Noname manuscript No.
(will be inserted by the editor)

Scribble-based Gradient Mesh Recoloring

**Liang Wan · Yi Xiao · Ning Dou ·
Chi-Sing Leung · Yu-Kun Lai**

Received: date / Accepted: date

Abstract Previous gradient mesh recoloring methods usually have dependencies on an additional reference image and the rasterized gradient mesh. To circumvent such dependencies, we propose a user scribble-based recoloring method, in which users are allowed to annotate gradient meshes with a few color scribbles. Our approach builds an auxiliary mesh from gradient meshes, namely control net, by taking both colors and local color gradients at mesh points into account. We then develop an extended chrominance blending method to propagate the user specified colors over the control net. The recolored gradient mesh is finally reconstructed from the recolored control net. Experiments validate the effectiveness of our approach on multiple gradient meshes. Compared with various alternative solutions, our method has no color bleedings nor sampling artifacts, and can achieve fast performance.

Keywords Gradient mesh · Recoloring by scribble · Control net · Scribble point set

L. Wan
Tianjin University
E-mail: lwan@tju.edu.cn

Y. Xiao (corresponding author)
Hunan University
E-mail: yixiao_csee@hnu.edu.cn

N. Dou
Tianjin University

C.-S. Leung
City University of Hong Kong

Y.-K. Lai
Cardiff University

1 Introduction

Graphics community has evidenced a trend of creating *photorealistic* vector arts in recent years. To represent multicolored objects with smooth color transition, many vector graphics artists (e.g. Morisaki [1], Forrest [2] and Highside [3]) adopt gradient meshes, a powerful vector graphics representation available in commercial softwares, such as Illustrator and Coreldraw. Gradient mesh is a regular 2D grid with attributes including position, color, and their gradients specified at each mesh point. To create gradient meshes, artists have to manually specify mesh grids and manipulate the associated attributes. This process requires skills and can be highly labor intensive. Given a pre-created gradient mesh, changing its color appearance may still involve a lot of selection and color assignment operations, particularly when the smooth color transition is to be maintained. For instance, manually recoloring the 11×10 gradient mesh (Fig. 1) may take an artist more than ten minutes to accomplish. The recoloring editing becomes more demanding when the vector art consists of multiple gradient meshes, or the gradient mesh has a large size.

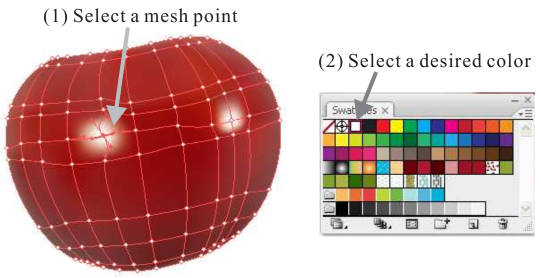


Fig. 1 Recoloring gradient meshes typically involves selecting grid points and adjusting the associated colors and their gradients.

In the literature, several methods have been developed to handle the gradient mesh recoloring problem. The first attempt was reported by Xiao et al. [6], which changes the color appearance of a gradient mesh by borrowing the color statistics of a reference image. This method was later extended to an optimization-based scheme [7]. Both works use the rasterized gradient meshes for calculating color statistics of gradient meshes. However, similar to other example-based color transfer works, it can be difficult for users to prepare a good reference image which contains desirable colors.

To circumvent dependencies on an additional reference image and the rasterized gradient mesh, we propose a user scribble-based method for gradient mesh recoloring. Our method allows users to indicate the target color distribution by directly drawing scribbles in desired colors over gradient meshes. To the best of our knowledge, no previous methods for gradient mesh recoloring rely on user scribbles only. In order to propagate color distribution yet respecting local color gradients in gradient meshes, we propose an auxiliary mesh, named as *control net*. The control net embeds local color gradients by constructing

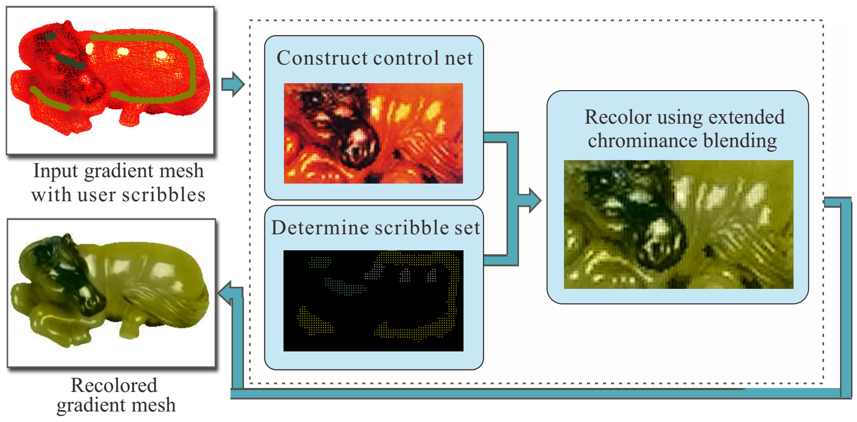

Input gradient mesh with user scribbles
Construct control net
Determine scribble set
Recolor using extended chrominance blending
Recolored gradient mesh

Fig. 2 Framework of our scribble-based gradient mesh recoloring.

auxiliary net points for each mesh patch. The indicated color scribbles are then propagated over the control net with special attentions paid to the topological structure of gradient meshes, which may contain image holes. The recolored gradient mesh is finally reconstructed from the recolored control net. Fig. 2 illustrates the framework of our method. In the experiments, we compare our method with several alternative solutions. As the experimental results show, the proposed method can produce pleasant recolored results.

2 Related Work

In the literature, Sun et al. [4] and Lai et al. [5] developed semi-automatic and automatic methods for generating gradient meshes from a raster image. Lai et al. [5] extended the topology property of gradient meshes to allow an arbitrary number of holes in image objects. With these generation methods, a straightforward way to do gradient mesh recoloring is to recolor rasterized images and then regenerate gradient meshes. However, this process is not only time consuming, but may also change the topological structure of gradient meshes or cause obvious color bleedings and sampling artifacts.

The work by Xiao et al. [6] is the first attempt to address gradient mesh recoloring. They borrow the idea from example-based image color transfer [8, 13–20], which takes the color distribution of another image as reference. Specifically, they calculated the color statistics of gradient meshes and the example image, and adopted a PCA-based color transfer to update the color and local color gradients of mesh points. They later extended the PCA-based recoloring to an optimization-based scheme [7]. It aims to minimize the differences of the color distribution of the example image and the transferred gradient mesh. By adding image gradients as a constraint, they got an optimal linear transfer function which can preserve the structure details of the gradient mesh. These two works both need an example image and have to do calculation on the rasterized gradient meshes.

In our work, we borrow the idea from scribble-based image color transfer [21–29], which allows the user to draw strokes in desired colors and diffuse the colors across the image. By this way, our method has no dependence on an example image and the rasterized gradient meshes.

In the following, we briefly review scribble-based color transfer methods. Levin et al. [21] proposed a global optimization method that minimizes a quadratic cost function derived from the color difference between a pixel and the weighted average of its neighbors. For images with complex textures, this method may require a very large number of scribbles to achieve high quality colorization. To address this problem, different methods [30,25,31] have been proposed to generate scribbles automatically. Sapiro [32] colorized a grayscale image constrained by image gradients and color scribbles that serve as boundary conditions. The resulted color is obtained via solving Poisson equations. Drew and Finlayson [26] made use of Di Zenzo gradient and adjusted color gradients to generate an appropriate contrast direction. The results have color-contrast that appears the same as the luminance-contrast of the original. Yatziv and Sapiro [24] developed a fast algorithm that colorizes grayscale pixels by blending color scribbles. The weights are proportional to geodesic distances between pixels and their corresponding scribbles in the luminance channel. Kawulok and Smolka [27] proposed a competitive approach for selecting an appropriate type of propagation path costs, and developed a modified chrominance distance computed along each path. They further considered textural features for path optimization [29].

In our work, we base our recoloring scheme on the colorization technique proposed in [24] and extend it for gradient mesh recoloring. The extension includes handling the topological structure of gradient meshes, and constructing a set of valid scribble points defined on gradient meshes from user-specified color scribbles.

3 Gradient Mesh Recoloring

3.1 Problem Formulation

Like image color transfer methods, we assume that neighboring pixels having close colors should have similar colors after recoloring. It is noted that pixel values of gradient meshes need to be computed via interpolating colors and color gradients at mesh points, as gradient meshes are defined in parametric domain. Therefore, recoloring gradient meshes involves updating both color values and color gradients at mesh points. If not considering color gradients, an $M \times N$ gradient mesh can be packed to form an $M \times N$ image. Then directly applying existing image color transfer methods on this image may result in distracting artifacts. Fig. 10(a) shows an example, in which the grid structure in the middle region is rather exaggerated.

To consider color gradients at mesh points, one straightforward approach is to sparsely discretize gradient meshes in parametric domain, and then re-estimate colors and color gradients after recoloring. However, we know that estimating color and color gradients from an image is a complicated task as pointed out in [5]. Furthermore, both the discretization and estimation steps are non-linear operations, which may cause a large computation burden.

In our work, we consider color gradients by proposing an approximate representation for gradient meshes, namely control net. For each gradient mesh patch, the control net constructs auxiliary net points according to partial derivative estimation. This representation guarantees fast computing performance since both the construction and re-estimation steps are linear operations. Also note that the control net is designed for facilitating recoloring purpose and has no practical physical meaning. Then gradient mesh recoloring is realized by 1) constructing a control net from gradient meshes, 2) performing recoloring on the control net, and 3) reconstructing gradient meshes from the recolored control net. Fig. 2 illustrates this process.

3.2 Gradient Mesh Representation

Following [4,5], a normal gradient mesh is a regular planar grid of Ferguson patches [33], each containing four control points (Fig. 3(a)). A control point has the attributes specifying the position $\{x, y\}$, color $\{r, g, b\}$, and gradients of position and color $\{\partial_u m, \partial_v m, \alpha\partial_u m, \beta\partial_v m\}$, where m is a component of either position or color, α and β are two scaling factors. Each point in a Ferguson patch, including its position and color, can be calculated via the following interpolation,

$$m(u, v) = UCQC^TV^T, \tag{1}$$

where

$$Q = \begin{bmatrix} m_0 & m_2 & \partial_v m_0 & \partial_v m_2 \\ m_1 & m_3 & \partial_v m_1 & \partial_v m_3 \\ \partial_u m_0 & \partial_u m_2 & 0 & 0 \\ \partial_u m_1 & \partial_u m_3 & 0 & 0 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix},$$

$U = [1\ u\ u^2\ u^3]$, $V = [1\ v\ v^2\ v^3]$, and $0 \le u, v \le 1$.

The gradient mesh representation is extended in [5] to allow topology holes. As illustrated in Fig. 3(b), the grid is sliced with a horizontal cut to model a hole. Each mesh point v_S within the cut is split into two vertices $\bar{v}_S$ and $\hat{v}_S$. For each end vertex, v_L for example, the two Ferguson patches adjacent to the hole have different u derivatives, and they satisfy $\partial_u \bar{m}_L = -\partial_u \hat{m}_L$ in order to ensure smoothness. A mesh with H holes can be converted to $H + 1$ normal gradient meshes without holes by reusing vertices along the cuts.

Gradient meshes can represent image objects that contain large regions of smooth color transition. In case that an image object has small or complex structures, dense mesh grid or multiple gradient meshes are usually used. As we

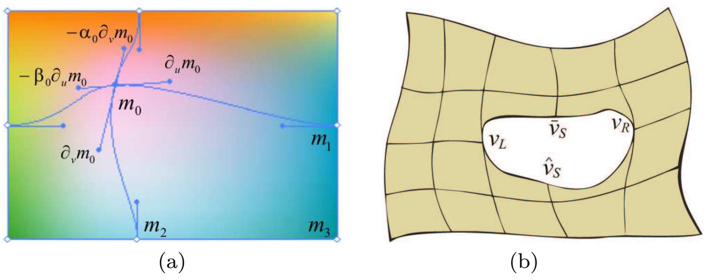


Fig. 3 Gradient mesh representation. (a) A normal gradient mesh contains four Ferguson patches. (b) A topology-preserving gradient mesh contains a hole.

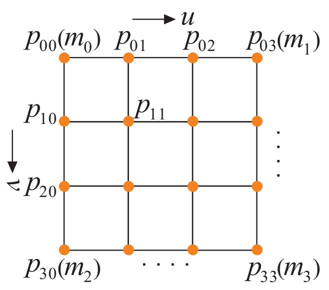


Fig. 4 Control net construction. The detailed explanation is referred to the text.

are focused on color editing while preserving the topology of gradient meshes, the position components remain intact and m is supposed to represent color.

3.3 Control Net Construction

Our control net structure is inspired by the geometric structure of bicubic Bézier patches, in which each patch is determined by 4×4 control points. We first introduce the control net constructed for a normal gradient mesh, and then discuss how to create the control net for a gradient mesh with holes.

3.3.1 Control Net for Normal Gradient Meshes

Control net construction. For a Ferguson patch, its control net contains 16 control points arranged in a 4×4 grid, as shown in Fig. 4. The control points are determined analytically in three ways:

1. The four corner points correspond to the four mesh points of a Ferguson patch, e.g., $p_{00} = m_0$.
2. The eight intermediate points on the boundaries are derived from the corner points by using the forward/backward difference formulas for computing partial derivatives. For example, the gradient along u direction for point m_0 is calculated as $\partial_u m_0 = p_{01} - p_{00}$; since $p_{00} = m_0$, we have $p_{01} = m_0 + \partial_u m_0$.

3. The four inner points are set to represent the mixed partial derivatives of the corner points. Let us take p_{11} for instance. The mixed partial derivative of m_0 is estimated via $\partial_{uv} m_0 = p_{11} - p_{01} - p_{10} + p_{00}$. Note that $\partial_{uv} m_0 = 0$ for the Ferguson patch. Hence, $p_{11} = p_{01} + p_{10} - p_{00} = m_0 + \partial_u m_0 + \partial_v m_0$.

In summary, the 16 control points in the control net have definitions as follows,

$$\begin{cases} p_{00} = m_0,\ p_{03} = m_1, p_{30} = m_2,\ p_{33} = m_3, \\ p_{01} = m_0 + \partial_u m_0, \quad\ p_{10} = m_0 + \partial_v m_0 \\ p_{02} = m_1 - \alpha_1 \partial_u m_1, p_{13} = m_1 + \partial_v m_1, \\ p_{31} = m_2 + \partial_u m_2, \quad\ p_{20} = m_2 - \beta_2 \partial_v m_2, \\ p_{32} = m_3 - \alpha_3 \partial_u m_3, p_{23} = m_3 - \beta_3 \partial_v m_3, \\ p_{11} = m_0 + \partial_u m_0 + \partial_v m_0, \\ p_{12} = m_1 - \alpha_1 \partial_u m_1 + \partial_v m_1, \\ p_{21} = m_2 + \partial_u m_2 - \beta_2 \partial_v m_2, \\ p_{22} = m_3 - \alpha_3 \partial_u m_3 - \beta_3 \partial_v m_3. \end{cases} \tag{2}$$

By simple computation, we know that the control net for an $M \times N$ gradient mesh has a grid size of $(3M - 2) \times (3N - 2)$.

Mesh reconstruction from control net. Given the control net, we can reconstruct the gradient mesh. Let $m_{i,j}$ refer to one *mesh point* in a normal gradient mesh, and $p_{3i,3j}$ denotes the corresponding *control point* in the control net. Then the color $m_{i,j}$ satisfies

$$m_{i,j} = p_{3i,3j}. \tag{3}$$

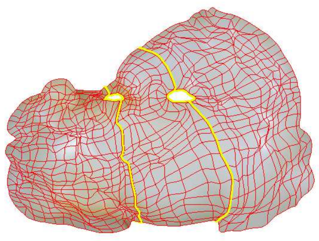
Suppose the partial derivative along u direction of $m_{i,j}$ to be $\partial_u m_{i,j}$. It meets the following conditions,

$$\begin{cases} p_{3i+1,3j} = p_{3i,3j} + \partial_u m_{i,j}, \\ p_{3i-1,3j} = p_{3i,3j} - \alpha_{i,j} \partial_u m_{i,j}. \end{cases} \tag{4}$$

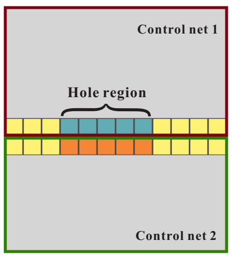
We estimate $\partial_u m_{i,j}$ via the central difference, given by

$$\partial_u m_{i,j} = \frac{p_{3i+1,3j} - p_{3i-1,3j}}{1 + \alpha_{i,j}}. \tag{5}$$

$\partial_v m_{i,j}$ is computed in a similar way. There is one thing deserving a mention: the factor α (and β) should keep its original value, although it can be estimated according to (4). This is because α and β are also used in the interpolation of mesh positions, and any change in their values may modify the shape of gradient meshes.



(a)



(b)

Fig. 5 Control net for topology-preserving gradient meshes. (a) An example of topology-preserving gradient meshes. (b) An illustrative concatenated control net containing one hole.

3.3.2 Control Net for Topology-Preserving Gradient Meshes

Recall that topology-preserving gradient meshes contain one or more holes (as shown in Fig. 5(a)). To construct the control net, we first decompose the gradient mesh with holes apart by reusing joint mesh points along the horizontal cuts (marked as yellow lines in Fig. 5(a)). Then, a control net is built for each decomposed gradient mesh. It is obvious that the decomposed normal gradient meshes and their control nets share the same width. As a result, we can simply concatenate these control nets to form a complete one (as illustrated in Fig. 5(b)).

The recoloring is applied to the concatenated control net. To reconstruct gradient meshes after recoloring, we separate the concatenated control net along the horizontal cuts and reconstruct each gradient mesh individually. During the final composition, the color values and gradients at the joint mesh points are averaged in order to maintain the smoothness.

3.4 Recoloring Using Extended Chrominance Blending

Considering both visual quality and speed performance, we adapt the fast colorization approach proposed in [24], which is based on weighted chrominance blending. Let $\mathbf{R}$ denote the set of points in the control net, a and b be two points in $\mathbf{R}$. Considering chrominance changes, we modify the geodesic distance function between a and b, which is originally based on luminance, to the following formula

$$d(a,b) = \min_{\Gamma(s)} \int_0^{l(\Gamma)} |\nabla Cb \cdot \Gamma(s)| + |\nabla Cr \cdot \Gamma(s)| ds, \tag{6}$$

where Cb and Cr denote chrominance values in $YCbCr$ color space, and $\Gamma(s)$ denotes a curve in $\mathbf{R}$ connecting a and b, parameterized by its arc length $s \in$

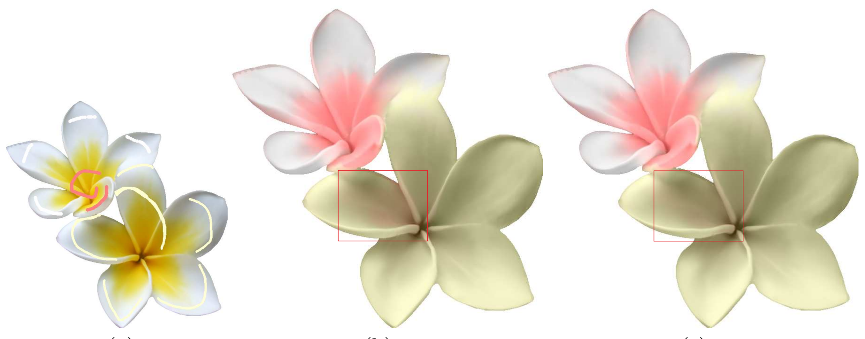


Fig. 6 Recoloring using chrominance blending: (a) input gradient mesh with user scribbles; (b) the result by using the geodesic distance defined by Eq. (6); (c) the result by using the modified geodesic distance defined by Eq. (8).

$[0, l(\Gamma)]$. This geodesic distance integrates Cb and Cr chrominance gradients along the curve $\Gamma(s)$.

Next, we compute the intrinsic distance $d_c(b)$ as the minimum distance from b to any point a with a certain scribble chrominance c. For an arbitrary point t to be recolored, its chrominance is computed as a weighted blending of chrominance of color scribbles, given by

$$chrom(t) = \frac{\sum_c w(d_c(t))c}{\sum_c w(d_c(t))}, \tag{7}$$

where the weighting function is set as $w(r) = r^{-\epsilon}$ [24]. The weights can be efficiently computed via Dijkstra algorithm, and ϵ is empirically set as $\epsilon = 1$. The recoloring computation has an average time complexity of $O(|\Omega| \cdot |\text{chrominances}(\Omega_c)|)^4$. In case that the mesh is very big, we can adopt the relaxation in [24] to reduce the time complexity to $O(|\Omega|)$, which blends the two or three most significant chrominance (the chrominance with the closest intrinsic distance).

For gradient meshes with holes, we add a stopping constraint on neighboring points when computing the geodesic distance over the control net. To be specific, we demand two neighboring points, which are located on each of the horizontal boundaries of a hole (except the two end points), to disconnect from each other (refer to Fig. 5(b)). Let $\mathbf{P}$ denote the set of such neighboring points. The modified geodesic distance between a and b is defined as

$$\bar{d}(a,b) = \begin{cases} \infty, & \text{if } (\Gamma(s_1), \Gamma(s_1 + ds)) \in \mathbf{P}, \\ d(a,b), & \text{others.} \end{cases} \tag{8}$$

The modified geodesic distance is used for the computation of blending weights. Fig. 6 compares recoloring results by using two distances defined by Eq. (6) and Eq. (8). We can see that without the stopping constraint, the red scribble propagates across the hole more easily (Fig. 6(b)).

3.5 Scribble Point Set Determination

As shown in Fig. 6(a), color scribbles are specified over the rasterized gradient mesh image. It is necessary to determine a proper set of mesh points to reflect color scribbles (denoted as Ω_c). The simplest way is to detect which mesh points are right on the color scribbles. However, in practice a color scribble may pass across mesh patches (i.e. Ferguson patches) without or just partially going through mesh points. This simple scheme may miss certain color scribbles (see Fig. 7(d)).

Here, we develop a heuristic method to decide Ω_c. Given a color scribble, we can find which patches it passes through, named as *scribble patches*. In a scribble patch, one mesh point is selected as a scribble point if one of the following conditions is satisfied,

1. The mesh point is on the color scribble;
2. One of its two associated mesh curves intersects with the color scribble;
3. The nearest distance between the mesh point and the color scribble is below a user-defined threshold L_τ.

Without loss of generality, the threshold L_τ is set as a fraction of the longest boundary length of the current scribble patch, given by

$$L_\tau = \rho \max_t L_t, \tag{9}$$

where $\{L_t | t = 0, 1, 2, 3\}$ denotes four boundaries of the current scribble patch. In our experiments, we choose the default value $\rho = 0.3$. Fig. 7(c) shows the scribble point set generated by this way, and Fig. 7(e) is the recoloring result, which is more reasonable than Fig. 7(d).

4 Experiments and Discussion

In our experiments, we use gradient mesh vector graphics developed using the methods of [5,6], which contain normal gradient meshes as well as topology-preserving gradient meshes with holes. Since there are no previous works on scribble-based gradient mesh recoloring, we make comparison with four possible recoloring solutions to evaluate our approach: 1) two alternatives of recoloring on rasterized images; 2) recoloring mesh points only; 3) applying Poisson-based recoloring on the control net. In the end, we investigate the impact of coefficients on the algorithm, and report time performance.

4.1 Comparison with Recoloring on Rasterized Images

As pointed out in [6], a naive approach to recolor a gradient mesh is recoloring on rasterized images, followed by regenerating gradient meshes. This approach is not only expensive, but may also change the topological structure of gradient meshes. In addition, recoloring on rasterized images may lead to color bleedings

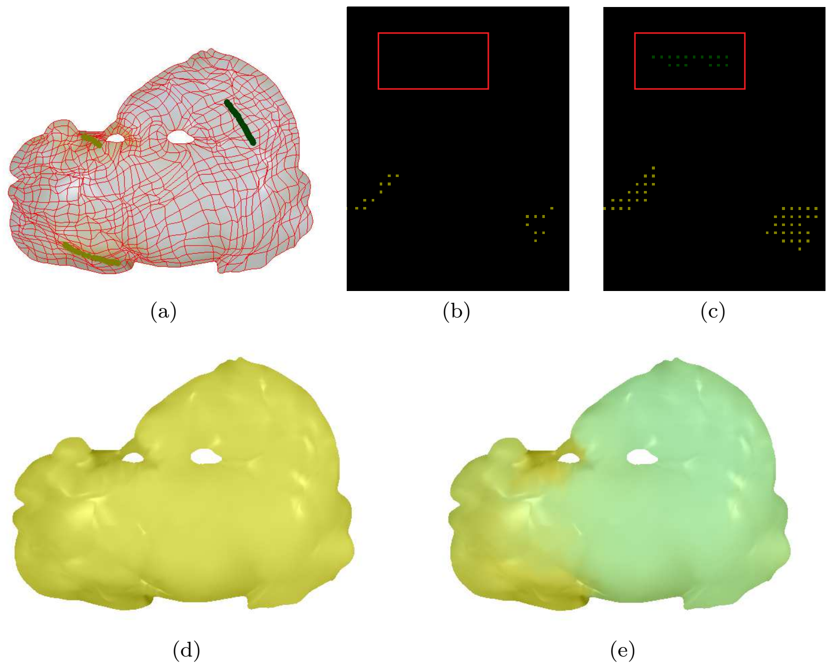


Fig. 7 Scribble point set determination. (a) User scribbles on gradient mesh. (b) Ω_c containing the control points on the scribbles only. (c) Ω_c determined by our method. Here, Ω_c is visualized over the control net domain. (d) The recoloring result using Ω_c in (b). (e) The recoloring result using Ω_c in (c).

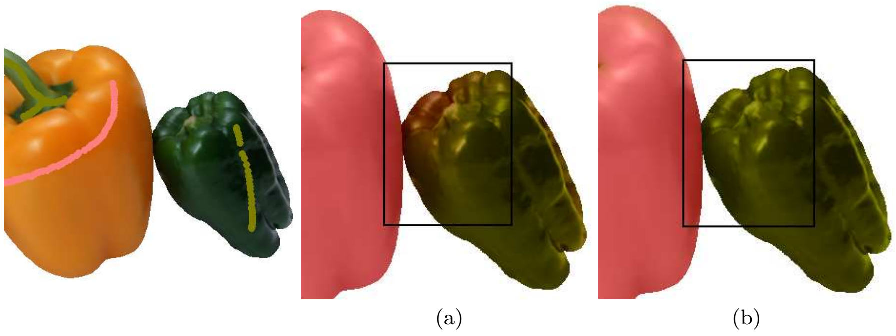


Fig. 8 Recoloring on the rasterized image of multiple gradient meshes in (a) v.s. our method in (b).

in boundary regions or overlapped regions of adjacent image objects. One example is shown in Fig. 8(a), in which the pink-orange color marked on the left pepper also affects the color appearance of the right pepper.

Another possible way is to recolor rasterized images first, then update gradient meshes by re-sampling color information from the recolored images. However, in comparison with our result in Fig. 9(d), this method may cause

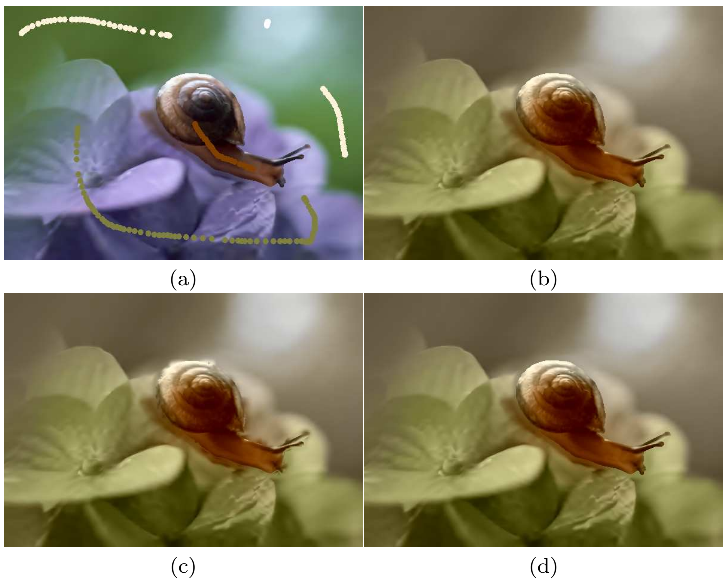

(a) (b) (c) (d)

Fig. 9 Recoloring on a rasterized image (b) followed by re-sampling color information for gradient meshes results in (c), which has obvious artifacts around the snail's back. Our gradient mesh recoloring (d) has no such artifacts.

obvious artifacts around the snail's back in Fig. 9(c). Note that the color bleedings and sampling artifacts are inevitable when recoloring the rasterized image. In other words, similar artifacts can still occur even more sophisticated image colorization methods are adapted in these two possible schemes.

4.2 Comparison with Recoloring Mesh Points Only

As mentioned before, packing mesh points can form a 2D image. This implies that a possible recoloring alternative is directly applying our extended chrominance blending on the naively packed image, and reuse the original color gradients. With experiments, we find this alternative generates results visually similar to our method. However, when recoloring a gradient image with more substantial color changes (such as Fig. 10(a)), it may generate weird color transitions. As shown in Fig. 10(b), we can easily see stripe-like artifacts in the middle region. It is because the original color gradients reflect the smooth color transition in Fig. 10(a). When the color appearance changes greatly, the original color gradients may not be applicable.

4.3 Comparison with Poisson-based Recoloring

In this section, we evaluate another possible alternative by extending Poisson-based image color transfer [32]. User scribbles serve as boundary conditions

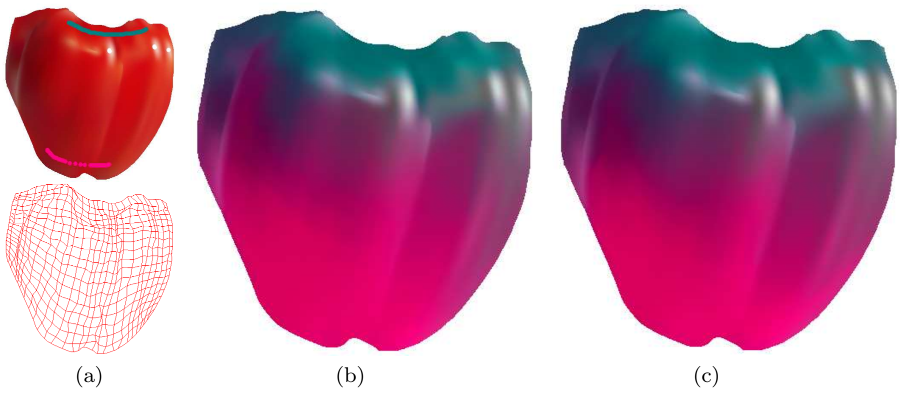


Fig. 10 Recoloring mesh points only in (b) v.s. our method in (c).

and the recoloring of the control net is obtained via solving Poisson equations. It it noted that when using this scheme to process gradient meshes with holes, constraints on hole boundaries have to be taken into consideration.

For simplicity, we conduct a comparative experiment between Poisson-based recoloring and our approach on gradient meshes without holes. Fig. 11 demonstrates results. We can see that Poisson-based recoloring generates recoloring results similar to ours, yet with distracting color bleedings on some petals. Besides, Poisson-based recoloring takes a longer computing time. For the example in Fig. 11 with a mesh size of 52×63, our chrominance blending uses 0.09 seconds, and Poisson-based recoloring spends 1.49 seconds.

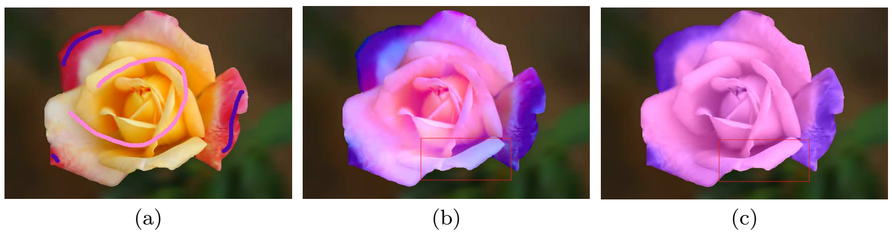


Fig. 11 Poisson-based recoloring in (b) v.s. our method in (c).

4.4 Coefficient Evaluation

Our method is affected by two coefficients, i.e. ρ in Eq. (9) for determining scribble points on gradient meshes, and ϵ used in Eq. (7) for weighting color scribbles in chrominance blending. We find varying coefficient values lead to slightly different color appearances. Generally speaking, a larger ρ makes color scribbles label more mesh points in gradient meshes, for example, the pink

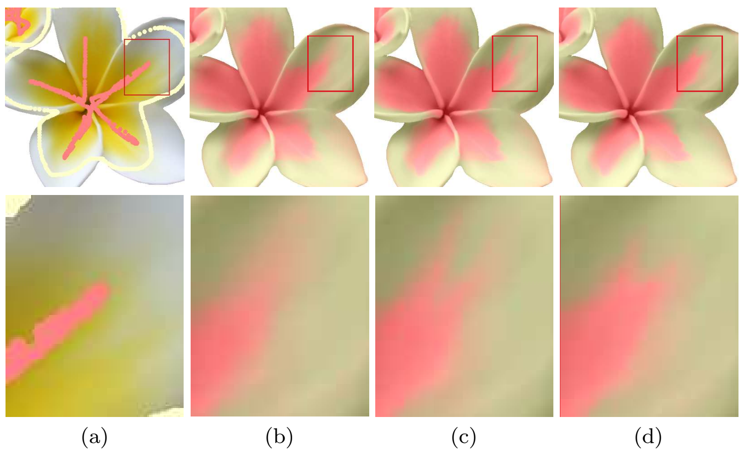


Fig. 12 ρ evaluation. Top row displays the source gradient mesh with user scribbles and result images with $\rho = 0.1, 0.5, 0.9$. Bottom row show the blowups.

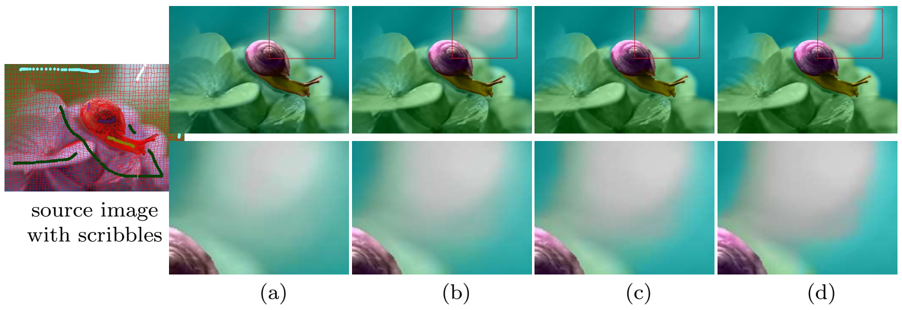


Fig. 13 ϵ evaluation. From (a) to (d) displays the recolored result with $\epsilon = 1, 2, 3, 5$.

stroke in the boxed region in Fig. 12. In terms of visual appearance, a small ρ generates smoother color transitions, while a large ρ may lead to sharper changes, as shown in Fig. 12(d). The choice of color blending coefficient ϵ changes the weights from different scribble colors, and consequently influences the smoothness of color transition. As Fig. 13 shows, the larger ϵ is, the stronger the colors blend. Here, we adopt $\rho = 0.3$ and $\epsilon = 1$ as the default values in our other experiments, as they generate pleasant visual effects along with relatively smoother color transitions.

4.5 Discussion

4.5.1 Control Net Approximation Evaluation

The control net representation is indeed an approximation for gradient meshes. To evaluate the approximation accuracy, we compute PSNR and SSIM values by comparing the rasterized images from the original gradient mesh and it-

Table 1 Control Net Approximation Evaluation

Figure	Mesh Size	Pixel Number	PSNR	SSIM
Fig. 2	44 × 78	35,265	68.5	0.9998
Fig. 6	37 × 62	226,995	67.7	0.9985
Fig. 7(a)	28 × 37	136,285	66.8	0.9949
Fig. 10	28 × 26	135,206	68.4	0.9956
Fig. 11	52 × 63	150,406	68.7	0.9970
Fig. 14(e)	51 × 51	111,517	69.9	0.9999

Table 2 Timing Performance in our experiments. Fig. "plumeria" has 1 hole. Fig. "jade2" and "snail" both have 2 holes. Other figures do not have holes.

Name	Figure	Mesh Size	Scribble point set detection(s)	Recoloring(s)	Total(s)
plumeria	Fig. 6	37 × 62	0.336	0.074	0.420
jade2	Fig. 7(a)	28 × 37	0.078	0.063	0.141
snail	Fig. 13	37 × 66, 51 × 60	0.738	0.430	1.168
jade3	Fig. 14(b)	57 × 61	0.105	0.109	0.214
plum	Fig. 14(f)	51 × 51	0.344	0.078	0.422
plum*	Fig. 14(h)	51 × 51	0.500	0.140	0.640

s control net representation. Table 1 lists the statistics for several gradient meshes. Here, the pixel number means the number of valid pixels of rasterized gradient mesh objects regardless of canvas regions. We can see that both P-SNR and SSIM values are rather high, which indicates that the control net is a high-quality approximation of gradient meshes.

4.5.2 Timing Performance

Our system is implemented with C++ language, executed on a desktop computer equipped with a 3.00Ghz Intel Core i5-3330 CPU and 8GB memory. Fig. 14 shows more recoloring results. Timing statistics of our experiments are listed in Table 2. For each example, the average running time is computed by repeating the experiment with the same settings for four times. The running time is split into two parts: one for the scribble point set determination and the other for the recoloring. Since the control net construction is very fast and almost spends no time, we combine its time with that of the scribble point set determination. In our UI, the scribbles are drawn by moving the mouse. When the mouse moves fast, the stroke seems to break into multiple dots, while we observe this has almost no impact on the recoloring results.

From Table 2, we find that gradient meshes with larger mesh sizes tend to take longer times when detecting the scribble point set. For instance, "jade2" (mesh size of 28×37) spends 0.078 seconds while "jade3" (mesh size of 57×61) spends 0.105 seconds. The detection time also becomes large when the number of scribble strokes increases, by comparing "plum", "jade3" and "plumeria" examples. On the other hand, we find that the recoloring step is very fast for gradient meshes, no matter whether there exist holes or not.

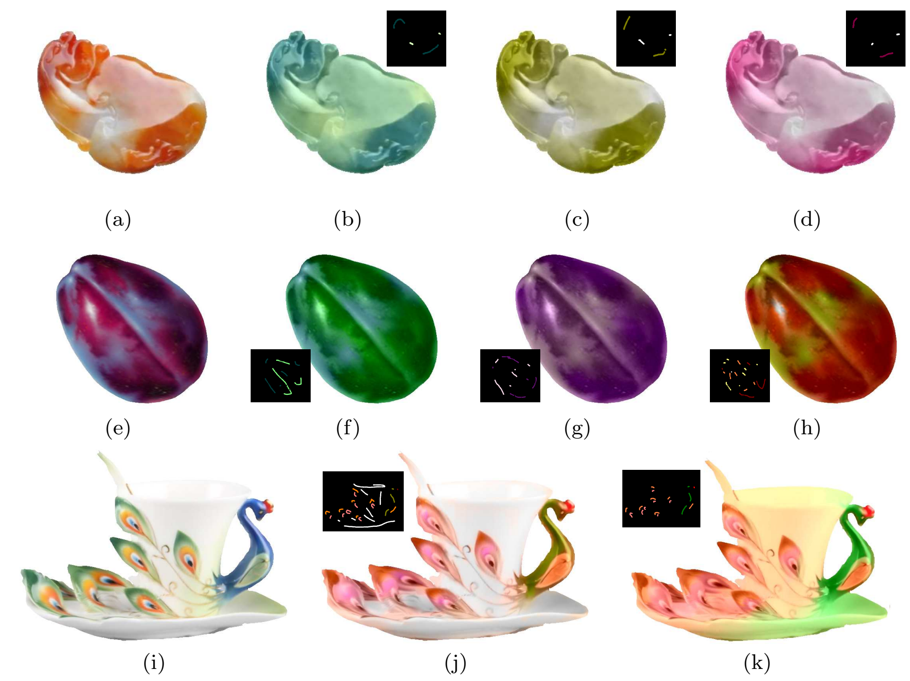


Fig. 14 More results.

4.5.3 Limitations

Our scribble point set determination assumes that one mesh point is affected by just one color scribble if it exists. In extreme cases, it is possible that color scribbles are close in position and fall in one patch or neighboring patches. In our implementation we simply choose the nearest scribble's color for the mesh point under consideration. Alternatively, we may record both colors and the distances between mesh point and color scribbles, which can be used in the chrominance blending process. Since this extreme case does not occur in our experiments, our method is sufficient to handle normal cases.

In addition, like other scribble-based methods for image color transfer, our method also depends on the distribution of color scribbles. For instance in Fig. 14(k), there are no color scribbles on the cup's body. As a result, the green color is propagated to some regions in the cup's body. Fig. 14(j) demonstrates a more pleasing recoloring result when more white color scribbles are assigned on the cup's body.

5 Conclusion

In this paper, we present a convenient and efficient recoloring method of gradient meshes by using user-specified color scribbles. Our approach is performed on mesh objects directly, without dependence on either the rasterized image of gradient meshes or an additional reference image. Our easy-to-use interactive tool allows users to annotate gradient meshes with a few color scribbles, and then automatically propagates the indicated colors while preserving the topology of gradient meshes. A control net, which considers both colors and local color gradients of mesh points, is built from gradient meshes, serving as the intermediate computing domain for the recoloring. The experimental results further demonstrate that our approach has better performance than several possible alternatives.

References

1. Takashi Morisaki. `http://www.real-trace.com/`, 2014.
2. Wayne Forrest. `http://www.wizard2.com/`, 2014.
3. Highside. `http://homepage3.nifty.com/highside/`, 2009.
4. Jian Sun, Lin Liang, Fang Wen, and Heung-Yeung Shum. Image vectorization using optimized gradient meshes. *ACM Transactions on Graphics*, 26(3):11, 2007.
5. Yu-Kun Lai, Shi-Min Hu, and Ralph R. Martin. Automatic and topology-preserving gradient mesh generation for image vectorization. *ACM Transactions on Graphics*, 28(3):1–8, 2009.
6. Yi Xiao, Liang Wan, Chi-Sing Leung, Yu-Kun Lai, and Tien-Tsin Wong. Example-based color transfer for gradient meshes. *IEEE Transactions on Multimedia*, 15(3):549–560, 2013.
7. Yi Xiao, Liang Wan, Chi-Sing Leung, Yu-Kun Lai, and Tien-Tsin Wong. Optimization-based gradient mesh colour transfer. *Computer Graphics Forum*, 34(6):123 – 134, 2015.
8. Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 21(5):34–41, 2001.
9. Baoyuan Wang, Yizhou Yu, and Ying-Qing Xu. Example-based image color and tone style enhancement. *ACM Transations on Graphics*, 30(4):64:1–64:12, July 2011.
10. Wei Zhang, Chen Cao, Shifeng Chen, Jianzhuang Liu, and Xiaoou Tang. Style transfer via image component analysis. *IEEE Transactions on Multimedia*, 15(7):1594–1601, 2013.
11. Zheng Miao, Yan Zhang, Zhibin Zheng, and Zhengxing Sun. Image palette: painting style transfer via brushstroke control synthesis. *Multimedia Tools and Applications*, pages 1–22, 2016.
12. T.V. Nguyen, Bingbing Ni, Hairong Liu, Wei Xia, Jiebo Luo, M. Kankanhalli, and Shuicheng Yan. Image re-attentionizing. *IEEE Transactions on Multimedia*, 15(8):1910 – 1919, 2013.
13. Yu-Wing Tai, Jia-Ya Jia, and Chi-Keung Tang. Local color transfer via probabilistic segmentation by expectation-maximization. In *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 747–754, Washington, DC, USA, 2005.
14. Arash Abadpour and Shohreh Kasaei. An efficient pca-based color transfer method. *Journal of Visual Communication and Image Representation*, 18(1):15–34, 2007.
15. Chung-Lin Wen, Chang-Hsi Hsieh, Bing-Yu Chen, and Ming Ouhyoung. Example-based multiple local color transfer by strokes. *Computer Graphics Forum*, 27(7):1765–1772, 2008.
16. Xiaobo An and Fabio Pellacini. User-controllable color transfer. *Computer Graphics Forum*, 29(2):263–271, 2010.
17. Zhuo Su, Daiguo Deng, Xue Yang, and Xiaonan Luo. Color transfer based on multiscale gradient-aware decomposition and color distribution mapping. In *Proceedings of the 20th ACM international conference on Multimedia*, MM '12, pages 753–756, New York, NY, USA, 2012. ACM.

18. Zhuo Su, Kun Zeng, Li Liu, Bo Li, and Xiaonan Luo. Corruptive artifacts suppression for example-based color transfer. *IEEE Transactions on Multimedia*, 16(4):988–999, 2014.
19. SangHyun Seo, YoungSub Park, and Victor Ostromoukhov. Image recoloring using linear template mapping. *Multimedia Tools and Applications*, 64(2):293–308, 2013.
20. Yue Yang, Hanli Zhao, Lihua You, Renlong Tu, Xueyi Wu, and Xiaogang Jin. Semantic portrait color transfer with internet images. *Multimedia Tools and Applications*, pages 1–19, 2015.
21. Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 23(3):689–694, 2004.
22. T. Horiuchi and H. Kotera. Colorization for monochrome image with texture. In *Color Imaging Conference*, pages 245–250, 2005.
23. Yi-Chin Huang, Yi-Shin Tung, Jun-Cheng Chen, Sung-Wen Wang, and Ja-Ling Wu. An adaptive edge detection based colorization algorithm and its applications. In *MM'05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 351–354, New York, NY, USA, 2005. ACM.
24. Liron Yatziv and Guillermo Sapiro. Fast image and video colorization using chrominance blending. *IEEE Transactions on Image Processing*, 15(5):1120–1129, 2006.
25. Qing Luan, Fang Wen, Daniel Cohen-Or, Lin Liang, Ying-Qing Xu, and Heung-Yeung Shum. Natural image colorization. In *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)*. Eurographics, June 2007.
26. Mark S Drew and Graham D Finlayson. Realistic colorization via the structure tensor. In *15th IEEE International Conference on Image Processing*, pages 457–460, 2008.
27. M. Kawulok and B. Smolka. Competitive image colorisation. In *Proceedings of 17th International Conference on Image Processing*, pages 405–408, 2010.
28. Chen Yao, Xiaokang Yang, Li Chen, and Yi Xu. Image colorization using bayesian nonlocal inference. *Journal of Electronic Imaging*, 20(2):023008–1–023008–6, 2011.
29. Michal Kawulok, Jolanta Kawulok, and Bogdan Smolka. Textural features for scribble-based image colorization. *Computer Recognition Systems 4*, 95:269–278, 2011.
30. Revital Irony, Daniel Cohen-Or, and Dani Lischinski. Colorization by example. In *Rendering Techniques*, pages 201–210, 2005.
31. Xiao-Pei Liu, Liang Wan, Ying-Ge Qu, Tien-Tsin Wong, Stephen Lin, Chi-Sing Leung, and Pheng-Ann Heng. Intrinsic colorization. *ACM Transactions on Graphics*, 27(5):1–9, 2008.
32. Guillermo Sapiro. Inpainting the colors. In *Proceedings of IEEE International Conference on Image Processing*, volume 2, pages 698–701, 2005.
33. James Ferguson. Multivariable curve interpolation. *J. ACM*, 11(2):221–228, 1964.