y_{i,3})$, then $P_i' = X_i - Y_i + P_i$ and $\|P_i' - P_i\|_1 = \sum_{j=1}^{3} X_{i,j} + Y_{i_j}$. We then substitute which into Eqs. (6), (7) and (9). As a result, these three terms can be rewritten as:

$$\begin{cases} L_{\text{edit}}' = \dfrac{1}{|D|}\sum_{i\in D} \|W_i(X+Y+P) - C_i'\|_2^2 \\ L_{\text{corr}}' = \dfrac{1}{\sum_{i=1}^{N}\omega_i}\sum_{i=1}^{N}\omega_i \|I_i - W_i(X+Y+P)\|_2^2 \\ L_{\text{sparse}}' = \dfrac{1}{M}\sum_{i=1}^{M}\sum_{j=1}^{3} X_{i,j} + Y_{i,j} \end{cases} \tag{11}$$

Where $(X - Y + P) \in \mathbb{R}^{M\times 3}$, its ith column is $X_i - Y_i + P_i$. So Eq. (5) can be further rewritten as:

$$X, Y = \arg\min_{X,Y} L' \quad \text{and} \quad L' = \lambda_1 L_{\text{edit}}' + \lambda_2 L_{\text{corr}}' + \lambda_3 L_{\text{sparse}}' \tag{12}$$

We employ a high-performance quadratic programming solver, Gurobi [30], to solve for X and Y. We then obtain the resulting edited palette P' with $P' = X - Y + P$.

The time complexity of the optimization process using Gurobi is approximate $\mathcal{O}(|P|(|S| + |D|)n)$. Here, $|P|$ denotes the palette size, $|S|$ denotes the number of sampling pixels for calculating the correlation term, $|D|$ denotes the total number of user-selected pixels (which is typically small), and n represents the number of iterations required for convergence. Typically, it can be converged in around 10 iterations and meets the real-time requirements.

After obtaining the changed palette P', we then propagate the color variations in P' to the whole image with $I' = WP'$. Generally speaking, our method is highly efficient and could meet user editing intentions without introducing undesirable global color changes in color editing.

5. Experiments

We perform all experiments on a desktop computer with an Intel i9-9900K 3.6 GHz CPU and 8 GB RAM, running Ubuntu 22.04 LTS as the operating system. Our algorithm is implemented in C++17 standard. We employ the source codes provided by Tan et al. [13] and Wang et al. [15] to generate the color palettes of input images.

We also developed a tool for edit propagation. As shown in Fig. 3, we place a color editing interface on the left, and display the input and edited images on the right. Our tool enables users to conveniently select pixels and pick colors via color wheel, RGB space, HSV space, etc. It is easy to use, supports quick feedback, and allows users to observe editing results instantly.

In this section, we first compare our approach with state-of-the-art palette-based, edit propagation -based and pixel-level image recoloring methods, to demonstrate the effectiveness of our approach. Then, we provide an ablation study to evaluate the effectiveness of three loss terms in the energy function (Eq. (5)). At last, we perform a user study to evaluate our method objectively.

5.1. Parameter evaluation

In Fig. 2, we evaluate the impact of the correlation term (Eq. (7)) on recoloring. We sample 16, 64, 256 and 1024 pixels to calculate the correlation term, respectively. The recoloring results along with the average processing time are presented in column 2, 3, 4 and 5, respectively. Generally, a small number of samplers struggles to keep the color of unrelated parts unchanged, and a larger number of samplers leads to better locality in recoloring. In the *Beach* example, the user wants to change the colors of the sky and the sea from blue to yellow. The cloud

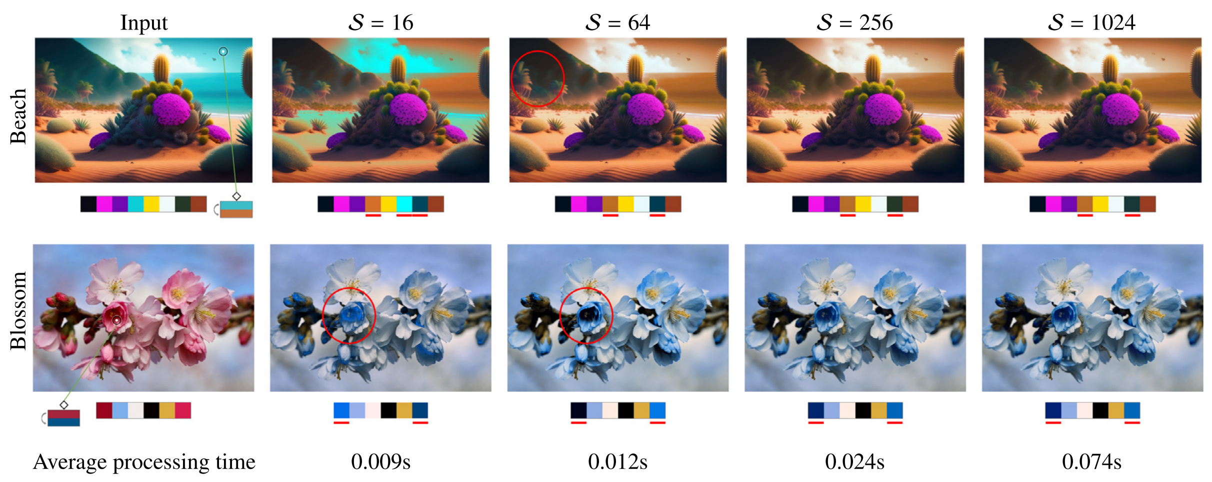
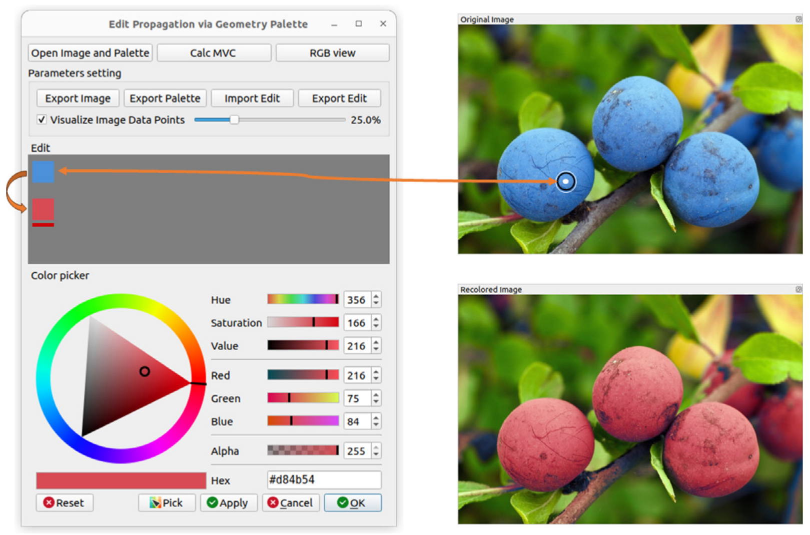


Fig. 2. Recoloring results generated by calculating the correlation term with varying numbers of pixels. In this experiment, we sample 16, 64, 256 and 1024 pixels for the correlation term calculation, respectively. From the results, we observe that $\mathcal{S} = 256$ provides better balance between computational efficiency and locality control.



✓ Visualize Image Data Points 25.0%

Hue 356
Saturation 166
Value 216
Red 216
Green 75
Blue 84
Alpha 255
Hex #d84b54

Fig. 3. Our GUI for color editing. The left side presents the editing tools, including import and export of images and color palettes, color selection and adjustment features. On the right side, a side-by-side comparison of the image before and after editing is displayed, allowing users to easily monitor changes.

is changed to unnatural cyan when $\mathcal{S} = 16$, the hill on the left is alerted to undesirable blue when $\mathcal{S} = 64$, and it produces a comparable result that faithful user edit intention when $\mathcal{S} = 254$ or 1024. In the *Blossom* example, when $\mathcal{S} = 16$ or 64, it produces unpleasant brightness and black inside the flower, while larger $\mathcal{S}$ generate more natural results. In order to balance both computational efficiency and recoloring effects, we set $\mathcal{S} = 254$ for all examples.

5.2. Ablation study

We conducted an ablation study to evaluate the effectiveness of the loss terms in our energy function (Eq. (5)). We give two examples in Fig. 4 to assess the effectiveness of the terms. For each sample, we provide the input and edits, the results generated when the correlation term or the sparsity term are removed, and the results generated with all loss terms. Note that the edit term cannot be removed. Otherwise, the input images will remain unchanged, as user interaction will be disabled in this case. Besides, we also show the extracted palettes and the palettes solved with our optimization method below the images.

In the *Leaf* example, the user intends to transform green leaves into red leaves. It is evident that removing the correlation term would result in an unexpected light green sky, which contradicts the user's editing intention. Similarly, eliminating the sparsity term alters five colors in the palette, causing the flowers to turn an unexpected shade of green. In the *Sunset* example, the user's goal is to change the golden light to red light. Excluding the correlation term leads to an undesirable color shift in the lake, and the flowers and plants exhibit an overly red hue. Omitting the sparsity term, on the other hand, results in a change to five colors and turns the sky green. In contrast, the outcomes generated by incorporating all loss terms align more closely with the user's editing intent.

In summary, the correlation and sparsity terms could avoid a wide range of color variations, and effectively preserve the colors of non-interested regions or objects. By incorporating the edit term, the correlation term, and the sparsity them, our approach achieves promising results that conform well to the user editing goal and preserve the colors of parts not concerned.

5.3. Comparisons

In this part, we qualitatively and quantitatively compare our method with state-of-the-art palette-based and edit propagation approaches. We provide 8 examples in Figs. 5, 6 and 7 for comparison. For each example, we provide the input image, the editing intention (under

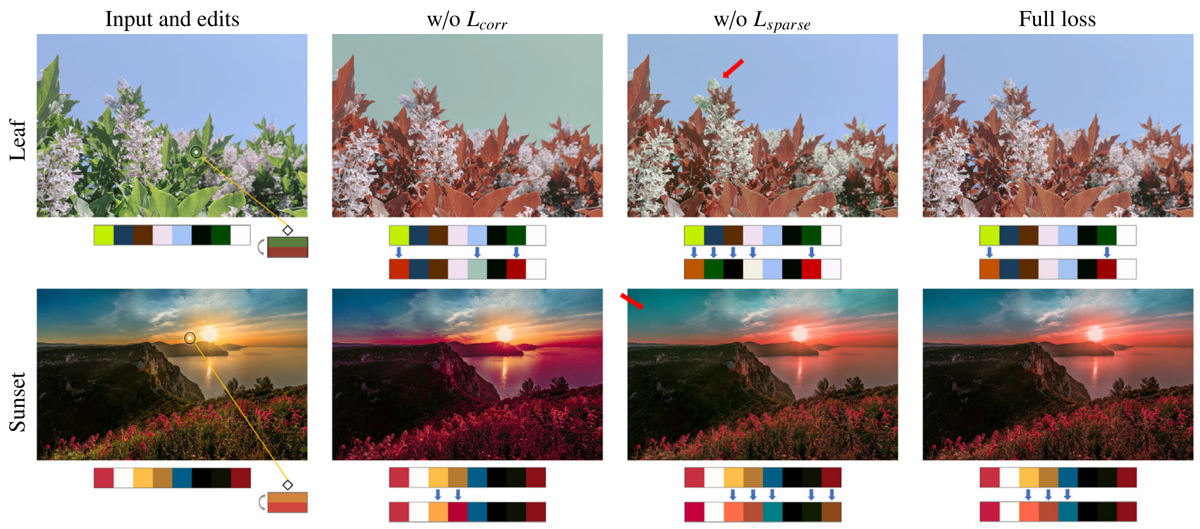


Fig. 4. Ablation study. In this figure, we provide two examples to evaluate the effectiveness of different terms in our loss function (Eq. (5)). We provide the inputs and edits (column 1), the results generated by removing the correlation term (column 2) and the sparsity term (column 3), and the results generated by using the full loss function (column 4). It suffers from global color change and unfaithful results when these two terms are disabled.

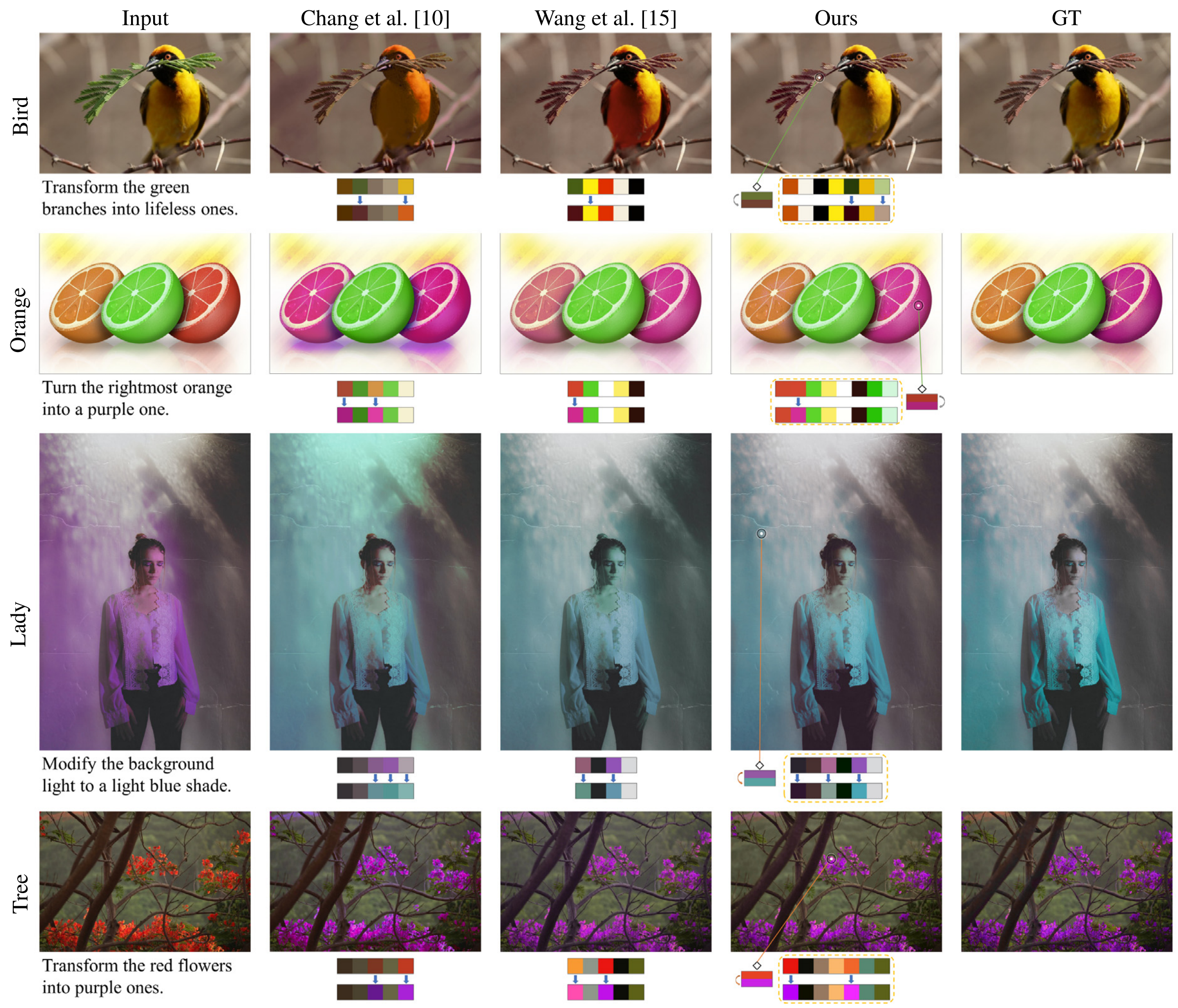

Input
Chang et al. [10]
Wang et al. [15]
Ours
GT
Bird
Transform the green branches into lifeless ones.
Orange
Turn the rightmost orange into a purple one.
Lady
Modify the background light to a light blue shade.
Tree
Transform the red flowers into purple ones.

Fig. 5. Visual comparison with palette-based approaches. In this figure, we compare our method with two palette-based approaches i.e., Chang et al. [10] and Wang et al. [15]. For each example, we provide the input image along with the edit intention (column 1), recoloring results generated by different methods (column 2–4) and the ground truth images (column 5). Generally, existing methods tend to produce undesirable global color change, while our results are more faithful to user edit intentions.

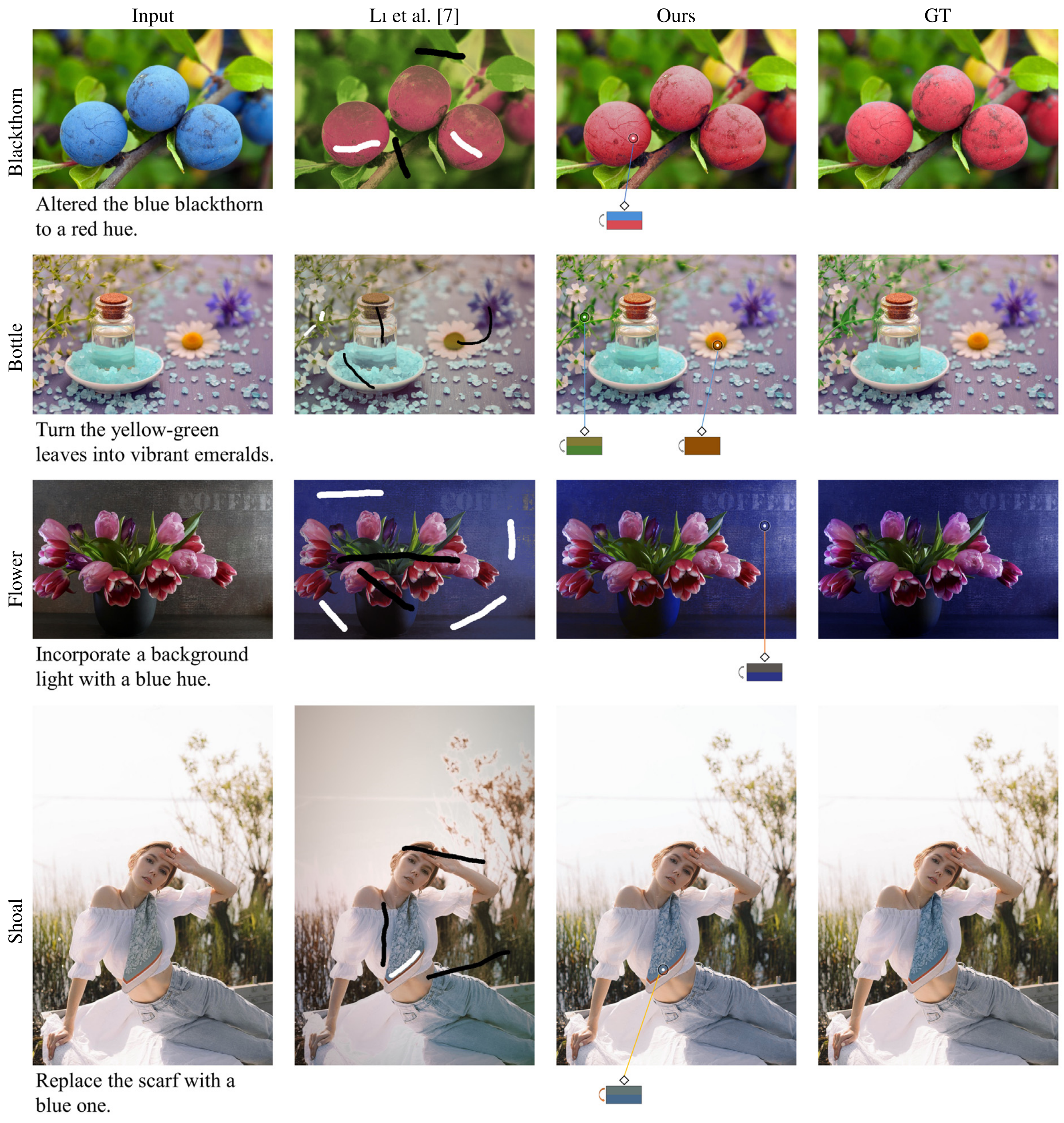

Input
Li et al. [7]
Ours
GT
Blackthorn
Altered the blue blackthorn to a red hue.
Bottle
Turn the yellow-green leaves into vibrant emeralds.
Flower
Incorporate a background light with a blue hue.
Shoal
Replace the scarf with a blue one.

Fig. 6. Visual comparison with edit propagation-based approach. In this figure, we compare our approach with Li et al. [7]'s edit propagation method. For each example, we provide the input image along with the edit intention (column 1), results generated by different methods (column 2–3) and the ground truth image (column 4). Compared with the existing method, our approach needs fewer user interactions and achieves better local control in recoloring.

the input), and edited results generated by different methods. For qualitative comparison, We focus on the quality of the generated image and how well it matches the edit intent. For quantitative comparison, in order to fairly evaluate different methods, we invited experts to manually edit the images using specialized software such as Adobe Photoshop, to obtain ground-truth (GT) reference. We then calculate the Mean Square Error (MSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index Measure (SSIM) between the results produced by each method and the ground truth as an objective evaluation metric. All quantitative comparison results of these examples are provided in Table 1.

5.3.1. *Comparisons with palette-based methods*

We first compare our method with two state-of-the-art palette-based methods: Chang et al. [10] and Wang et al. [15]. The former employs modified k-means clustering to extract the color palette, and uses Radial Basis Function (RBF) interpolation to transfer the color change of the palette to the whole image. The latter calculates the convex hull of colors in the input image and use its vertices as the color palette, and then leverages Mean value Coordinates (MVC) to achieve real-time color editing. Both methods allow the user to adjust the appearance of an image by modifying the extracted palette. While powerful, it does not support direct pixel-level editing and always produces undesirable global color changes in color editing. In contrast, our approach supports direct editing of images, producing results that better fulfill user editing intent.

In Fig. 5, we provide four examples to compare our method with Chang et al. [10] and Wang et al. [15]. For each method, we present the color palettes before and after editing under the edited image. Note that our approach does not manipulate the color palette directly. In the fourth column, we just show the original palette and the palette solved by our optimization method. In our method, the selected pixels and

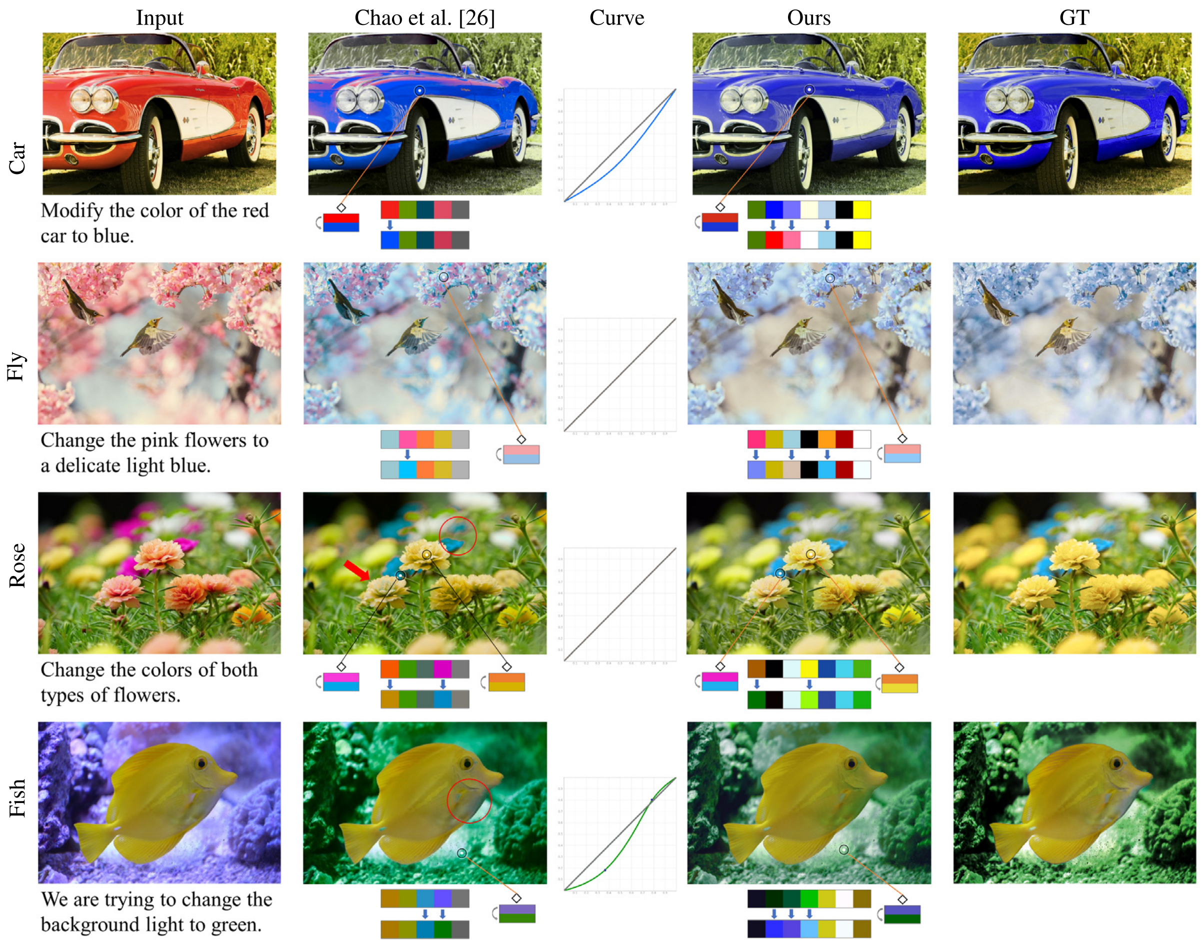

Input
Chao et al. [26]
Curve
Ours
GT
Car
Modify the color of the red car to blue.
Fly
Change the pink flowers to a delicate light blue.
Rose
Change the colors of both types of flowers.
Fish
We are trying to change the background light to green.

Fig. 7. Visual comparison with Chao et al. [26]'s pixel-level image editing method. For each example, we provide the input along with the user editing intend (column 1), the recoloring result and edits on image space (column 2) and curve (column 3) of Chao et al. [26], our result (column 4), and the GT (column 5). In all examples, the results generated by our method better match user intends.

Table 1
Quantitative comparison with existing methods.

	Example	MSE↓			PSNR(dB)↑			SSIM↑		
		[10]	[15]	Ours	[10]	[15]	Ours	[10]	[15]	Ours
Comparison with palette-based methods	Bird	0.0055	0.0010	**0.0002**	22.59	29.83	**36.37**	0.9489	0.9682	**0.9873**
	Orange	0.0078	0.0026	**0.0007**	21.09	25.84	**31.36**	0.9333	0.9703	**0.9783**
	Lady	0.0032	0.0013	**0.0007**	24.89	28.76	**31.78**	0.9569	0.9744	**0.9828**
	Tree	0.0022	0.0013	**0.0010**	26.52	28.85	**29.67**	0.9288	0.9536	**0.9546**
		[7]		Ours	[7]		Ours	[7]		Ours
Comparison with edit propagation method	Blackthorn	0.0118		**0.0020**	19.26		**26.80**	0.8029		**0.9410**
	Bottle	0.0035		**0.0004**	24.46		**33.25**	0.9468		**0.9848**
	Flower	0.0072		**0.0026**	21.45		**25.77**	0.8353		**0.9254**
	Shoal	0.0139		**0.0002**	18.57		**36.80**	0.9635		**0.9968**
		[26]		Ours	[26]		Ours	[26]		Ours
Comparison with pixel-level image editing method	Automobile	0.0116		**0.0079**	19.34		**21.04**	0.7767		**0.8673**
	Fly	0.0035		**0.0021**	24.51		**26.86**	0.9214		**0.9652**
	Rose	0.0083		**0.0044**	20.83		**23.55**	0.8676		**0.9401**
	Fish	0.0144		**0.0086**	18.40		**20.64**	0.6999		**0.8045**

colors before and after changes are given in the edited images. Besides, the ground truths edited by experts are displayed in the last column.

In the *Bird* example, the user expects to change the green branch in the bird's beak to a dead branch to create a feeling of depression. Existing methods produce unexpected color changes in feathers, making their colors turn red. Besides, Chang et al. [10] darken the image's overall brightness. In the *Orange* example, oranges on both sides have similar colors. The user intends to make the red orange on the right into a purple orange. Existing methods struggle to distinguish these two similarly colored oranges and edit them separately. As we can see, the color of the orange on the left is changed inevitably. In the *Lady* example, the user has to modify multiple colors to change the purple background to light blue, as shown in the reference. In the *Tree* example, the user wants to adjust the flowers from red to purple. Existing methods produce unexpected color changes on the branches.

In summary, existing palette-based methods tend to produce global changes in appearance. This is because the color palettes generated by existing methods usually contain fewer colors. While it is easy to manipulate, the mixing weights are not sparse enough to produce global color variations. In contrast, Our approach supports palettes that contain more colors because we do not operate directly on the palette. The color palette just assists us with efficient color propagation. Compared to the state-of-the-art palette-based methods, the results generated by our method better fulfill user intent.

5.3.2. Comparisons with edit propagation

In this part, we compare our method with the edit propagation approach. Since some recent works such as [9,20] did not release their source code, we only compare our method with a similar real-time edit propagation method proposed by Li et al. [7]. They formulate the edit propagation as an efficient interpolation problem. They first define a set of Radial Basis Functions (RBFs) on stroked pixels, and then the edited color of each pixel is expressed as a linear combination of these RBFs. While it significantly accelerates previous methods, it needs density interactions.

In Fig. 6, we also provide four examples to compare our method with Li et al. [7]. For each sample, we give the input (1st column), user editing intention (under the input), results generated by Li et al. [7] (2nd column), and our method (3rd column), and the reference images edited by experts (4th column). In Li et al.'s results, there are two types of user strokes: white strokes indicate color modification, while black strokes indicate that the color stays the same. After putting these strokes, it then automatically propagates to the whole image.

In the *Blackthorn* example, the user wants to change its color to red. As can be seen in Li et al. [7]'s result, the fruits and leaves in the upper right corner look noticeably darker than the input colors. In contrast, our results are more natural, and only need to modify one pixel on the fruit to generate a desirable result. In the *Bottle* example, the user intends to make the yellow plants to green plants. For li et al. [7]'s method, while we put stroke to retain the color of the flower on the ground, their method still fails to preserve the stamen's color. In the *Flower* example, the results generated by Li et al. [7] and our method are similar, but their approach needs much more user interaction. In the *Shoal* example, the user attempts to change the color of the scarf to blue. The existing method fails to keep the colors of the sky and the lady's jeans. In contrast, our method effectively achieves the editing goal with minimal impact on the rest of the image.

In general, the edit propagation-based approach relies entirely on user strokes, often leading to unexpected global color changes in color editing. It usually requires users to put fine-tuned and density strokes to achieve good results, which burdens novice or inexperienced users. Our approach typically only requires the user to modify a handful of pixels and well maintain the color of non-interested regions.

5.3.3. Comparisons with pixel-level image editing method

In Fig. 7, we compare our method against Chao et al. [26], a recent pixel-level recoloring approach. Although their method allows for easy adjustment of image brightness, it often struggles to produce smooth recoloring results for complex scenes, producing some unexpected artifacts. This is due to two main issues: first, their method does not adequately consider preserving the colors of uninterested regions; second, their energy function is overly complex, making it difficult to converge to an acceptable result in a short time. In contrast, our method better preserves the color of the uninterested region in a simpler form, resulting in improved recoloring results and faster convergence.

In the *Car* example, the user expects to modify the color of the car from red to blue, Chao et al. [26] method produces unexpected red color on the car, while our result is more reasonable without any artifact. Similarly, in the *Fly* example, the user wants to change the color of the flowers from pink to light blue, under the same user edits, Chao et al. [26]'s result exhibits obvious artifacts, while our method produces smoother and more harmonious recoloring results. In the *Rose* example, the user aims to adjust the colors of two different flowers in the foreground and background. Chao et al. [26]'s method tends to introduce unexpected blue at the center of the flower. In contrast, our method effectively captures the user's intention without such issues. In the *Fish* example, Chao et al. [26]'s method creates unnatural color transitions at the edges of the fish, whereas our method achieves smoother color transitions.

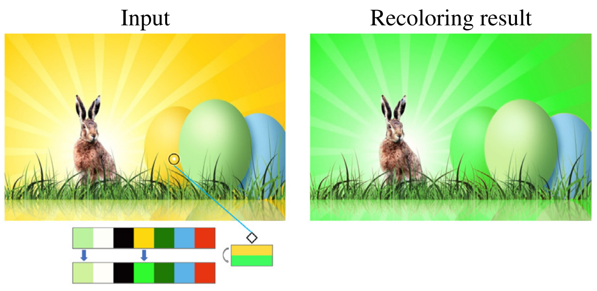


Fig. 8. Failure case. Our method cannot distinguish different objects with the same color and perform separate color editing to these objects. In this example, We try to change the yellow egg to green, but the color of the background is also altered to green.

At the end of the comparison subsection, we would like to clarify two things. Firstly, the palette size of different methods are automatically determined. Secondly, readers may notice that our palettes contain more colors than existing methods in many examples. With regard to the second fact, we have two considerations. On the one hand, our method does not directly use the palette for color editing, instead, the palette serves as a hidden tool, therefore, a larger palette size does not impose an additional burden to users in color editing. On the other hand, a palette with more colors usually ensures better local control in recoloring. However, existing palette-based methods typically require the user to edit the palette, thus limiting the palette to contain fewer colors. Theoretically, Wang at al [15]'s method is capable of achieving comparable color editing results when utilizing the same color palette as our method. Nevertheless, it consumes considerable time to select and adjust colors accordingly, whereas our approach demonstrates greater efficiency by operating directly on the image.

We provide more recoloring results in Fig. 9 to further demonstrate the effectiveness of our method.

5.4. User study

To further illustrate the effectiveness of our method, we conducted a user study. We invited 61 participants to evaluate the results generated by existing methods and ours. Specifically, our experiments included three aspects:

- **Results evaluation 1**. We showed each participant with 5 examples. For each example, we provided the input, the edit intention, and the results generated by Li et al. [7], Chang et al. [10], Wang et al. [15] and ours, then we asked participants to choose the result that best matched the edit intention.
- **Results evaluation 2**. Based on Results evaluation 1, for each example, we also provided the result generated by the artist based on the editing intent as a reference. And then, we asked the participant to choose the results closest to the reference.
- **Ease of use evaluation**. We asked participants to edit the images using different methods and to evaluate their convenience.

The results of the user study are generally in line with our expectations. For "Results evaluation 1'', our results received more votes in all examples. On average, 66% agree that our results better match the given edit intentions. While 7% prefer Li et al. [7], 17% choose Chang et al. [10] and 10% for Wang et al. [15]. For "Results evaluation 2", on average, 56% believe that our results are closer to the references. In addition, results generated by Chang et al. [10] receive the most votes with 25%. For "Ease of use evaluation", over 60% of the participants find our editing interface more intuitive and easy to use.

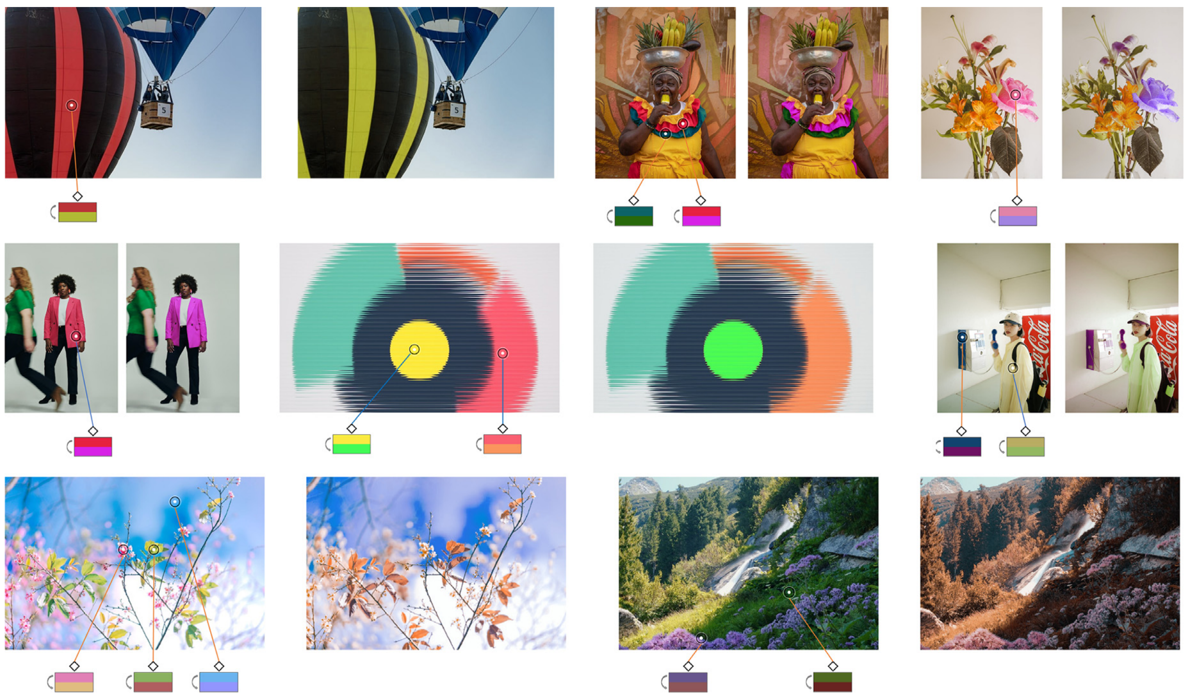


Fig. 9. Gallery. Diversity recoloring results produced by our method. For each pair of images, the left one is the input along with the user edits, the right one is the recoloring result.

6. Conclusion

Inspired by the advance of palette-based methods, in this paper, we introduce color palette into edit propagation, to achieve efficient yet easy-to-use color editing. We further formulate the edit propagation task as an efficient palette optimization problem. It allows the user to directly modify pixels to adjust the color of the input image. And users can get real-time feedback in color editing. Extensive experiments and a user study have demonstrated the effectiveness of our method.

Our method has a primary limitation. Our method works in RGB color space, so only color similarity is considered in edit propagation. Our approach struggles to achieve the desired results for some challenging application scenarios. As shown in Fig. 8, if an image contains a yellow egg and a yellow background, it would be difficult for user to recolor them to different colors using our method without introducing semantic information. In the future, we would like to extend the current approach from RGB space to high-dimensional semantic space, to achieve more challenging color editing tasks.

CRediT authorship contribution statement

Zi-Xun Xia: Writing – original draft, Validation, Software, Methodology. **Jian-Yu Hao:** Software, Resources. **Kang Li:** Visualization, Validation. **Ao-Xiang Tian:** Investigation. **Zheng-Jun Du:** Writing – review & editing, Supervision, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

We appreciate the anonymous reviewers for their valuable comments and suggestions. This work is supported by the Youth Program of the Natural Science Foundation of Qinghai Province (Project Number: 2023-ZJ-951Q).

References

[1] Levin A, Lischinski D, Weiss Y. Colorization using optimization. In: ACM SIGGRAPH 2004 papers. 2004, p. 689–94.
[2] Pellacini F, Lawrence J. AppWand: Editing measured materials using appearance-driven optimization. In: ACM SIGGRAPH 2007. 2007, p. 54–es.
[3] An X, Pellacini F. Appprop: all-pairs appearance-space edit propagation. In: ACM SIGGRAPH 2008 papers. 2008, p. 1–9.
[4] Xu K, Wang J, Tong X, Hu S-M, Guo B. Edit propagation on bidirectional texture functions. In: Computer graphics forum, vol. 28, no. 7. Wiley Online Library; 2009, p. 1871–7.
[5] Xu K, Li Y, Ju T, Hu SM, Liu TQ. Efficient affinity-based edit propagation using kd tree. ACM Trans Graph 2009;28(5):1–6.
[6] Chen X, Zou D, Zhao Q, Tan P. Manifold preserving edit propagation. ACM Trans Graph 2012;31(6):1–7.
[7] Li Y, Ju T, Hu S-M. Instant propagation of sparse edits on images and videos. In: Computer graphics forum, vol. 29, no. 7. Wiley Online Library; 2010, p. 2049–54.
[8] Chen X, Zou D, Li J, Cao X, Zhao Q, Zhang H. Sparse dictionary learning for edit propagation of high-resolution images. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2014, p. 2854–61.
[9] Li F, Ou C, Gui Y, Xiang L. Instant edit propagation on images based on bilateral grid. Comput Mater Continua 2019;61(2).
[10] Chang H, Fried O, Liu Y, DiVerdi S, Finkelstein A. Palette-based photo recoloring. ACM Trans Graph 2015;34(4):1–11.
[11] Tan J, Lien JM, Gingold Y. Decomposing images into layers via RGB-space geometry. ACM Trans Graph 2016;36(1):1–14.
[12] Aksoy Y, Aydin TO, Smolić A, Pollefeys M. Unmixing-based soft color segmentation for image manipulation. ACM Trans Graph 2017;36(2):1–19.
[13] Tan J, Echevarria J, Gingold Y. Efficient palette-based decomposition and recoloring of images via RGBXY-space geometry. ACM Trans Graph 2018;37(6):1–10.

[14] Tan J, Echevarria J, Gingold Y. Palette-based image decomposition, harmonization, and color transfer. 2018, arXiv preprint arXiv:1804.01225.
[15] Wang Y, Liu Y, Xu K. An improved geometric approach for palette-based image decomposition and recoloring. In: Computer graphics forum, vol. 38, no. 7. Wiley Online Library; 2019, p. 11–22.
[16] Lischinski D, Farbman Z, Uyttendaele M, Szeliski R. Interactive local adjustment of tonal values. ACM Trans Graph 2006;25(3):646–53.
[17] Bie X, Huang H, Wang W. Real time edit propagation by efficient sampling. In: Computer graphics forum, vol. 30, no. 7. Wiley Online Library; 2011, p. 2041–8.
[18] Xiao C, Yongwei N, et al. Efficient edit propagation using hierarchical data structure. IEEE Trans Vis Comput Graph 2010;17(8):1135–47.
[19] Endo Y, Iizuka S, Kanamori Y, Mitani J. Deepprop: Extracting deep features from a single image for edit propagation. In: Computer graphics forum, vol. 35, no. 2. Wiley Online Library; 2016, p. 189–201.
[20] Gui Y, Zeng G. Joint learning of visual and spatial features for edit propagation from a single image. Vis Comput 2020;36(3):469–82.
[21] Zhu JY, Park T, Isola P, Efros AA. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision. 2017, p. 2223–32.
[22] Ju T, Schaefer S, Warren J. Mean value coordinates for closed triangular meshes. In: ACM Siggraph 2005 papers. 2005, p. 561–6.
[23] Sun Q, Nie Y, Zhang Q, Li G. Building coarse to fine convex hulls with auxiliary vertices for palette-based image recoloring. IEEE Trans Vis Comput Graphics 2023.
[24] Du ZJ, Lei KX, Xu K, Tan J, Gingold Y. Video recoloring via spatial-temporal geometric palettes. ACM Trans Graph 2021;40(4):1–16.
[25] Zhang Q, Nie Y, Zhu L, Xiao C, Zheng WS. A blind color separation model for faithful palette-based image recoloring. IEEE Trans Multimed 2021;24:1545–57.
[26] Chao CKT, Klein J, Tan J, Echevarria J, Gingold Y. ColorfulCurves: Palette-aware lightness control and color editing via sparse optimization. ACM Trans Graph 2023;42(4). http://dx.doi.org/10.1145/3592405.
[27] Cho J, Yun S, Mu Lee K, Young Choi J. Palettenet: Image recolorization with given color palette. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2017, p. 62–70.
[28] Akimoto N, Zhu H, Jin Y, Aoki Y. Fast soft color segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 8277–86.
[29] Chao CKT, Klein J, Tan J, Echevarria J, Gingold Y. LoCoPalettes: Local control for palette-based image editing. Comput Graph Forum (CGF) 2023;42(4). http://dx.doi.org/10.1111/cgf.14892, Special issue for Eurographics Symposium on Rendering (EGSR).
[30] Gurobi Optimization, LLC. Gurobi optimizer reference manual. 2023, URL: https://www.gurobi.com.