Palette-based Content-Aware Image Recoloring

Zheng-Jun Du[1,2][0000−0002−6763−2892], Jia-Wei Zhou[1][0009−0000−6623−0050], Zi-Xun Xia[1][0009−0006−4685−920X], Bing-Feng Seng[1][0009−0000−4005−030X], and Kun Xu[3⋆][0000−0002−2671−4170]

[1] Department of Computer Technology and Application, Qinghai University, Xining, 810016, China
{dzj,zjw,xia.zixun,sbf}@qhu.edu.cn
[2] Qinghai Provincial Key Laboratory of Media Integration Technology and Communication, Xining, 810016, China
[3] BNRist, Department of CS&T,Tsinghua University, Beijing, 100084, China
xukun@tsinghua.edu.cn

Abstract. Palette-based image recoloring is a popular image editing technique in recent years. It allows users to perform global color edits to an image by manipulating a small set of representative colors. Many approaches have been proposed for palette extraction and palette-based image recoloring. However, existing methods primarily leverage low-level visual information to extract color palettes, so that different objects with similar colors will share the same palette colors. It is impossible to individually recolor one of multiple objects with similar colors, as altering a specific palette color may cause unwanted color changes to many non-interesting objects. To address this issue, in this paper, we present a novel, palette-based content-aware image recoloring approach. Different from previous methods, we extract the color palette of an image in a high-dimensional space that combines low-level visual features and high-level semantic features, allowing generating separate palette colors for different objects with similar colors. This enables users to perform targeted local editing, i.e., distinguish and recolor objects with similar colors separately, without producing unexpected global color changes. Extensive experiments demonstrate the flexibility, local control, and effectiveness of our method.

Keywords: palette · recoloring · content-aware · semantic · local editing.

1 Introduction

Palette-based image recoloring has attracted increasing attention in recent years. Many approaches have been proposed for palette extraction and recoloring, and have demonstrated promising results in color editing. In these approaches, a color palette consisting of a small set of colors is first extracted from the input image, to characterize its color distribution. Subsequently, a predetermined mapping function is employed to transfer the modifications made to the color palette to

⋆ Kun Xu is the corresponding author

the entire image. So that users could easily adjust the image by modifying the palette colors. These methods are generally efficient, easy to use and learn, and can generate natural, artifact-free results.

Despite these advantages, a long-standing problem is that these methods are limited to handle only global color editing while do not support content-aware local color editing. For example, when an image comprises multiple objects with identical or similar colors, users struggle to alter the color of one object without impacting the others. The primary reason is that existing methods typically extract color palettes in a low-dimensional color space (e.g., RGB or Lab color space), objects with similar colors will share the same palette colors. Therefore, modifying a palette color may cause all objects with similar colors to change at the same time. Existing methods cannot distinguish different objects with similar colors, nor can they recolor each of these objects separately.

To address this problem, in this paper, we present a novel, palette-based content-aware image recoloring method. Our method contains two steps: color palette extraction and content-aware recoloring. In the first step, unlike existing algorithms that usually extract color palettes in 3D RGB or Lab color space, we project the input image into a high-dimensional feature space that contains both low-level visual information and high-level semantic information, and then employ a variant K-means clustering to obtain the palette. So that different objects with similar colors will be associated with different palette colors. In the second step, we design a color transfer function to map the change of the color palette to the whole image. In the color transfer function, the color change of each pixel is defined as a weighted sum of palette color changes, while the weight of a pixel with respect to a specific palette color is determined by their similarity. Our approach is efficient and could generate desirable results that match user edit intention. More importantly, it enables local and content-aware recoloring.

We have demonstrated the effectiveness of our method on extensive experiments. Compared with existing methods, the contributions of our paper lie in:

- We propose to extract the color palette of an image in a high-dimensional space that combines both low-level visual features and high-level semantic features.
- With the extracted color palette, we design a color transfer function to map the changes of palette colors to the whole image. Our approach enables palette-based content-aware recoloring for the first time, allowing users to perform targeted localized editing in complex scenes.

2 Related works

Image recoloring is a frequent operation executed by graphical designers. Its purpose is to adjust the colors of an image to improve the quality, enhance the artistic effects, or meet some specific design needs. Three types of methods have been proposed for image recoloring, i.e., palette-based methods, stroke-based methods (edit propagation), and example-based methods (style transfer). Next, we will briefly review these three types of methods.

2.1 Palette-based image recoloring

Palette-based image recoloring is a popular topic in image editing. It offers an intuitive yet efficient solution for color adjustment. It enables users to interactively change the color of an image by modifying a color palette.

The pioneering work of palette-based image recoloring was introduced by Chang et al. [4]. They employed a modified K-means clustering to extract the color palette of an image and mapped the changes of the color palette to the entire image with a radial basis function weighted transformation. Similarly, Zhang et al. [29] also used a similar clustering method to obtain the color palette, but they represented image pixels as the weighted sum of palette colors through optimization. During recoloring, the weights are fixed, and users modify the palette colors to change the color of the image. A series of methods based on convex hull in RGB space have been proposed for image recoloring. Tan et al. [25] calculated the simplified convex hull of image colors in RGB space and used its vertices as the color palette. They then decomposed the input image into ordered layers corresponding to the palette colors. Users adjust the image by modifying the layers or the color palette. Later, Tan et al. [23, 24] proposed a more efficient method for palette extraction and image recoloring using RGBXY space geometry. Wang et al. [27] further proposed an optimization method to iteratively move the vertices (colors) of the simplified convex hull (palette), to improve the compactness and representativeness of the color palette. Du et al. [9] Extended the palette-based image recoloring to video scenario, and achieved natural yet time-varying color editing. More recently, Chao et al. [5] proposed the "ColorfulCurves" to achieve both color and lightness adjustment. So that users can directly modify palette colors' hue and saturation and per-palette tone curves, or image pixels, to recolor the input image. However, when different objects or regions have similar colors, their method can not recolor them separately. Chao et al. [6] introduced an adaptive solution for recoloring under arbitrary image-space constraints and automatically splits the image into soft sub-regions with more representative local palettes when the constraints cannot be satisfied.

All these methods extract color palettes of input images in RGB or Lab color space. That is to say, they only leverage the low-level color information to obtain the palettes. Different regions or objects that have similar colors necessarily share the same palette color. Thus, modifying some palette color will inevitably result in color changes of multiple regions or objects with similar colors. However, in this paper, we extract the color palette considering both the low-level visual color information and the semantic information. It enables the color palette to incorporate specific semantic information, thereby effectively differentiating between objects with similar colors.

2.2 Edit propagation (stroke-based image recoloring)

Edit propagation is a well-studied technique for image recoloring. It allows users to put sparse edits directly onto the image, which are then automatically propa-

gated to the rest of the image. This technique has been widely used in colorization, color editing, material editing, etc.

The first edit propagation method was presented by Levin et al. [16] for stroke-based grayscale image colorization. They started by converting the grayscale image into the YUV color space, and assumed that pixels that have similar luminance (Y) values should receive similar chrominance (UV) values. Based on this assumption, the colorization task is formulated as a quadratic energy optimization problem and further reduced to solving a linear system. Later, Lischinski et al. [18] introduced the idea of edit propagation into tonal adjustment and achieved impressive results. Pellacini et al. [20] further extended this method to edit measured materials. An et al. [3] proposed the first edit propagation-based approach for color editing. They calculated the similarities in all pairs of pixels to achieve distant propagation. All of the above methods require solving large-scale linear systems, which are computationally and storage expensive. To speed up these algorithms and reduce the memory burden, Xu et al. [28] proposed an efficient affinity-based edit propagation using a kd tree, which significantly accelerates this algorithm and saves memory overhead. Li et al. [17] formulated the edit propagation task as an interpolation problem with radial basis functions (RBFs), which first achieved real-time color editing. Recently some deep learning-based approaches have proposed to edit propagation. The first deep learning-based edit propagation method was proposed by Endo et al. [10], termed "DeepProp", which utilizes a convolutional neural network (CNN) to extract high-dimensional features for edit propagation. Gui et al. [12] formulated edit propagation as a multi-class classification problem and utilized a fully convolutional network capable of end-to-end training to extract visual and spatial features and predict the resulting image. These methods are sensitive to user edits and cannot generate results in real-time.

Edit propagation-based approaches typically require users to make density fine-tuned edits, and cannot enable content-aware color editing. Our method only requires users to manipulate a small set of colors to adjust the appearance of an image. Moreover, it can achieve content-aware recoloring.

2.3 Style transfer (example-based image recoloring)

Style transfer is another widely researched method for image recoloring, which involves transferring the colors from a reference image to a target image.

A bunch of methods have been proposed for style transfer. For instance, Reinhard et al. [22] mapped the style from the reference image to the target one by aligning their color distributions. Neumann et al. [19] employed a 3D histogram matching technique for color transfer, enabling natural results even when the correspondence between the reference and target images is faint. Pitie et al. [21] developed a continuous transformation that maps one n-dimensional distribution to another, effectively transferring color between two images with varying content. However, these methods primarily focus on low-level visual features, which may lead to significant artifacts when the reference and target images are considerably different. In recent years, deep learning-based algorithms

have made substantial advancements in the field of image style transfer. Notable examples include CNN-based style transfer algorithms such as [11, 26] and GAN-based style transfer algorithms like [8, 30]. These techniques take into account semantic feature correspondence between the reference and target images.

Despite the style transfer is powerful, these approaches offer limited editing controls to users beyond the selection of the reference image. Furthermore, these methods focus on global color editing rather than object-level color adjustment. Whereas palette-based approaches make a better balance between user control and convenience, and support color editing for local objects.

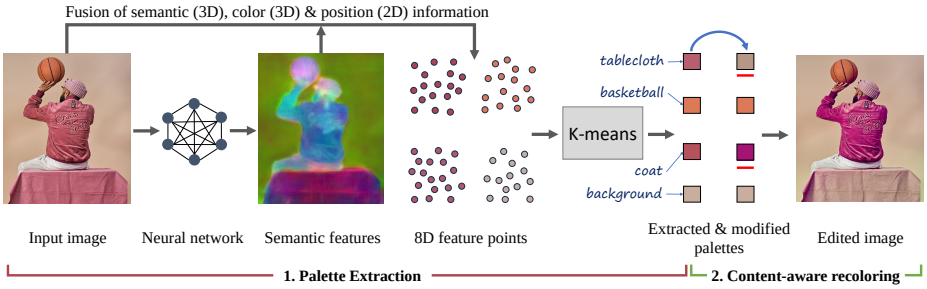


Fig. 1. The pipeline of our approach. Given the input image, we first feed it into a neural network to obtain per-pixel semantic features. Next, we combine it with color and position information, and project them into an 8D feature space, followed by a K-means clustering to extract the palette. Finally, the user modifies the palette colors to locally adjust object colors. In the input image, the man's coat and tablecloth have a similar red color. It is challenging to recolor them separately with existing palette-based methods. Our palette successfully extracts two similar red colors that correspond to the coat and the tablecloth, respectively, so the user can easily recolor the coat and the tablecloth into different colors.

3 Method

3.1 Overview

The goal of our method is two folded. First, we would like to extract a color palette that can effectively capture the dominant colors of different objects or regions in the input image, so that each palette color can be associated with a specific object or region. Second, we would like to achieve content-aware color editing based on the extracted palette.

Hence, our method could be naturally divided into two stages. The first stage is palette extraction (Section 3.2). We first extract per-pixel semantic features from the input image using a neural network. Then, we project the input image into a high-dimensional feature space that incorporates semantic, color, and location information, so that each pixel is regarded as a vertex in this feature space. Finally, we acquire the color palette with a modified K-means clustering method. The second stage is content-ware recoloring (Section 3.3). This is achieved by

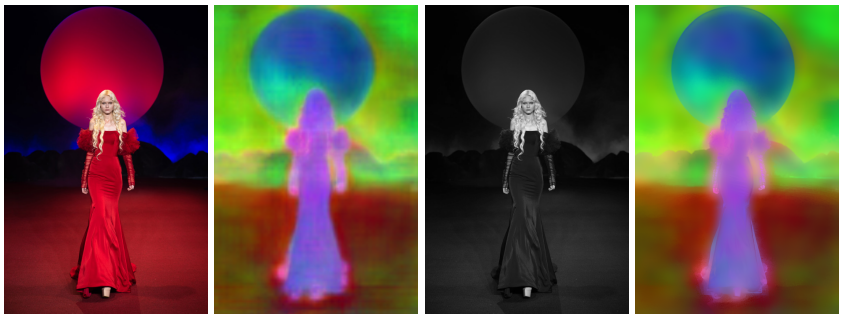
performing a color transfer function to map the changes of the palette colors to corresponding objects or regions of the input image. The pipeline of our method is illustrated in Fig. 1.

3.2 Palette extraction

Our palette extraction algorithm takes an image along with the palette size as input and outputs a palette. Generally, it consists of two steps, i.e., high-dimensional feature space construction and palette extraction with K-means.

High-dimensional feature space construction In this paper, we first utilize the method proposed by Aksoy et al. [2] to generate per-pixel semantic features of the input image. In their work, they designed a feature extractor based on DeepLab-ResNet-101 [7], and cascaded with metric learning [14], to generate features that are as similar as possible if they belong to the same class, and distant from each other otherwise. This feature extraction network takes an image as input and generates a 128-dimensional feature vector for each pixel, and has shown impressive results in semantic segmentation of images.

To remove redundant data and reduce computational overhead, we adopt principal component analysis (PCA) [15] to reduce the dimension of feature vectors from 128 to 3. We visualize the reduced semantic feature map as a three-channel color image and find that it contains some unwanted noise. To reduce noises, we further smooth the extracted semantic features by guided filtering [13] which takes the grayscale image of the input as the guiding image, to better preserve the edges of objects or regions during smoothing. We provide such an example in Fig. 2. From the results, we could find that the features extracted by the neural network are intuitive and can roughly reflect the semantics of different objects in the image. The initial semantic feature image contains a lot of noise, while the filtered feature image is smoother and the boundaries of different objects can be well preserved.



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

Fig. 2. Visualization of initial and filtered semantic feature maps. (a) Input, (b) Initial semantic feature, (3) The guiding image, (d) Filtered semantic features.

To facilitate the color palette to effectively capture different objects or regions in the image and their representative colors, we project the input image into a

high-dimensional feature space for palette extraction. Specifically, we build an 8D feature space that combines semantic, color, and location information together. So any pixel I_i in the input image can be viewed as an 8D point:

$$p_i = \left(r_i, g_i, b_i, x_i, y_i, f_i^1, f_i^2, f_i^3\right) \tag{1}$$

Where (r_i, g_i, b_i), (x_i, y_i) and $\left(f_i^1, f_i^2, f_i^3\right)$ are the color, location and semantic features of pixel I_i, respectively. Therefore, the input image can be naturally regarded as a point set in this 8D feature space.

Palette extraction with K-means Next, we employ K-means clustering to extract the palette of the input image. However, directly performing clustering on all pixels is computationally expensive. To speed up, we adopt a classical superpixel segmentation method, i.e., Simple Linear Iterative Clustering (SLIC) [1], to segment the input image into superpixels and use their centroids as the sampling points. Subsequently, we perform K-means clustering on these sampling points.

It is well known that the K-means algorithm is sensitive to the initial cluster centers. To get better initial values, we utilize a similar algorithm to farthest point sampling to choose the initial k cluster centers. The input of the algorithm includes: the sampling point set that contains the centriods of all superpixels $P = \{p_i\}$, and the desired number of palette colors (i.e., desired number of clusters) k. The output is the initial cluster center set C later used for K-means. Initially, $C = \varnothing$. The algorithm performs as follows:

1) For each sampled point $p_i \in P$, assign it a saliency value $\varphi_i = n_i$, where n_i denotes the pixel count of the superpixel where p_i is located.
2) Select the point $p_i \in \{P - C\}$ with the largest saliency value and add it into the clustering center set C.
3) Update the saliency value of all candidate points by:

$$\varphi_j = \left(1 - \exp\left(-\|p_i - p_j\|^2\right)\right) \cdot \varphi_j, \tag{2}$$

   where p_i is the newly selected point and p_j iteratives over all sampled points.
4) Return to 2) until k points are selected.

Once the initial cluster center set $C = \{C_i\}$ is determined, we perform K-means clustering on the sampling point set P for further refinement of C.

In the clustering process, the distance $d\left(p_i, C_j\right)$ from a point p_i to a clustering center C_j is defined as the weighted sum of a color term, a location term and a semantic term:

$$d\left(p_i, C_j\right) = \theta_c d\left(p_i^{RGB}, C_j^{RGB}\right) + \theta_p d\left(p_i^{XY}, C_j^{XY}\right) \quad + \theta_s d\left(p_i^{F}, C_j^{F}\right) \tag{3}$$

Where $d\left(p_i^{RGB}, C_j^{RGB}\right)$, $d\left(p_i^{XY}, C_j^{XY}\right)$ and $d\left(p_i^{F}, C_j^{F}\right)$ denote the L2 distance of color, location and semantic feature between p_i and C_j, respectively. θ_c, θ_p and θ_s denote the relative contributions of these three terms, respectively. We

empirically set $\theta_c = 1.0$, $\theta_p = 0.2$ and $\theta_s = 5.0$ in our experiments. Finally, the converged cluster centers $C = \{C_i\}$ is the palette we need for the input image.

The algorithm mentioned above is presented in Algorithm 1. Note that we have actually extended the original definition of a color palette. In our method, each entry of a palette contains a 8D feature (i.e., 3D color + 2D location + 3D semantics), which no longer solely represents a color, but also includes location and semantic information. For the sake of clarity, we will continue to refer to it as the color palette in this paper. Such an extended palette enables performing content-aware recoloring.

Algorithm 1 Palette Extraction

Input:
an image I and the palette size k
Output:
an 8D palette $C = \{C_1, C_2, \cdots C_k\}$
build an 8D vector (3D color + 2D location + 3D semantics) for each pixel (Eq. 1)
segment the input image into superpixels S and sample a set of points P
let $C = \varnothing$
assign each sampled point p_i a saliency value $\phi_i = n_i$
while not all k seeds are determined **do**
    Select the point $p_i \in \{P - C\}$ with the largest saliency value and add it into the clustering center set C.
    update the saliency value of all candidate points (Eq. 2)
end while
perform K-means clustering on all pixels with the seeds C
return the converged centers as the palette of the input image

3.3 Content-aware recoloring

In this section, we design a color transfer function to map the changes of the palette colors to the input image. Our color transfer function mainly follows the principle of similarity, i.e., when a palette entry is changed during recoloring, the colors of objects or regions that are semantically, chromatically, and spatially similar to it will change accordingly, while other irrelevant objects or regions will be kept as unchanged as possible. Specifically, for any pixel I_i (its corresponding 8D feature point is p_i), its edited color I'_i is defined as:

$$I'_i = I_i + \sum_{j=1}^{k} w_j^{p_i} \left(C'_j - C_j\right)_{0:3} \quad \text{and} \quad \sum_{j=1}^{k} w_j^{p_i} = 1 \tag{4}$$

Where k is the palette size, C_j and C'_j are the j-th palette entry before and after modifying, $(\cdot)_{0:3}$ denotes the color component of an 8D vector, $w_j^{p_i} \in [0, 1]$ denotes the similarity weight of p_i with respect to the palette entry C_j.

Before defining $w_j^{p_i}$, we first define a similarity function $S_j(x)$ for any palette entry C_j, to measure the similarity between any pixel x and it. It is defined as

a linear combination of a set of radial basis functions:

$$S_j(x) = \sum_{i=1}^{k} \lambda_{j,i} \phi\left(x, C_i\right) \tag{5}$$

Where $\lambda_{j,i}$ is the coefficient to be solved, and $\phi\left(x, C_i\right)$ is a radial basis function which is defined as the product of three Gaussian kernels:

$$\begin{aligned} \phi(x, C_i) = & \exp\left(\frac{-\left(x^{RGB} - C_i^{RGB}\right)^2}{2\sigma_c^2}\right) \cdot \exp\left(\frac{-\left(x^{XY} - C_i^{XY}\right)^2}{2\sigma_p^2}\right) \cdot \\ & \exp\left(\frac{-\left(x^{F} - C_i^{F}\right)^2}{2\sigma_s^2}\right) \end{aligned} \tag{6}$$

These three kernels are used to measure the color, location and semantic similarity between x and C_i, respectively. σ_c, σ_p and σ_s denote the corresponding standard deviation of color, location and semantic feature, respectively. This is derived by calculating the average of the colors, coordinates and semantic features of all palette entries.

We design this radial basis function (Eq. 6) based on the idea that the similarity between a pixel and a palette entry should be determined by a combination of their colors, positions and semantic features. From the definition, we could know that x and C_i are similar if and only if all their features are close.

We would like to constrain that $S_j(x) = 1$ if $x = C_j$, and $S_j(x) = 0$ if $x = C_{i \neq j}$. That is to say, each entry in the palette is most similar to itself (with a similarity of 1) and least similar to other entries in the palette (with a similarity of 0). Given the constraints, we can build a linear system with k^2 equations, and the coefficients $\lambda_{j,i}(j, i \in 1, 2, \cdots k)$ in Eq. 5 can be efficiently obtained by solving this linear system.

Once the similarity function of each palette entry is determined, the similarity weight $w_j^{p_i}$ could be obtained through:

$$w_j^{p_i} = \frac{S_j\left(p_i\right)}{\sum_{j=1}^{k} S_j\left(p_i\right)} \tag{7}$$

During recoloring, the semantic and location components of each palette entry are fixed, and users are allowed to modify the color components of the palette entries to adjust the appearance of the input image. When users modify a palette color, only objects or regions that are semantically close to it (belonging to the same class of objects), similar in color, and close in position will have color changes, thus achieving good local control in color editing. The process of image recoloring is presented in Algorithm 2.

4 Experiments

We perform all experiments on a desktop computer with an Intel i7-11700 2.5GHz CPU and 16 GB RAM. Our algorithm is implemented in C++ language.

Algorithm 2 Image Recoloring

Input:
an image I, the palette C, the modified palette C'

Output:
the recolored image I'

1: solve for the coefficient $\lambda_{j,i}$ of the similarity function (Eq. 5)
2: determine the similarity weights of each pixel to the palette entries (Eq. 7)
3: generate the recolored image (Eq. 4)
4: return the recolored image I'

Fig. 3. Our recoloring GUI. Which presents the original and modified color palettes on the left, and the original and recolored images on the right. The numbers below the palette colors correspond to objects or regions in the original image. The color picker (lower left) is used to change the palette colors.

We employ the source code provided by Aksoyet al. [2] to extract per-pixel semantic features from an input image. Besides, we developed a GUI for interactive recoloring using Qt as shown in Fig. 3.

4.1 Results

We generated four recoloring results with our method in Fig. 4. For each example, we provide the input image, the extracted and modified palettes (modified colors are marked with red underlines), and the recolored result. All these examples contain objects or regions with similar colors, and our method is able to modify them to different colors. For example, in Fig. 4 (a), our palette captures two similar blue colors of the sky and balloon, and we can separately alter the sky's color to cyan and change the blue regions in the balloon into purple. In Fig. 4 (c), our method extracts the dominant colors of multiple yellow objects, and successfully recolored them into different colors. Note that while there may be several entries with very similar or identical colors in the palette (e.g., the palette below the input image in Fig. 4 (c)). This is because each entry in the palette is

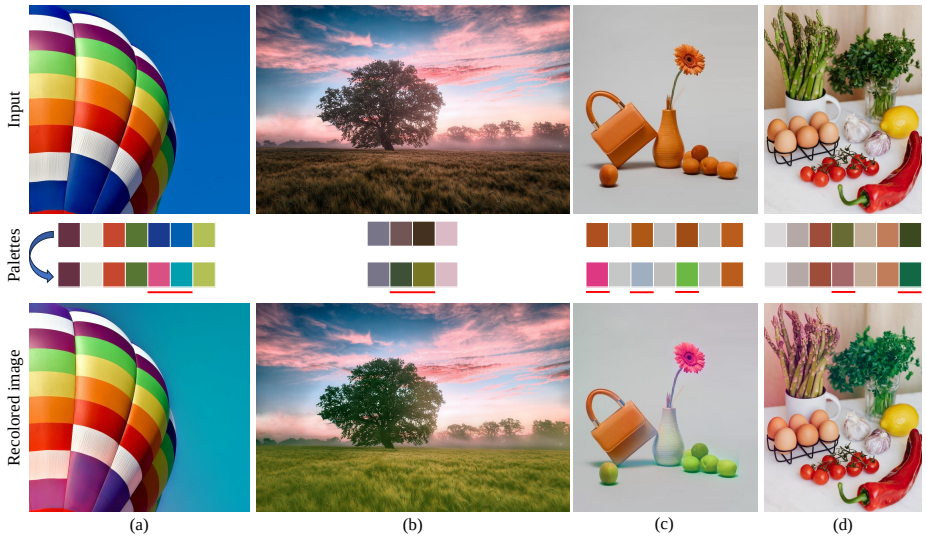


Fig. 4. Recoloring results generated by our method.

in fact an 8D vector, and the semantic and location components of these entries may be quite different.

4.2 Evaluation

Evaluation of weight parameters in distance measurement. Here, we evaluate the weighting parameters of the distance metric function (Eq. 3) in the K-means clustering, including the weight of the color term θ_c, the weight of the location term θ_p and the weight of the semantic term θ_s. We give an example in Fig. 5, and provide the extracted palettes and corresponding weight maps with different parameter settings. The results generated with the default parameters ($\theta_c = 1.0, \theta_p = 0.2, \theta_s = 5.0$) are presented in the first row. Followed by the results generated by adjusting θ_c, θ_p and θ_s, respectively. When evaluating one parameter, others are fixed to default values.

Here, a weight map w_i can be viewed as a single-channel grayscale image of the same size as the input image I and is associated with a palette entry C_i. Where any pixel's color $w_i^p \in [0, 1]$ equals the weight of I_p to C_i (Eq. 7).

From the definition of the color transfer function (Eq. 4), the weight map actually denotes the regions affected by the corresponding palette entry during recoloring. In theory, we expect each weight map to contain only a single object with similar semantics and colors, so that this region can be accurately edited by modifying the corresponding palette color.

For θ_c, a small value usually causes the palette entry to affect a larger region containing pixels with quite different colors (i.e., the 2nd weight map in row 2 contains both yellow hair and red skirt), while a larger value leads to the region contains multiple semantically different objects (i.e., the 3rd weight map in row

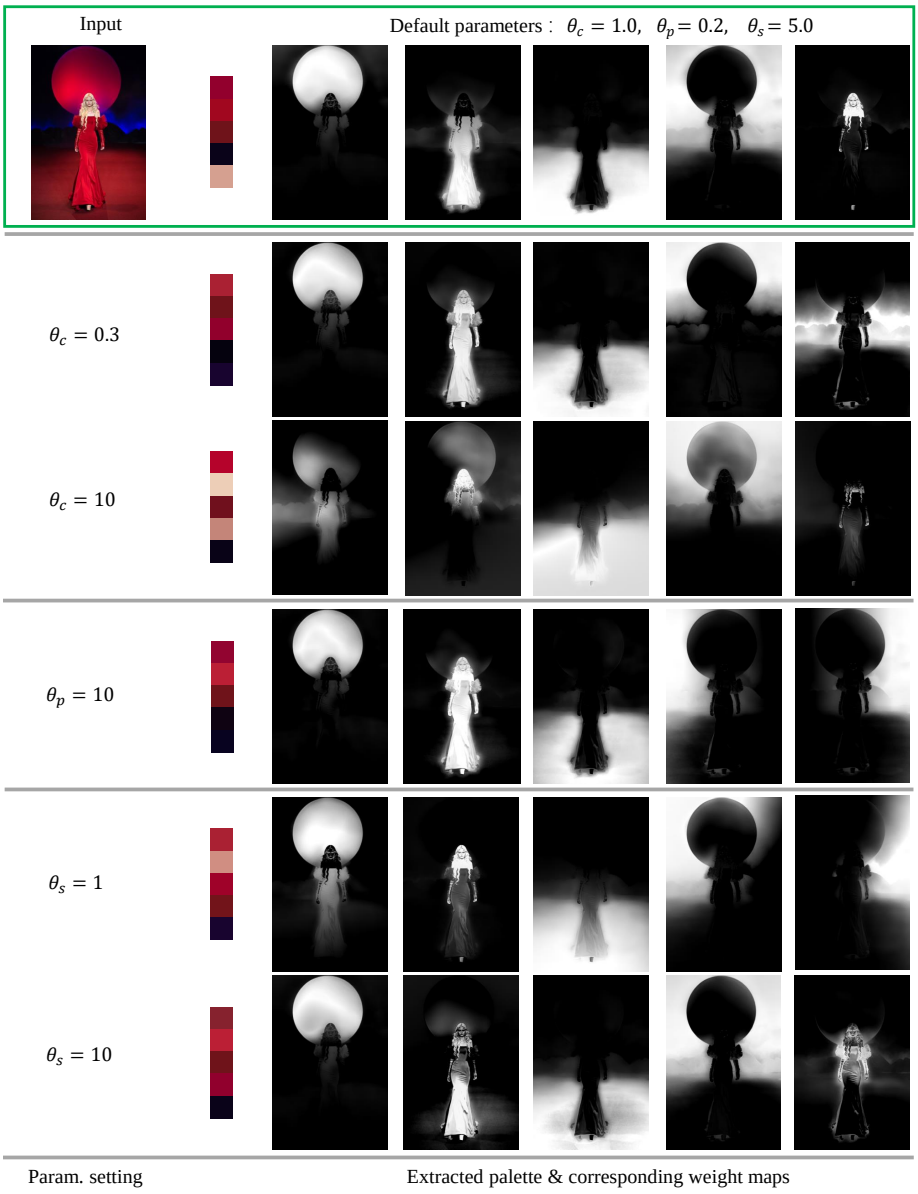


Fig. 5. Parameter evaluation.

3 contains both the red skirt and the red carpet). For θ_p, a larger value leads to the same object being scattered across multiple weight maps, thus it will be affected by multiple palette colors during recoloring (i.e., the background is divided into two parts in the last two weight maps in row 4). For θ_s, a small value leads to the region containing different objects (i.e., the 3rd weight map

in 5 contains both the red skirt and the red carpet), while a larger value usually leads to better results.

In summary, the default values of these parameters could generate the most satisfactory palettes and weight maps.

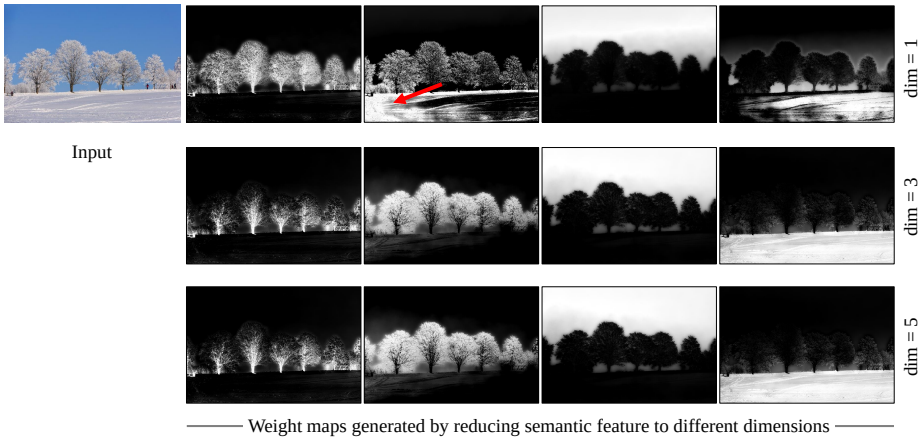


Fig. 6. Evaluation of the semantic feature dimension.

Evaluation of the dimension of semantic features. Here we reduce per-pixel's semantic feature from 128D to different dimensions (dim = 1, 3, 5), and combine the color and location information to build a high-dimensional feature for each pixel, and finally perform K-means to extract the palette. In Fig. 6, we expect to extract a palette consisting of 4 entries that correspond to the sky, trunks and branches of the trees, and the earth. We provide weight maps generated using semantic features of different dimensions. It can be seen that using lower dimensional semantic feature cannot accurately distinguish between different objects (e.g., the 2nd weight maps in rows 1). In contrast, using higher dimensional semantic feature yield more accurate distinctions. In this example, it yields improved results when the dimension of the semantic feature is 3 or higher. To reduce storage overhead and effectively differentiate the semantics of various objects, we use the 3D semantic feature along with the 3D color and 2D location as the feature of each pixel.

4.3 Comparisons

Comparison of palettes and weight maps. We first compare the palettes and corresponding weight maps generated by Chang et al. [4] (1st row), Tan et al. [23] (2nd row), Wang et al. [27] (3rd row) and our method (4th row) in Fig. 7. In this example, the banana and background share a similar yellow color. Existing methods fail to separate them in weight maps. For example, in the 1st weight map in Chang et al. [4], in the 4th weight map in Tan et al. [23], and in the 4th weight map in Wang et al. [27]), both the banana and

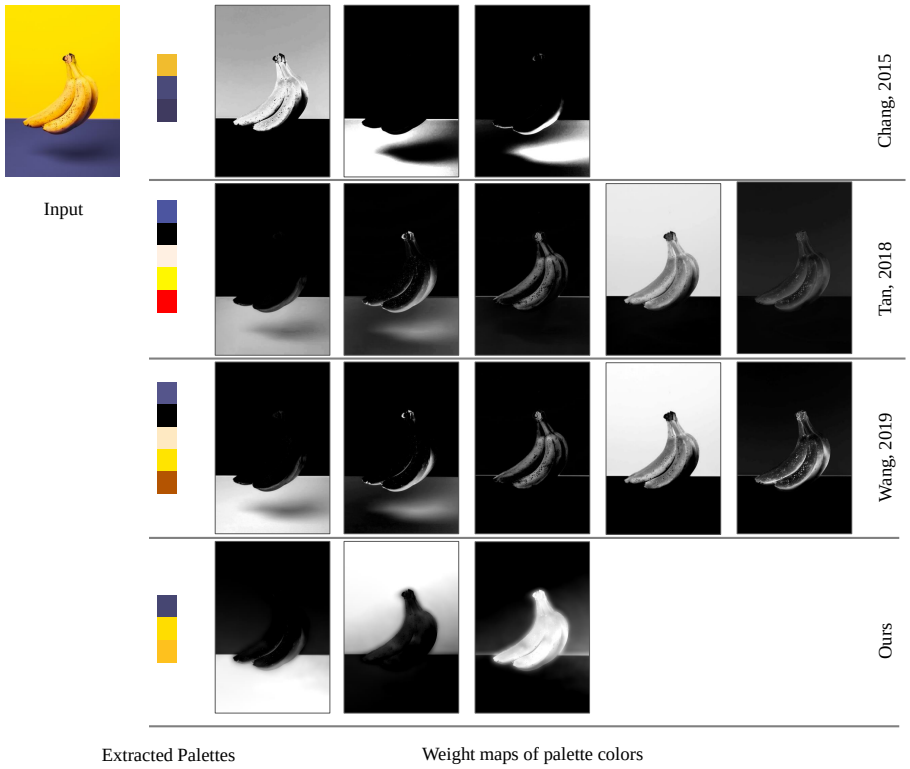


Fig. 7. Comparison of color palettes generated by different algorithms.

upper background are included in the same weight map. This means users can't edit them individually. In contrast, our palette captures three primary objects much better: the banana's yellow, the upper background's yellow, and the lower background's blue, are presented in three separate weight maps. This allows users to edit the colors of these objects separately during recoloring without affecting each other.

Comparison of recoloring results In Fig. 8 and Fig. 9, we show more recoloring examples to compare our method against approaches proposed by Chang et al. [4], Tan et al. [23], Wang et al. [27] and Chao et al. [5]. For each example, we give the input, two editing intends (red and blue texts below the input), the extracted palettes (1st row), the recoloring results generated by different methods, and the modified palettes (below the recolored images). The changed palette colors are marked with red underlines.

All provided examples contain at least two objects with similar colors. It is challenging for existing palette-based methods to recolor these objects individually. While our method could effectively capture the dominant features (semantics + color + location) of different objects with similar colors, which

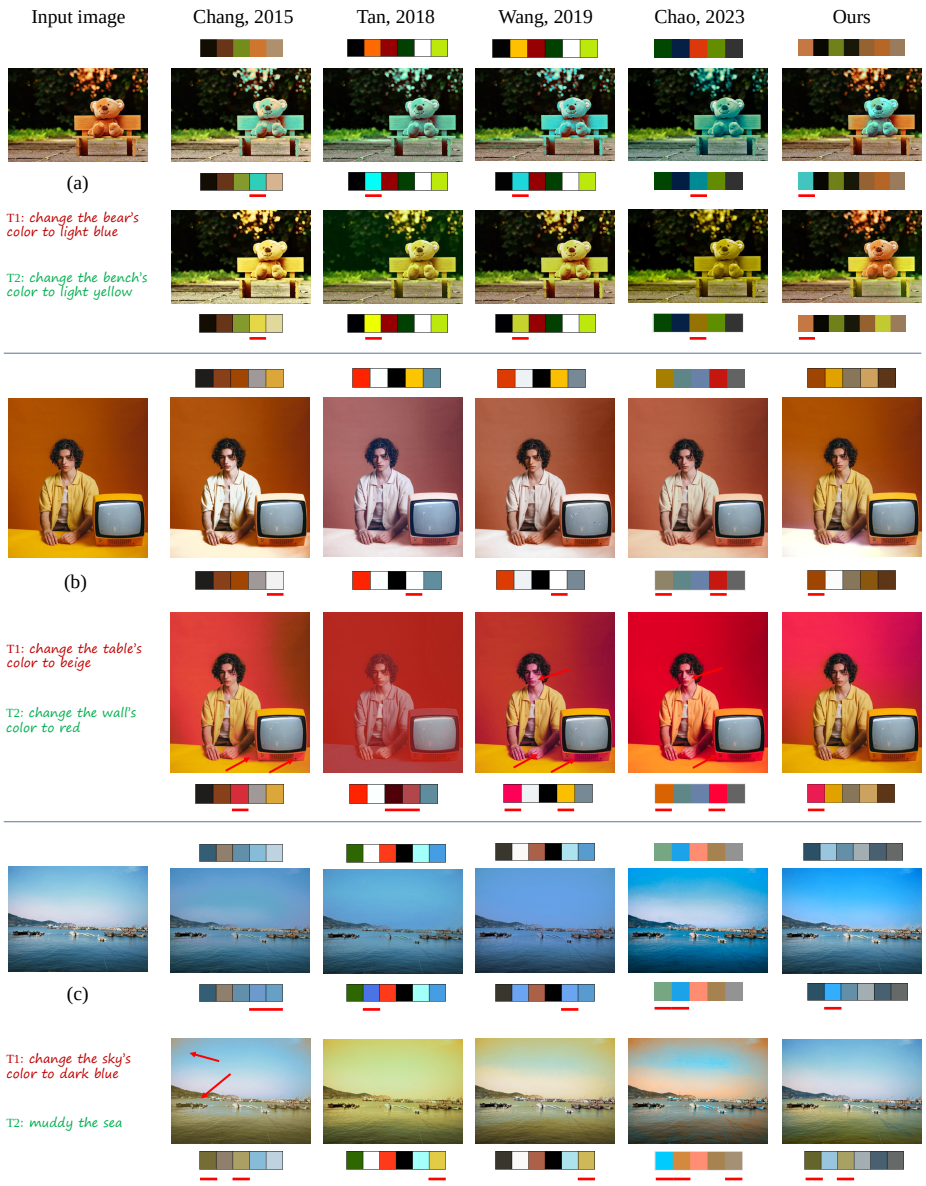


Fig. 8. Comparison of recoloring results of different methods.

thus generates better recoloring results. For example, in Fig. 8 (a), the teddy bear, the chair, and the upper light share similar yellow colors. The two edit intentions given are 1) to change the bear's color to light blue and 2) to change the bench's color to light yellow. Existing methods struggle to recolor them separately without affecting each other, our method accurately recolors them to different colors, aligning well with the user's intent. In Fig. 8 (b), the man's shirts, the TV, and the table have similar colors. When changing the color of a table to beige, existing methods make all these objects' colors unexpectedly beige. Our method successfully adjusts the color of the table without affecting other non-interested objects. In Fig. 8 (c), the input image depicts a beautiful view of the sea melted into the sky. Users want to 1) make the sky's color clear blue and 2) muddy the sea. Existing methods cannot distinguish the sky and the sea effectively, when modifying the sky, the color of the sea changes accordingly and vice versa. Our method better fulfills the user's intention.

Comparisons show that our approach has natural advantages in content-aware recoloring. This is primarily due to the palettes extracted from the high-dimensional feature space contain semantic information. As a result, each palette entry corresponds to a specific object or region. This makes content-aware image recoloring possible.

5 Conclusion, Limitation and Future work

In this paper, we have presented a novel palette-based approach that enables content-aware image recoloring for the first time. To achieve this goal, we first extract the palette of an image in a high-dimensional feature space which fuses the semantic, color and spatial information. This makes our color palette contain semantic information and enables each palette entry to correspond to a particular object or region. We then transfer the changes of palette colors to the input image through a color transfer function. Our method is simple and easy to implement, making it useful for users to perform content-aware local editing. Extensive experiments have demonstrated the effectiveness of our method.

Our method still has two primary limitations to be further improved. First, currently, the palette size k needs to be specified by the user. A color palette with numerous hues might make it challenging to semantically distinguish between different objects, while a limited palette size may result in the inability to accurately represent the colors of certain objects. Therefore, determining the appropriate size of the color palette may require users to invest some time in experimentation. We provide such a failure case in Fig. 10. In the future, it is desirable to explore ways to automatically determine the palette size, by considering both color diversity and semantic complexity of the input image. Second, the accuracy of our palette extraction algorithm builds on top of a semantic feature detection neural network while it is not always accurate, especially for complex scenes. In the future, we would like to explore adaptive palette extraction, and detect semantic features with a more advanced neural network to make the palette extraction and recoloring more robust.

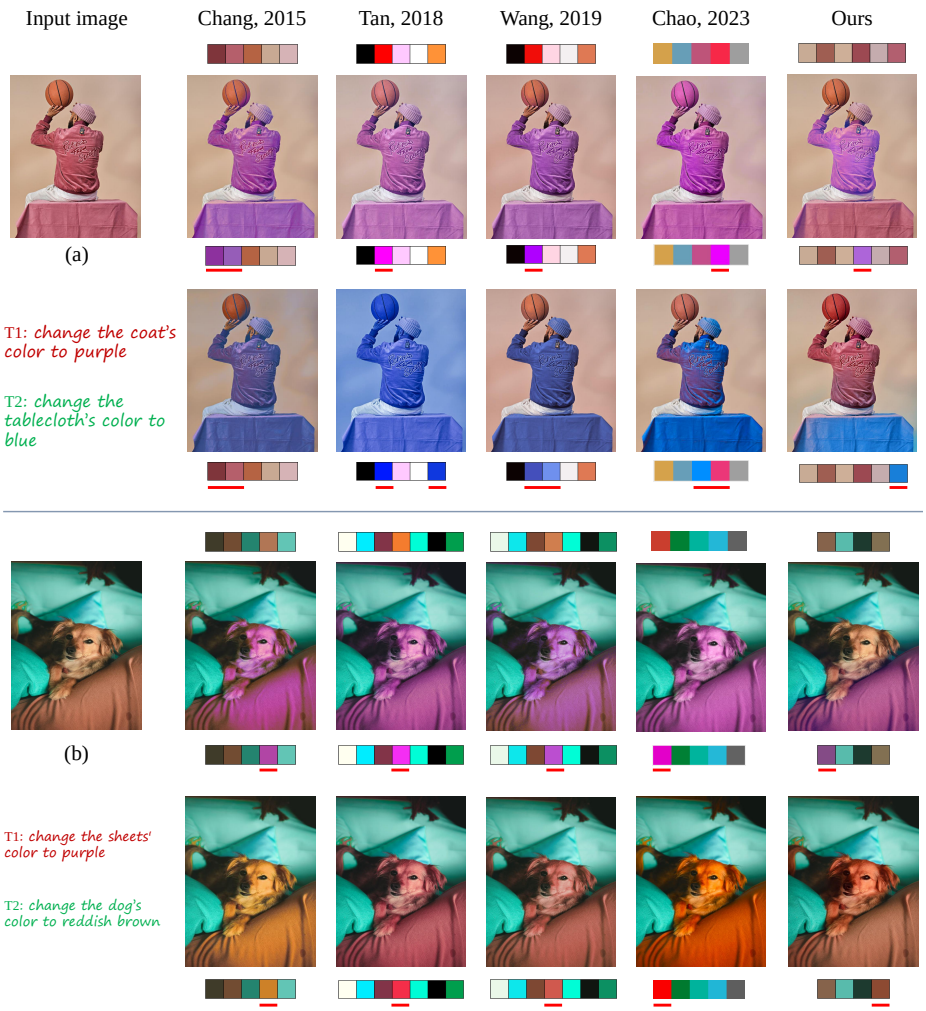


Fig. 9. Comparison of recoloring results of different methods.

6 Acknowledgements

We are grateful to the anonymous reviewers for their insightful comments and constructive suggestions, which have significantly contributed to the improvement of our manuscript. This work was supported the Youth Program of Natural Science Foundation of Qinghai Province (Project Number: 2023-ZJ-951Q).

References

1. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: Slic superpixels compared to state-of-the-art superpixel methods. IEEE transactions on pattern analysis and machine intelligence **34**(11), 2274–2282 (2012)

(a) Input & palette (b) Recolored image & edited palette

Fig. 10. Failure case. In this example, the palette contains only a limited number of six entries, some objects or regions are not captured with the K-means clustering. As a result, the girl's face, hair, and the text on the door cannot be semantically distinguished. When we recolor the cloth and the wall with different colors, the colors of other objects are changed accordingly.

2. Aksoy, Y., Oh, T.H., Paris, S., Pollefeys, M., Matusik, W.: Semantic soft segmentation. ACM Transactions on Graphics (TOG) **37**(4), 1–13 (2018)
3. An, X., Pellacini, F.: Appprop: all-pairs appearance-space edit propagation. In: ACM SIGGRAPH 2008 papers, pp. 1–9 (2008)
4. Chang, H., Fried, O., Liu, Y., DiVerdi, S., Finkelstein, A.: Palette-based photo recoloring. ACM Trans. Graph. **34**(4), 139–1 (2015)
5. Chao, C.K.T., Klein, J., Tan, J., Echevarria, J., Gingold, Y.: ColorfulCurves: Palette-aware lightness control and color editing via sparse optimization. ACM Transactions on Graphics (TOG) **42**(4) (Jul 2023). https://doi.org/10.1145/3592405, https://doi.org/10.1145/3592405
6. Chao, C.K.T., Klein, J., Tan, J., Echevarria, J., Gingold, Y.: LoCoPalettes: Local control for palette-based image editing. Computer Graphics Forum (CGF) **42**(4) (Jun 2023). https://doi.org/10.1111/cgf.14892, special issue for Eurographics Symposium on Rendering (EGSR)
7. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence **40**(4), 834–848 (2017)
8. Choi, Y., Choi, M., Kim, M., Ha, J.W., Kim, S., Choo, J.: Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8789–8797 (2018)
9. Du, Z.J., Lei, K.X., Xu, K., Tan, J., Gingold, Y.: Video recoloring via spatial-temporal geometric palettes. ACM Transactions on Graphics (TOG) **40**(4), 1–16 (2021)
10. 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, pp. 189–201. Wiley Online Library (2016)

11. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2414–2423 (2016)
12. Gui, Y., Zeng, G.: Joint learning of visual and spatial features for edit propagation from a single image. The Visual Computer **36**(3), 469–482 (2020)
13. He, K., Sun, J., Tang, X.: Guided image filtering. IEEE transactions on pattern analysis and machine intelligence **35**(6), 1397–1409 (2012)
14. Hoffer, E., Ailon, N.: Deep metric learning using triplet network. In: Similarity-Based Pattern Recognition: Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015. Proceedings 3. pp. 84–92. Springer (2015)
15. Holland, S.M.: Principal components analysis (pca). Department of Geology, University of Georgia, Athens, GA **30602**, 2501 (2008)
16. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. In: ACM SIGGRAPH 2004 Papers, pp. 689–694 (2004)
17. Li, Y., Ju, T., Hu, S.M.: Instant propagation of sparse edits on images and videos. In: Computer Graphics Forum. vol. 29, pp. 2049–2054. Wiley Online Library (2010)
18. Lischinski, D., Farbman, Z., Uyttendaele, M., Szeliski, R.: Interactive local adjustment of tonal values. ACM Transactions on Graphics (TOG) **25**(3), 646–653 (2006)
19. Neumann, L., Neumann, A.: Color style transfer techniques using hue, lightness and saturation histogram matching. In: CAe. pp. 111–122 (2005)
20. Pellacini, F., Lawrence, J.: Appwand: Editing measured materials using appearance-driven optimization. In: ACM SIGGRAPH 2007. p. 54–es (2007)
21. Pitie, F., Kokaram, A.C., Dahyot, R.: N-dimensional probability density function transfer and its application to color transfer. In: Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1. vol. 2, pp. 1434–1439. IEEE (2005)
22. Reinhard, E., Adhikhmin, M., Gooch, B., Shirley, P.: Color transfer between images. IEEE Computer Graphics and Applications **21**(5), 34–41 (2001). https://doi.org/10.1109/38.946629
23. Tan, J., Echevarria, J., Gingold, Y.: Efficient palette-based decomposition and recoloring of images via RGBXY-space geometry. ACM Transactions on Graphics (TOG) **37**(6), 1–10 (2018)
24. Tan, J., Echevarria, J., Gingold, Y.: Palette-based image decomposition, harmonization, and color transfer. arXiv preprint arXiv:1804.01225 (2018)
25. Tan, J., Lien, J.M., Gingold, Y.: Decomposing images into layers via RGB-space geometry. ACM Transactions on Graphics (TOG) **36**(1), 1–14 (2016)
26. Ulyanov, D., Lebedev, V., Vedaldi, A., Lempitsky, V.: Texture networks: Feed-forward synthesis of textures and stylized images. arXiv preprint arXiv:1603.03417 (2016)
27. Wang, Y., Liu, Y., Xu, K.: An improved geometric approach for palette-based image decomposition and recoloring. In: Computer Graphics Forum. vol. 38, pp. 11–22. Wiley Online Library (2019)
28. Xu, K., Li, Y., Ju, T., Hu, S.M., Liu, T.Q.: Efficient affinity-based edit propagation using kd tree. ACM Transactions on Graphics (ToG) **28**(5), 1–6 (2009)
29. Zhang, Q., Xiao, C., Sun, H., Tang, F.: Palette-based image recoloring using color decomposition optimization. IEEE Transactions on Image Processing **26**(4), 1952–1964 (2017)
30. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of the IEEE international conference on computer vision. pp. 2223–2232 (2017)