Deep Exemplar-based Color Transfer for 3D Model

Mohan Zhang, Jing Liao and Jinhui Yu

Abstract—Recoloring 3D models is a challenging task that often requires professional knowledge and tedious manual efforts. In this paper, we present the first deep-learning framework for exemplar-based 3D model recolor, which can automatically transfer the colors from a reference image to the 3D model texture. Our framework consists of two modules to solve two major challenges in the 3D color transfer. First, we propose a new feed-forward Color Transfer Network to achieve high-quality semantic-level color transfer by finding dense semantic correspondences between images. Second, considering 3D model constraints such as UV mapping, we design a novel 3D Texture Optimization Module which can generate a seamless and coherent texture by combining color transferred results rendered in multiple views. Experiments show that our method performs robustly and generalizes well to various kinds of models.

Index Terms—3D model texture, color transfer, deep learning

1 Introduction

Color transfer between images is a long-standing goal that seeks to transfer the colors of a reference image onto another source image. By choosing different references, one can keep the content of the source image and accurately alter the color style to emulate different illumination, weather conditions, scene materials, or even artistic color effects. Corresponding applications include movie post-production, artistic design, and photo enhancement. Early color transfer methods apply spatially-invariant color transformation [1], [2], [3], or spatial color mappings [4], [5], [6], [7], [8], [9], [10] to address this problem. Recently, deep-learning-based methods [11], [12], [13] can achieve more advanced semantic-level color transfer, such as face-to-face or cloth-to-cloth transfer, by leveraging deep features.

Compared to image recolor, a more challenging task is 3D model recolor, which requires 3D design knowledge and is often done manually by professional 3D artists. To circumvent tedious user interactions, we aim to design a framework that can automatically transfer colors from a given reference image to a 3D model, as shown in Figure1. As we know, the geometry of a 3D model is usually saved as a triangle mesh, and a texture is painted over the mesh. The texture is flattened and stored as an image that is applied to the model by using the so-called UV-mapping. A simple way for 3D recolor is to apply existing image color transfer methods to the texture image. However, flattening a texture often breaks its semantic structures thus leads to the poor performance of semantic-level color transfer. Moreover, seams will be visible on the rendered results, because the color transfer is not consistent along with the split patches of the texture. Another naive extension is to render the 3D model into frames and transfer colors to each frame independently. However, without considerations of temporal and view coherency, it is easy to produce flickering artifacts.

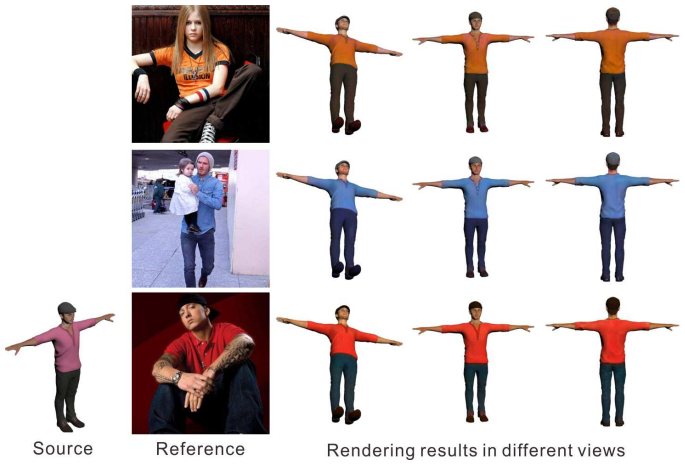


Fig. 1. Our deep exemplar-based 3D color transfer method provides the capability of generating multiple plausible color transfer results for a 3D model by giving different references.

Addressing the above challenges, we propose a novel 3D color transfer framework that incorporates 3D model constraints into the semantic-level color transfer. We first render the 3D model in different views. The color of each view is transferred with a given reference image by using our Color Transfer Network. This network uses a non-local component [14] to align the reference to the source image based on semantic correspondences and learns natural color transfer from big data with Generative Adversarial Network (GAN). Compared to the iterative optimization based color transfer methods [11], [12], [13], our feed-forward network can achieve higher or comparable quality but hundreds of times acceleration. After transferring color to each view, we optimize the colors of 3D model by combining multi-view results and back-propagating the combination objective into the UV-mapped texture through the render. This 3D Model

- *M. Zhang is with the Sate Key Lab. of CAD&CG, Zhejiang University, Hangzhou, 310058, China.E-mail: zhangmohan@zju.edu.cn*
- *J. Liao is with the Department of Computer Science, City University of Hong Kong, Hangkong, 999077, China. J. Liao is the corresponding author.*
- *J. Yu is with the Sate Key Lab. of CAD&CG, Zhejiang University, Hangzhou, 310058, China. He is also a guest professor at the Department of Computer Science, Harbin Finance University, 150020, China.*

Manuscript received April 19, 2005; revised August 26, 2015.

Texture Optimization module can eliminate the artifacts in a single view, producing a seamless and coherent color-transferred texture for rendering.

In brief, our major contributions are:

- We present the first deep-learning-based method to transfer the color from a reference image to a 3D model.
- We propose a new feed-forward convolutional neural network (CNN) for accurate semantic-level image color transfer by finding dense semantic correspondences.
- Our 3D model texture optimization module can generate seamless and coherent texture by combining multi-view transferred results with the UV mapping constraint.

We show how our color transfer technique can be effectively and efficiently applied to a variety of 3D models, such as characters, animals, and buildings. Potential applications of our technique range from art design, game development, film industry to digital arts.

2 Related Work

In this section, we review techniques that are related to our work, including image color transfer, image colorization, and 3D style/color transfer.

2.1 Image Color Transfer

Traditional methods. Global color transfer algorithms process an image by applying a spatially invariant function which is effective in global color shifts (e.g., sepia) and tone curves (e.g., high or low contrast). It was first introduced in [1] as a simple histogram reshaping, where the mean and variance of each channel are transferred separately by LAB color space. Pitie et al. [2] matched two 3D distributions through rotations and 1D histogram projections. Freedman and Kisilev [3] proposed a method to compute the transfer for each histogram bin with the mean and variance of pixel values in the bin, which strikes a compromise between mean-variance based methods and histogram based methods. These methods only consider global color statistics, ignoring the spatial layout of the image. Local color transfer algorithms that apply spatial color mappings are more expressive and can deal with a broad class of applications including time-of-day hallucination [7], [15], weather and season change [7], [8] and style transfer [16], [17], [18]. They either require user interaction [4], [10] or rely on automatic image segmentation algorithms to estimate regional correspondence [5], [19], [20]. However, because of some pixels to be transferred to inaccurate colors, such local matches are not yet precise enough.

Deep-learning-based color transfer. Chang et al. [21] proposed a color transfer algorithm that recolors the image based on a user-modified palette. However, it is a global color transfer, which does not consider local semantic correspondences. Gatys et al. [22], [23] employed feature maps of discriminatively trained VGG-19 [24] to achieve groundbreaking performance for painterly style transfer. Luan et al. [11] extended this work to color transfer and later [25] added a post-processing step with screened Poisson for photorealistic improvement. These two methods require additional segmentation maps [26] to indicate the region-level semantical correspondences. Based on the observation that deep features of pre-trained CNN such as VGG-19 can serve as a good descriptor of image semantics, Liao et al. [12] presented a method to take advantage of multi-scale deep features for semantical dense correspondence estimation and local color transfer. He et al. [13] then improved the color transfer quality of [12] by progressively transferring the color from coarse to fine. Although these deep-learning-based color transfer methods can achieve accurate semantic-level color transfer, they are prohibitively slow because of the iterative optimization nature. Recently, some end-to-end photorealistic style transfer works [27], [28] showed impressive performance on the color transfer task. However, they only guarantee region-to-region transfer via additional segmentation masks, which are not accurate at the pixel level. In contrast, our feed-forward network could achieve more accurate color transfer results by finding dense semantic correspondences.

Colorization. Adding colors to a gray image, known as colorization, is an ill-conditioned problem since there are potentially many colors that can be assigned to the gray pixels. To address this issue, early approaches rely on user scribbles [29] and learning-based algorithms [30], [31] learn color statistics of natural images from large extensive datasets. Given one reference image instead of user input, some automatic methods [4], [20] transfer the chrominance from the color reference to the gray image based on their correspondences. He et al. [32] integrate reference images into a learning-based method to achieve automatic exemplar-based colorization and Zhang et al. [33] extend it to video colorization. Exemplar-based colorization is different from color transfer because colorization is limited to transfer the chrominance only and keep the luminance unchanged, while the color transfer is capable of transferring both luminance and chrominance. Allowing changes in the luminance channel can make the transferred results more faithful to the reference, but will impose new challenges on image structure preservation. This requires us to design new loss and structure in our color transfer network compared to existing colorization networks [32], [33].

2.2 3D Style/Color Transfer

Neural style transfer [22] is to migrate an artistic style from one image to another by leveraging deep neural networks. Some recent works push the neural style transfer even further to the 3D models. Kato et al. [34] proposed an approximate gradient for rasterization that enables the integration of rendering into neural networks so that vertices of 3D mesh could be adjusted based on a given reference image to satisfy the style transfer loss. Later, Liu et al. [35] proposed a general-purpose back-end optimization that can back-propagate change in the image domain to the 3D mesh vertex positions by constructing a differentiable triangle mesh renderer. 3D style transfer is one of its applications. Different from [34], [35], Mordvintsev et al. [36] did not modify the geometry of the 3D models. Instead, they used a 3D rendering process to turn 3D models into 2D images which can be fed into the network, and then back-propagated an objective function through the render to optimize the texture of 3D models. Besides, Fišer et al. [37] presented an extended synthesis algorithm that better preserves the visual richness of hand-created style exemplars by taking into account

illumination effects. However, [34], [35], [37] and [36] all aim at transferring the painterly style rather than color so they will introduce undesired style patterns when applied to the color transfer task. Moreover, their style transfer is global without considering the semantic correspondences between the texture and the given reference.

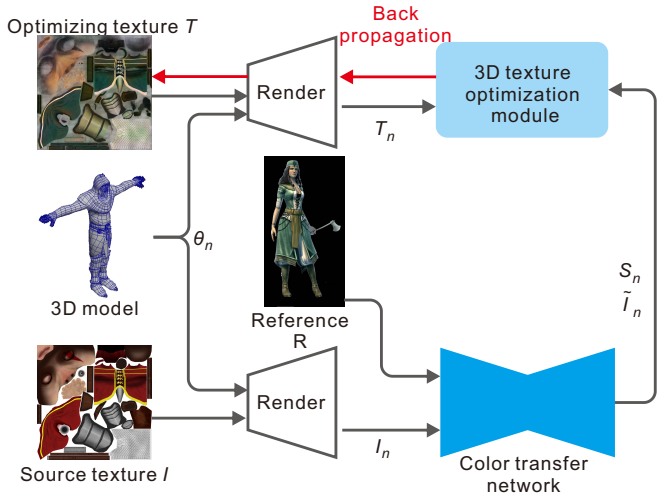


Fig. 2. The overall framework of our deep exemplar-based 3D color transfer.

To our knowledge, there is only one attempt at the 3D model color transfer. Liu et al. [38] achieved it by segmenting the 3D model and the reference image into several parts. The region-based correspondences are built between them, and the patch-match algorithm is applied for region-to-region color transfer. However, low-level features instead of high-level semantics are used in their correspondences estimation, so it could only deal with some simple models. In contrast, our method of deep learning is to build semantic correspondences between the reference image and 3D model, and thus, we can achieve more accurate color transfer and deal with more complex 3D models.

3 Overall Framework

Our framework takes a source 3D model texture I, and a reference image R as input. Note R is an intrinsic image generated by intrinsic decomposition [39], to remove lighting influences in the reference image. The overall framework of our 3D color transfer method is illustrated in Fig. 2. We start from a randomly initialized target texture T and then optimize it iteratively. At the nth iteration, we set a random camera position, which is oriented towards the center of the bounding box of the 3D object, and render two images of it: one with the source texture, denoted as I_n; the other with the target texture that we are currently optimizing, denoted as T_n. We construct a network ξ to transfer colors from the reference R to the rendering image I_n, yielding the color transferred image $\widetilde{I_n}$ and a matching confidence map S_n:

$$(\widetilde{I_n}, S_n) = \xi(I_n\ ,\ R). \tag{1}$$

We enforce T_n to be similar to $\widetilde{I_n}$ by defining an objective function measuring their differences and using S_n to weight different pixels. The gradients of the objective function are back-propagated through the render to update the target texture T. These rendering, color transfer, and optimization procedures are iterated until T converge, which means the reference color is successfully transferred to the target texture. Since the render in our framework is a standard one, in the following sections, we focus on the description of the other two major parts: the color transfer network and the 3D texture optimization module.

4 Color Transfer Network

Inspired by two-stage framework used in the exemplar-based colorization methods [32], [33], we also construct our color transfer network ξ with two subnets: a correspondence subnet $\mathcal{C}$ which learns the semantic correspondences between the source and reference images and a recolor subnet $\mathcal{R}$ which migrates colors form the reference to the source based on correspondences, as illustrated in Fig. 3. Different from [32], [33] which only predicts chrominance channels, our network predicts both luminance and chrominance channels. This requires new loss and structure designs in our network. Besides, our aim is the color transfer for the 3D model, so our network should output the confidence map which is required in the stage of multi-view optimization. Next, we will describe our color transfer network in detail.

4.1 Correspondence Subnet

The correspondence subnet $\mathcal{C}$ builds the semantic correspondence by using deep features extracted from pre-trained VGG-19. Given the source image I_n and the reference R at the nth iteration, we extract feature maps from layers of $relu2\_1$, $relu3\_1$, $relu4\_1$ and $relu5\_1$ for both of them. These feature maps are then fed into several residual blocks parameterized by $\delta_{\mathcal{C}}$ and upsampled to the same resolution. We then concatenate the outputs as $\Phi_{In}, \Phi_R \in \mathbb{R}^{H\times W\times C}$ for I_n and R, and reshape them into feature vectors: $F_{In}, F_R \in \mathbb{R}^{HW\times C}$ respectively. Similar to the non-local component [14], the dense correspondence between I_n and R could be indicated by a correlation matrix $\mathcal{M}_n \in \mathbb{R}^{HW\times HW}$:

$$\mathcal{M}_n(i,j) = \frac{(F_{In}(i) - \mu_{In}) \cdot (F_R(j) - \mu_R)^t}{||F_{In}(i) - \mu_{In}||_2 ||F_R(j) - \mu_R||_2} \tag{2}$$

where μ_{In} and μ_R represent mean feature vectors of F_{In} and F_R respectively, and $\mathcal{M}_n(i,j)$ represents the similarity of F_{In} at position i and F_R at position j. According to this correlation matrix, we could warp the reference image R to the input I_n by approximately calculating the weighted sum of R:

$$\mathcal{W}_n(i) = \sum_j \underset{j}{\texttt{softmax}}(\mathcal{M}_n(i,j)/\tau) \cdot R(j). \tag{3}$$

We set $\tau = 0.01$ so that the row vector $\mathcal{M}_n(i,\cdot)$ approaches to one-hot vector and weighted color $\mathcal{W}_n$ approximates selecting one pixel in the reference with largest similarity score.

Because the warped color is not accurate everywhere, we output the matching confidence map $\mathcal{S}_n$ which is obtained by calculating the maximum matching score for each position i of I_n, to measure the reliability of sampling the reference color for each position:

$$\mathcal{S}_n(i) = \max_j \mathcal{M}_n(i,j). \tag{4}$$

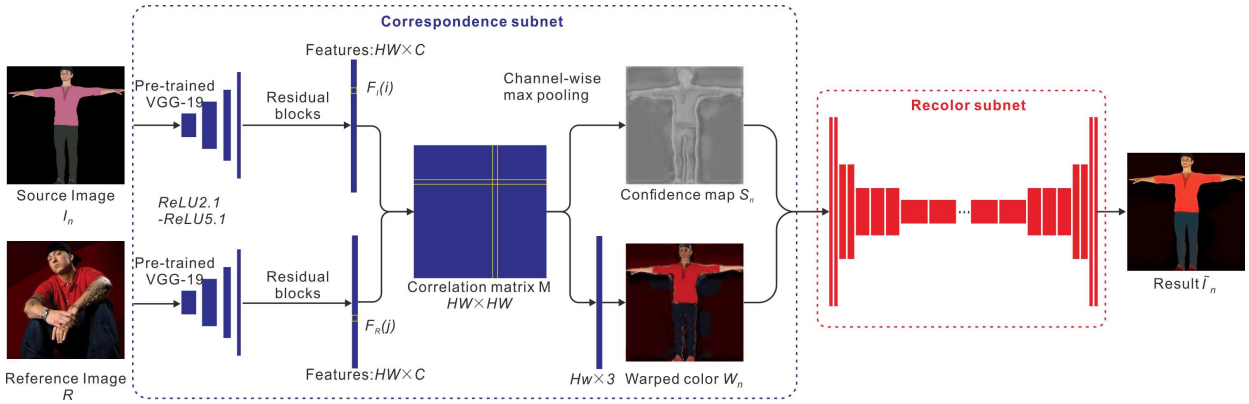


Fig. 3. The architecture of our color transfer network. The correspondence subnet finds the correspondence between source image I and reference image R in the deep feature domain, and aligns the reference color accordingly. Based on the intermediate result of the confidence map, the recolor subnet predicts the color for the result image.

In summary, we could acquire two outputs in the nth iteration from the correspondence network: warped color $\mathcal{W}_n$ and confidence map $\mathcal{S}_n$ to guide the color transfer in the next step:

$$(\mathcal{W}_n, \mathcal{S}_n) = \mathcal{C}(I_n, R; \delta_{\mathcal{C}}) \tag{5}$$

4.2 Recolor Subnet

The correspondence is not accurate everywhere, we, therefore, design a recolor subnet $\mathcal{R}$, parameterized by $\delta_{\mathcal{R}}$, to learn selection, propagation, and prediction of colors simultaneously. As illustrated in Fig. 3, $\mathcal{R}$ takes four-channel map as the input, which concatenates the warped color map $\mathcal{W}$ and the confidence map S_n. With these as the input, the recolor subnet predicts the color map in the nth iteration and could be expressed as:

$$\widetilde{I_n} = \mathcal{R}(\mathcal{W}_n, S_n; \delta_{\mathcal{R}}). \tag{6}$$

4.3 Loss

Our network is supposed to produce the output $\widetilde{I_n}$ to be as close as possible to the reference R, say, the color transfer result should resemble the reference in the corresponding regions. On the other hand, we encourage the color transfer result to be natural, even when no reliable reference color is available. To achieve these objectives, we impose the following losses.

Contextual Loss. First, to encourage colors in the output to be as close as those in the reference, we employ the contextual loss [40]. In the nth iteration, we exact feature maps of both the resulting image $\widetilde{I_n}$ and the reference image R from the VGG-19 network, then upsample and concatenate them together to obtain the concatenated feature maps $\Phi_{\widetilde{I_n}}$ and Φ_R. The normalized cosine distances between each pair of feature points $i \in \Phi_{\widetilde{I_n}}$ and $j \in \Phi_R$ is calculated as:

$$\widetilde{d}(i,j) = \frac{d(i,j)}{\min_k d(i,k) + \epsilon}, \epsilon = 1e-5 \tag{7}$$

where $k \in \Phi_R$ and $d(i,j)$ indicates the cosine distance between i and j. The pairwise affinities $A(i,j) = \mathtt{softmax}_j(1-\widetilde{d}(i,j)/h)$ measures the similarity of $\Phi_{\widetilde{I_n}}$ and Φ_R, in which, we set the bandwidth parameter $h = 0.1$ as recommended in [40]. Consequently, the contextual loss between the result $\widetilde{I_n}$ and the reference R could be obtained:

$$\mathcal{L}_{CX} = -\log(\frac{1}{N}\sum_i \max_j A(i,j)) \tag{8}$$

where N denotes the total feature point number.

Gradient Difference Loss. The contextual loss only encourages the result colors to be similar to those in the reference, but there is no guarantee that the content structure of the source image will be preserved. To penalize structure distortions in the transferred result, we add a gradient difference loss, which directly measures the L1 difference of image gradients:

$$\mathcal{L}_{GD} = ||\nabla\widetilde{I_n^l} - \nabla I_n^l||_1 \tag{9}$$

where ∇ represents the gradient operator. $\widetilde{I_n^l}$ is the luminance channel in the LAB color space of the resulting image $\widetilde{I_n}$ and I_n^l is that of the source image I_n. We employ the Sobel operators that are convolved with the image to calculate the derivatives, for horizontal $\widetilde{G_x}$ and vertical $\widetilde{G_y}$, for the resulting image, we have:

$$\widetilde{G_x} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \widetilde{I_n^l}, \widetilde{G_y} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * \widetilde{I_n^l} \tag{10}$$

Then the gradient of the result image could be obtained by $\nabla\widetilde{I_n^l} = \sqrt{\widetilde{G_x}^2 + \widetilde{G_y}^2}$. Here we only consider the luminance channel because compared to chrominance channels it is more important to reflect image structures.

Identity Loss. We found when the gradient difference loss is applied, image structures could be preserved, but some global color shift may happen. Therefore we employ the identity loss to serve as the color anchor. To train with this loss, we regard the same image as both the source image and the reference image in some batches, and calculate pixel-level and feature-level L1 differences between the resulting image $\widetilde{I_n}$ and the source image I_n:

$$\mathcal{L}_{ID} = (||\widetilde{I_n} - I_n||_1) + \lambda_{feat}\sum_{i=1}^{L} ||\phi_{\widetilde{I_n}}^L - \phi_{I_n}^L||_1 \tag{11}$$

where ϕ^L denotes $reluL\_1$ level VGG feature map. λ_{feat} and is the identity loss weights and set to be 0.2 in our experiments.

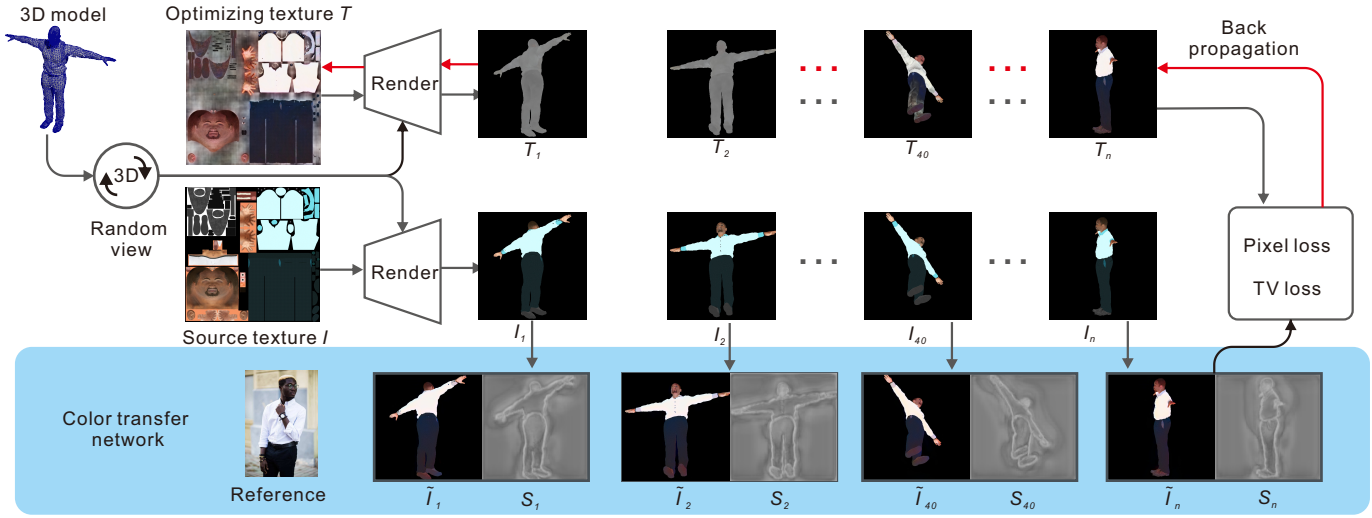


Fig. 4. The pipeline of 3D model texture optimization module.

Adversarial Loss. Besides, we employ an adversarial loss [41] to constraint the color transfer output look photo-realistic. For more stable training, we adopt the least-squares GAN [42]. The loss is defined as:

$$\mathcal{L}_{adv} = \mathbb{E}_{y\sim\mathcal{P}_y}[(D(y)-1)^2] + \mathbb{E}_{\widetilde{In}\sim\mathcal{P}_{\widetilde{In}}}[(D(\widetilde{I_n}))^2] \quad (12)$$

where our color transfer network ξ tries to generate images that look similar to images from real photo domain P_y, and the discriminator D aims to distinguish between the generated image $\widetilde{I_n}$ and the real photo y.

Objective Function for Color Transfer. Combined with the above four losses, we have the objective function to be optimized as:

$$\mathcal{L}_c = \lambda_{CX}\mathcal{L}_{CX} + \lambda_{adv}\mathcal{L}_{adv} + \lambda_{GD}\mathcal{L}_{GD} + \lambda_{ID}\mathcal{L}_{ID} \quad (13)$$

where λ denotes weights of each term. With the guidance of these losses, we successfully unify the correspondence estimation and color transfer within a single network ξ, which learns to generate plausible results based on the reference image.

5 3D Model Texture Optimization

Now we can transfer colors between images using our network ξ. In this section, we would apply it to the 3D model color transfer. We use a 3D rendering process to turn the 3D model texture into 2D images which can be fed into the network, then back-propagate through the rendering process to optimize the texture of the 3D model.

5.1 Rendering Process

As we know, the 3D model geometry such as '.obj' file, is usually saved as a collection of interconnected mesh, and the texture of the 3D model is usually saved as an image such as '.png' file. Each vertex of mesh is associated with a coordinate, which is necessary during the rendering process, that is, the color of each pixel in the screen could be sampled from the texture. Then combined with any illumination model, the 3D model could be rendered on the screen. While in our rendering framework we don't adopt any light and illumination model, so the rendering process could be regarded as a transformation of the texture.

In the nth iteration, we render the 3D model with the source texture I and the optimizing target texture T to acquire two rendering images I_n and T_n respectively. We use θ_n to indicate the rendering parameters such as a random camera position in nth iteration, so the rendering images could be indicated as $I_n = f(I, \theta_n)$ for I and $T_n = f(T, \theta_n)$ for T, as illustrated in Fig. 2.

5.2 Multi-view based Optimization

We initialize the target texture T with random noises and optimize it by an iterative method. In the nth iteration, we feed the rendering image I_n and the reference image R into the color transfer network ξ proposed in Sec. 4, to obtain the color transferred result $\widetilde{I_n}$ and the matching confidence map S_n. To enforce the other image T_n which has been rendered in the same view but with a different texture T, close to the color transfer result $\widetilde{I_n}$, we define an objective function L_{3D} between them. The target texture T is updated by minimizing L_{3D}:

$$\arg\min_{T} L_{3D}(T_n, \widetilde{I_n}, S_n). \quad (14)$$

Since the texture coordinates of vertexes are rendered to the frame buffer, we know how each pixel in T_n is corresponding to some pixels in T. This supports to back-propagate the gradients from T_n to T, and optimize the target texture T via a gradient descent method.

However, in each iteration, $\widetilde{I_n}$ rendered in a single view only cover parts of the texture. Hence we need to randomly sample another camera position to render the 3D model with an updated texture T in the next iteration and repeat the above optimization. After several iterations, we could seamlessly combine multi-view results to optimize the whole target texture T.

5.3 Energy Function

Next we introduce components in the loss functions L_{3D}.

Pixel Loss. First, to transfer colors to the target texture T, we should require its rendered image T_n to be as similar as the color transferred result $\widetilde{I_n}$. We therefore adopt a loss measuring pixel-level L2 difference between T_n and $\widetilde{I_n}$ in the nth iteration:

$$\mathcal{L}_{pix} = ||(T_n - \widetilde{I_n}) \otimes S_n||_2^2 \tag{15}$$

where $\otimes$ indicates the Hadamard product. We adopt the confidence map S_n to weight pixels because we want to reduce the influence of these bad color-transferred regions which often have low confidences in the semantic matching.

Total Variation Loss. We also employ a total-variation loss to eliminate noise in the optimized result T, which could be calculated by:

$$\mathcal{L}_{tv} = \sum_{j \in \mathbb{N}(i)} ||T_n(i) - T_n(j)||_2^2 \tag{16}$$

where i, j denote pixel coordinates of T_n and $\mathbb{N}(i)$ defines the 8-connected neighborhoods of pixel i.

In summary, the objective function to optimize T is defined as:

$$L_{3D} = \mathcal{L}_{pix} + \lambda_{tv}\mathcal{L}_{tv}, \tag{17}$$

where $\lambda_{tv} = 1.0 \times 10^{-5}$is the weight to balance two terms.

Fig. 5. Ablation study for different losses.

6 Experiment

Our system is implemented using Pytorch, Lucid Library, and runs on a PC with 3.7GHz CPU (Intel(R) Core i7-8700k), 32GB memory, and NVIDIA GeForce GTX 1080Titan $\times$ 2. In this section, we did some ablation studies to validate two major parts of our framework. We first qualitatively and quantitatively compare our color transfer network to some existing techniques. Then we compare our 3D texture optimization module with some naive extensions of color transfer from 2D to 3D.

6.1 Discussion on Color Transfer Network

Ablation on different losses. We first ablate the loss functions individually and evaluate their importance, as shown in Fig. 5. When we remove $\mathcal{L}_{CX}$, the result does not resemble the reference style. Without $\mathcal{L}_{adv}$, the result appears washed out in color and contains some undesired artifacts. If $\mathcal{L}_{ID}$ is ablated, some global color shift may happen. When removing $\mathcal{L}_{GD}$, the image structure is destroyed. In contrast, our full model produces vivid results with fewer artifacts.

Comparison with color transfer methods. We compare our color transfer network with several representative previous works: the traditional color transfer method [1], and more recent deep-learning-based methods [11], [12], [13] as shown in Fig. 6. To provide a fair comparison, all the results are run in their publicly available code with default parameter settings.

TABLE 1
Runtime of different state-of-art methods

methods	[11] (s)	[12] (s)	[13] (s)	[32] (s)	[33] (s)	Ours (s)
Girl	128	63	121			∼1
House	104	33	224			∼1
Lord	112	48	127			∼1
Knight	121	39	89			∼1
SWAT				35	∼1	∼1
Bear				25	∼1	∼1
Assassin				48	∼1	∼1

The global color transfer method [1] only matches the global color statistics between source images and reference images, thus limiting the ability to conduct more sophisticated color transformations. For instance, in 1st, 3rd and 4th results, cloth colors could not match these in reference images. In contrast, our method can handle the object-object color transfer because the semantic correspondences of the source and reference images are built by our correspondence subnet.

For methods of Luan et al. [11] and Yoo et al. [28], they match the statistics of deep features between source and reference images and guarantee region-to-region transfer via additional segmentation masks. However, such region-to-region correspondences are not accurate enough at the pixel level, which causes, for example, some posterization artifacts in the hand region of the $4th$ result of Luan et al. [11] and color mismatches to the reference image in the cloth region of the $1st$ result of Yoo et al. [28]. On the contrary, our method does not require segmentation masks. It builds pixel-to-pixel correspondences based on deep features extracted from a different level, thus could transfer colors in finer granularity..

The methods of [12] and [13] also find dense semantic correspondences between source images and reference images, yielding the results that are more similar to ours. However, their methods are not learning-based, and thus some unnatural colors caused by mismatching cannot be resolved. For example, in the bottom row of Fig. 6, the face region is not available in the reference image, so their methods transfer wrong colors to the face. In contrast, our method learns natural color distribution from big data (over 20000 natural image pairs collected from ImageNet), especially with the adversarial loss, which could predict plausible colors even when no reliable reference color is available.

Moreover, our method based on the feed-forward network is much more efficient than these optimization-based methods, in the inference stage. We unified the size of the source and reference images in Fig. 6 to 512×512, which is affordable by a modern GPU, and tested the run-time performance of our method and other methods, as shown in Tab. 1. It can be seen that our method is hundreds of times

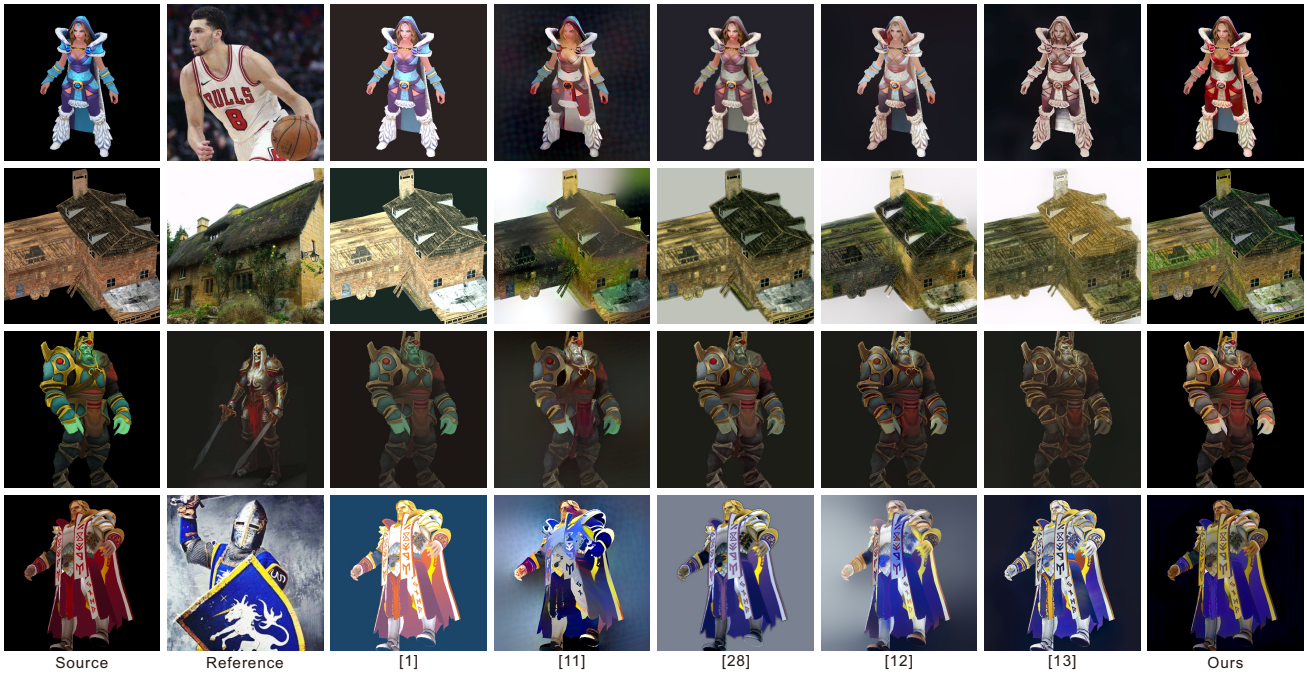


Fig. 6. Comparison with state-of-the-art image color transfer methods. Note that additional segmentation masks of source and reference images should be applied for region-to-region based methods [11], [28].

faster compared to others. This acceleration is especially important when extending to 3D color transfer since we need to process many images rendered in different views.

Comparison with Colorization methods. We also compare our method against recent exemplar-based colorization methods [32], [33] quantitatively and qualitatively, as shown in Tab. 1 and Fig. 8 respectively. He et al. [32] find dense semantic correspondences with an optimization-based method [12] instead of the feed-forward network, so it is hundreds of times slower than ours. The method of [33] adopts the feed-forward network, so its time consuming is similar to ours, but there are still some major differences compared to ours. First, [32] and [33] aim at colorization of gray-scale image, so the luminance channel of the source image remains unchanged in the result. Therefore, the colors from the reference could not be faithfully transferred to the result, especially when the luminance in the source and reference images are different, as shown by the first two examples in Fig. 8. On the contrary, our network transferring both luminance and chrominance channels could yield result colors that are more similar to the reference. On the other hand, our network is capable of keeping the source color in some regions where its confidence scores are low, as shown by the zoom-in region of the 3rd example in Fig. 8. This advantage is helpful to reduce artifacts and generate plausible results in contrast to colorization methods which ignore all color information in the source image.

6.2 Discussion on 3D Texture Optimization

In this part, we first perform an ablation study for energy functions in Sec. 5.3. Then, to validate the effectiveness of our 3D texture optimization module, we try to replace it with three naive extensions from 2D color transfer to 3D.

Ablation on energy functions. We conduct an ablation study on the energy function terms to evaluate their importance, as shown in Fig. 7. When we remove $\mathcal{L}_{pix}$, colors could not be transferred from the reference image to the result. Without $\mathcal{L}_{tv}$, the result appears noise because of overfitting, better seen in the enlarged part of Fig. 7. In contrast, the full model produces a plausible result with less noise.

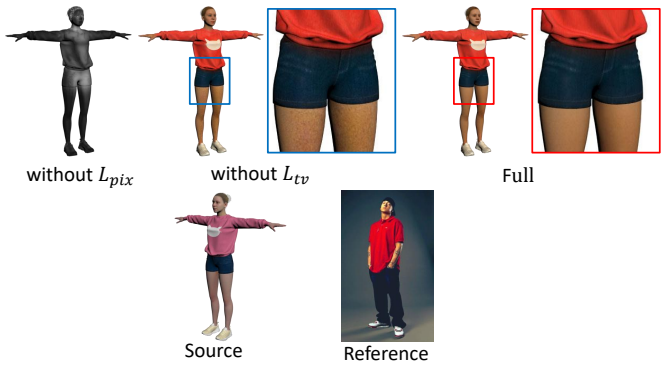


Fig. 7. Ablation study on energy functions.

Color transfer to the flattened texture. The first naive extension we have considered is to flatten the 3D model texture into an image, as shown in the 1st-row 2nd column of Fig. 9. Then we directly fed this image together with the reference image into our color transfer network. Fig. 9 shows results of the color transferred textures (2nd column) and rendering images in different views (3rd $\sim$ 4th columns). Obviously, the naive method (2nd row) presents a poor performance of semantic-level color transfer (e.g. cloth colors), because the texture breaks the semantic structures. Moreover, the naive method suffers from seams as shown in the close-ups (5th column). This is because the color transfer is not consistent along with the split patches of the texture. In contrast, our method maps colors correctly from the reference to the texture by semantic correspondences and generates

Fig. 8. Comparison with colorization methods that use similar network structure with ours.

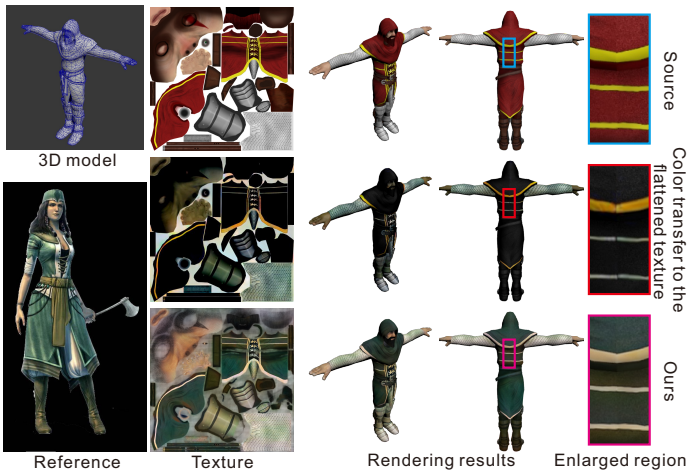


Fig. 9. Comparison with color transfer to the flattened texture. This naive method presents the poor performance of semantic-level color transfer, and seams are visible on the rendering results.

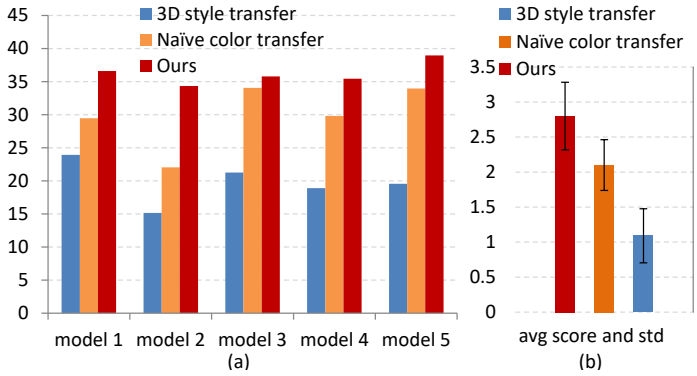


Fig. 10. (a). Quantitative comparison of PSNR, in which the x-axis represents 5 models and the y-axis indicates the average PSNR of 5 views in each model. The proposed method outperforms others in terms of PSNR. (b). User study results on 3D color transfer, and ours are more preferred by human subjects.

seamless results, because the texture is optimized through a 3D rendering process obeying the UV mapping constraint. Last but not least, by using the result of the naive method as the initial value of the target texture, we could also acquire plausible results, and it could help to reduce the required number of iterations in half on average.

Color transfer to rendered frames. Another naive extension is to render the 3D model with the source texture into an animation sequence and then transfer colors of each frame independently by using the image color transfer method. Here we adopt our color transfer network. We compare the transferred frames with our results in Fig. 11. This naive extension causes inconsistent colors between frames, thus flickering artifacts in the animation. For example, in the 1st row of Fig. 11, the cloth in adjacent frames is recolored differently. Unlike it, our method transfers colors to the 3D texture. Rendering different frames with the same target texture can guarantee the view and temporal consistency.

Texture stitching via blending. Given that color transfer to the flattened texture or rendered frames cannot work, we consider another way to extend color transfer from 2D to 3D. Similar to our method, we first render 3D models with source textures in different views to cover the whole texture. For the frame in each view, its colors are transferred by leveraging our color transfer network. Unlike our 3D texture optimization module, in this extension, we directly map these transferred frames back to the texture by reversing the rendering process and stitching them together via linear blending on overlapping regions. However, color transferred results at some viewpoints may be inaccurate because of mismatching, as shown in the 3rd∼4th columns of Fig. 12, so stitching via blending may cause artifacts in results, as shown in 5th∼6th columns in Fig. 12. To address this problem, our 3D texture optimization module combines the multi-view transferred results into a complete texture by iteratively minimizing the objective function (Eq. 17). The pixel loss weighted by the confidence map (Eq. 15) can reduce the influence of inaccurate transferred regions in a single frame (3rd∼4th columns of Fig. 12), and the TV loss (Eq. 16) can guarantee the spatial smoothness in the target texture. Thus our method can generate spatially coherent results which can be seen in the 7th∼8th columns of Fig. 12.

Evaluation. We aware that there is no ground truth for the recolored models thus it might be hard to make a

Fig. 11. Comparison with the method that transfers colors to rendered frames independently. This naive method causes temporal inconsistencies.

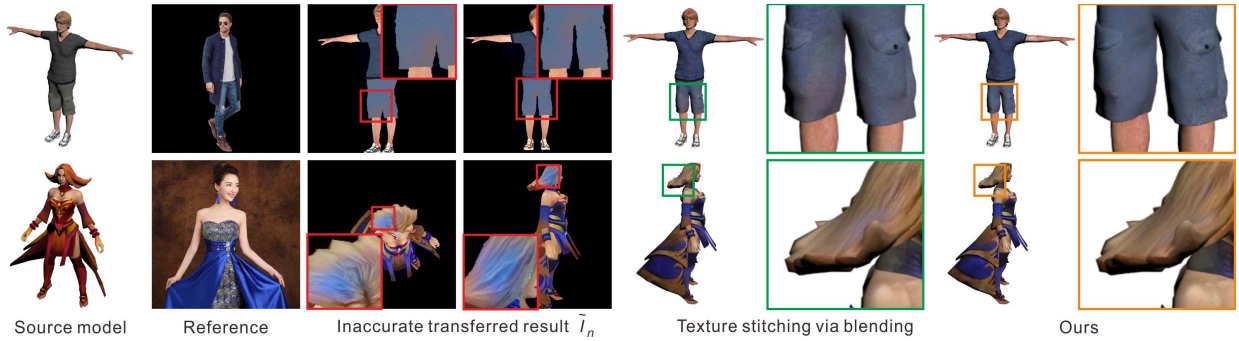


Fig. 12. Comparison to the method of stitching texture via blending. This naive method causes artifacts.

Fig. 13. Comparison with exiting 3D model style transfer method [36] and color transfer method [38]. The 1st∼2nd columns show original models and the 3rd column shows the reference images. The 4th∼5th, 6th∼7th and 8th∼9th columns are final rendering results of different methods.

quantitative evaluation. We therefore propose a pseudo-ground truth to make it easier to evaluate the abilities of our method. We transfer between a reference image and a 3D model of the same content (the reference image is rendered by the 3D model) and measure PSNR between the result and ground truth model. We adopt five 3D models and five views on each model, and corresponding PSNR is shown in Fig. 10(a), in which the x-axis represents five models and the y-axis indicates the average PSNR of five views in each model. We compare our method to the naive color transfer method (Para. 3 in Sec. 6.2) and 3D style transfer [36]. Our method outperforms others in terms of PSNR and the number is 36.3, which demonstrates that our method can reconstruct the ground truth well.

User study. We randomly select 10 3D models from our experiment, and compare our result to the naive extensions of color transfer from 2D to 3D (Para. 3 in Sec. 6.2) and the state-of-the-art 3D style transfer method [36]. 21 volunteers who have background knowledge in computer graphics or deep learning are recruited as the subjects for this task. For each 3D model, volunteers are asked to score each presented result in a 3-to-1 scale from "satisfied color transfer" to "unsatisfied color transfer", and Fig. 10(b) demonstrates the average scores and standard deviations of each method. The result shows that our 3D color transfer results are preferred by human subjects.

7 Comparison and Result

Comparison with 3D transfer methods. We compare our results with two existing 3D model style or color transfer works: [36] (style) and [38] (color), as shown in Fig. 13. Although the method of [36] also adopts multi-view optimization, and combines it with the neural style transfer technique for the 3D model, there are some major differences between this method and ours. The method of [36] adopts style and

Fig. 14. Results of some human character models.

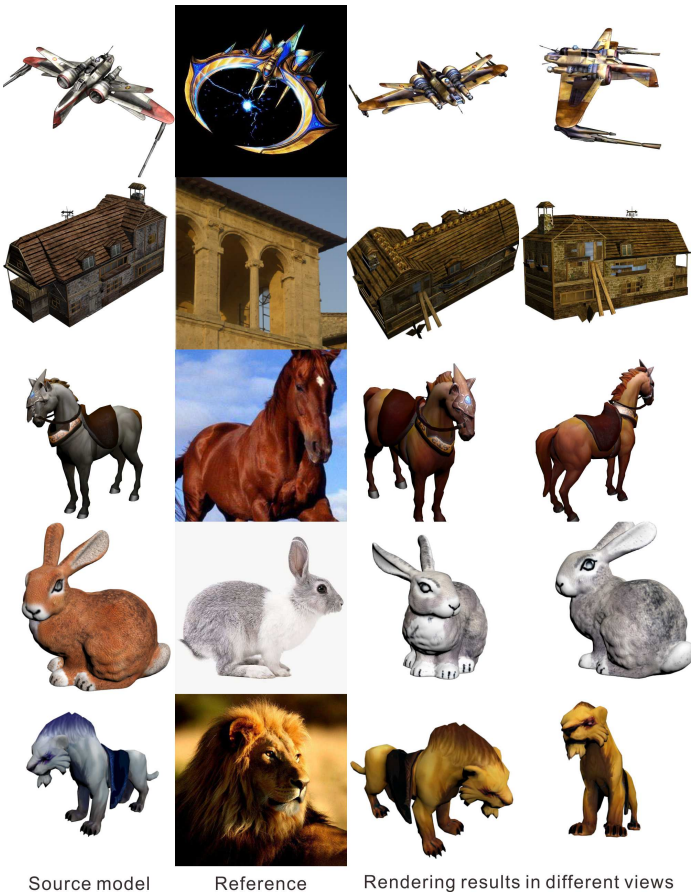


Fig. 15. Results of building, aircraft, and animal models.

content losses, since it focuses on painterly style rather than color. So it will introduce undesirable style patterns when it is applied to color transfer, as shown by the patterns on the skin of the beast in the lower row of Fig. 13. Besides this method is based on a global style transfer, which does not take the semantic correspondence between the model texture and the reference image into consideration, thus suffering from spillovers sometimes, e.g., with the monster skin taking colors from the ground as shown in the Fig. 13. In contrast, our method adopts pixel and TV losses, which could only change colors and preserve the gradient and details in results. Besides, our method also uses the confidence maps to effectively alleviate artifacts existing in individual views, which is not considered by the method of [36]. The method of [38] focuses on color transfer. It segments the 3D model and reference image into several parts to build region-based correspondence and conducts patch-match [43] to synthesize the model texture. However, the segmentation method it adopts is based on quad-tree decomposition, and the matching method is based on low-level features. Thus when the model or the reference image becomes complex as in Fig. 13, it requires manual specification. Otherwise, it results in mismatching and inaccurate color transfer. For example, in the top row of Fig. 13, a few artifacts in the back regions are caused by region mismatches. In comparison, our method builds more accurate and finer-level correspondences by leveraging deep features and thus leads to better semantic-level color transfer.

Convergence & Iterations. Fig. 16 illustrates the relationship between energy (Eq. 14) and iterations in 3D model texture optimization. The x-axis represents the number of iterations, and the y-axis indicates the average energy and standard deviations of each model in our experiments. The energy is converging after 270 iterations. In our implementation, we set 300 iterations as default for each model, and the average process time is about 320 seconds.

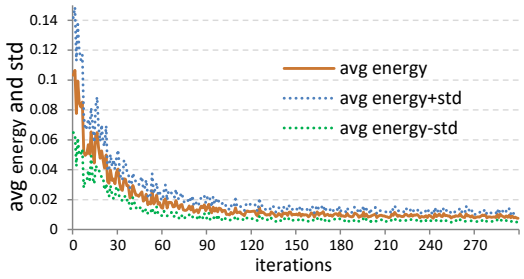


Fig. 16. Convergence analysis of iteration for 3D texture optimization, and the energy is converging after about 270 iterations.

Robustness. To show the robustness of our method, we have applied it to various kinds of models. Fig. 14 presents the results of human character models in which colors of results could be changed upon the semantic (like face-to-face, hair-to-hair, and cloth-to-cloth) correspondences between model textures and references. Our method could also deal with transportation, building and animal models as shown in Fig. 15. Besides, some CG characters which are unusual in reality could be processed by our method as well, and plausible results are shown in Fig. 17. More

results could be seen in the PDF supplemental material, and the animation results could be better seen in the video supplemental material.

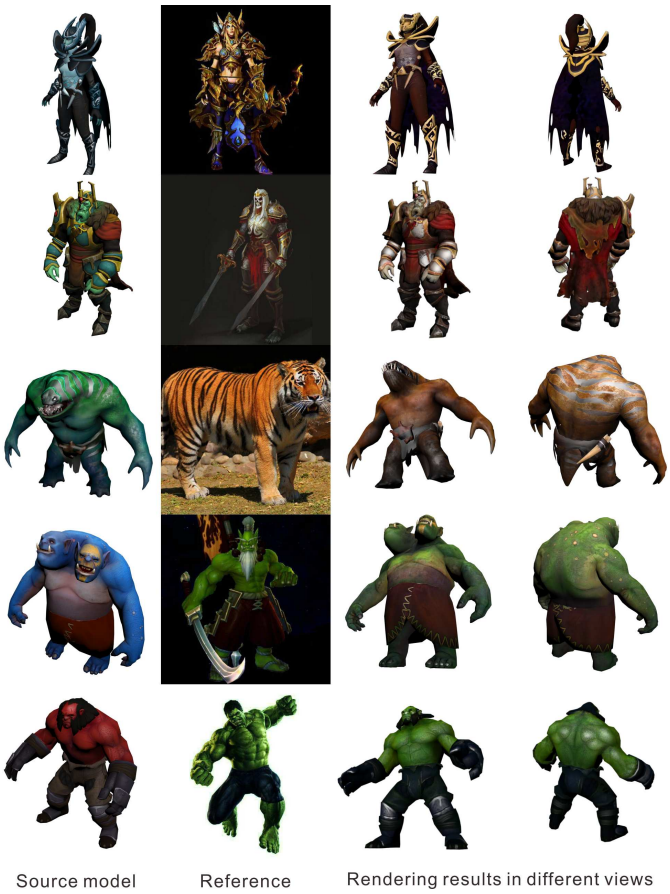


Fig. 17. Results of some CG character models.

8 Conclusion

In this paper, we present the first deep-learning framework for an exemplar-based 3D model recolor. The proposed method takes a 3D model with texture and a reference image as input and automatically transfers the colors from the reference image to the 3D model texture based on their semantic correspondences. Efficient run-time and visually pleasant quality show the potential of our method to be applied in the areas of the art design, game development, film industry, for the fast 3D model editing.

However, our method still has limitations. Our method cannot handle models without texture, since colors and textures are important in building semantic correspondences. Besides, with the help of the confidence map in our multi-view texture optimization, our method is robust to mismatches and could generate correct results in most cases. However, in rare cases, the correspondence subnet may build inaccurate correspondences for a specific region in all views, and then the artifact in that region is unavoidable. We show one failure case caused by this in Fig. 18, where the arm region of the 1st character appears pink artifacts. Moreover, in the stage of rendering, the 3D model is rendered in different views but the same pose such as T-pose, which makes some occluded texture regions in the

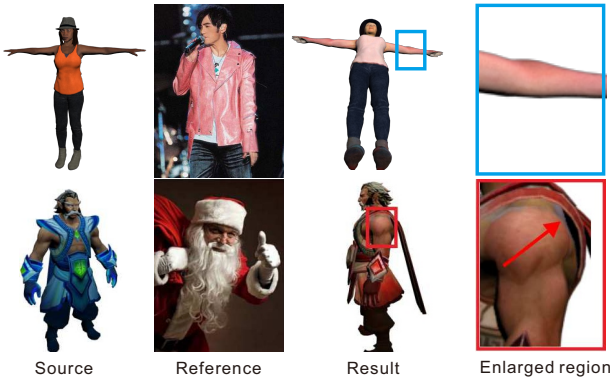


Fig. 18. Some examples of failure cases

model seldom be rendered in the iterative optimization, so these regions could not be optimized in the stage of 3D model texture optimization. For instance, in Fig. 18, the region on the 2nd character's skin covered by cloth presents the initial color in the optimizing texture. In the future, to solve this problem, we will render and optimize the 3D model in different poses, or divide the 3D model into several parts and optimize the texture for them respectively. Besides, transferring the roughness texture of the 3D model is another future direction of this work.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China No. 61772463, the Hong Kong Research Grants Council (RGC) Early Career Scheme under Grant 9048148 (CityU 21209119) and the CityU of Hong Kong under APRC Grant 9610488.

References

[1] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Computer graphics and applications*, vol. 21, no. 5, pp. 34–41, 2001.

[2] F. Pitie, A. C. Kokaram, and R. Dahyot, "N-dimensional probability density function transfer and its application to color transfer," in *Tenth IEEE International Conference on Computer Vision Volume 1*, vol. 2, pp. 1434–1439 Vol. 2, Oct 2005.

[3] D. Freedman and P. Kisilev, "Object-to-object color transfer: Optimal flows and smsp transformations," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 287–294, June 2010.

[4] T. Welsh, M. Ashikhmin, and K. Mueller, "Transferring color to greyscale images," *ACM Trans. Graph.*, vol. 21, pp. 277–280, July 2002.

[5] Y.-W. Tai, J. Jia, and C.-K. Tang, "Local color transfer via probabilistic segmentation by expectation-maximization," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, (Washington, DC, USA), pp. 747–754, IEEE Computer Society, 2005.

[6] K. Sunkavalli, M. K. Johnson, W. Matusik, and H. Pfister, "Multi-scale image harmonization," *ACM Trans. Graph.*, vol. 29, pp. 125:1–125:10, July 2010.

[7] Y. Shih, S. Paris, F. Durand, and W. T. Freeman, "Data-driven hallucination of different times of day from a single outdoor photo," *ACM Trans. Graph.*, vol. 32, pp. 200:1–200:11, Nov. 2013.

[8] P.-Y. Laffont, Z. Ren, X. Tao, C. Qian, and J. Hays, "Transient attributes for high-level understanding and editing of outdoor scenes," *ACM Trans. Graph.*, vol. 33, pp. 149:1–149:11, July 2014.

[9] S. Bae, S. Paris, and F. Durand, "Two-scale tone management for photographic look," *ACM Trans. Graph.*, vol. 25, pp. 637–645, July 2006.

[10] X. An and F. Pellacini, "User-controllable color transfer," *Computer Graphics Forum*, 2010.

[11] F. Luan, S. Paris, E. Shechtman, and K. Bala, "Deep photo style transfer," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 00, pp. 6997–7005, July 2017.

[12] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang, "Visual attribute transfer through deep image analogy," *ACM Trans. Graph.*, vol. 36, pp. 120:1–120:15, July 2017.

[13] M. He, J. Liao, L. Yuan, and P. V. Sander, "Progressive color transfer with dense semantic correspondences," *CoRR*, vol. abs/1710.00756, 2017.

[14] X. Wang, R. B. Girshick, A. Gupta, and K. He, "Non-local neural networks," *CoRR*, vol. abs/1711.07971, 2017.

[15] J. R. Gardner, M. J. Kusner, Y. Li, P. Upchurch, K. Q. Weinberger, and J. E. Hopcroft, "Deep manifold traversal: Changing labels with convolutional features," *CoRR*, vol. abs/1511.06421, 2015.

[16] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 327–340, ACM, 2001.

[17] C. Li and M. Wand, "Combining markov random fields and convolutional neural networks for image synthesis," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2479–2486, June 2016.

[18] A. Selim, M. Elgharib, and L. Doyle, "Painting style transfer for head portraits using convolutional neural networks," *ACM Trans. Graph.*, vol. 35, pp. 129:1–129:18, July 2016.

[19] J.-D. Yoo, M.-K. Park, J.-H. Cho, and K. H. Lee, "Local color transfer between images using dominant colors," *Journal of Electronic Imaging*, vol. 22, no. 3, p. 1, 2013.

[20] B. Arbelot, R. Vergne, T. Hurtut, and J. Thollot, "Local texture-based color transfer and colorization," *Computers & Graphics*, vol. 62, pp. 15 – 27, 2017.

[21] H. Chang, O. Fried, Y. Liu, S. DiVerdi, and A. Finkelstein, "Palette-based photo recoloring," *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 34, July 2015.

[22] L. A. Gatys, M. Bethge, A. Hertzmann, and E. Shechtman, "Preserving color in neural artistic style transfer," *CoRR*, vol. abs/1606.05897, 2016.

[23] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman, "Controlling perceptual factors in neural style transfer," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3730–3738, July 2017.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[25] R. Mechrez, E. Shechtman, and L. Zelnik-Manor, "Photorealistic style transfer with screened poisson equation," *CoRR*, vol. abs/1709.09828, 2017.

[26] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 834–848, April 2018.

[27] Y. Li, M.-Y. Liu, X. Li, M.-H. Yang, and J. Kautz, "A closed-form solution to photorealistic image stylization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 453–468, 2018.

[28] J. Yoo, Y. Uh, S. Chun, B. Kang, and J.-W. Ha, "Photorealistic style transfer via wavelet transforms," in *International Conference on Computer Vision (ICCV)*, 2019.

[29] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM Trans. Graph.*, vol. 23, pp. 689–694, Aug. 2004.

[30] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification," *ACM Trans. Graph.*, vol. 35, pp. 110:1–110:11, July 2016.

[31] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *Computer Vision – ECCV 2016*, (Cham), pp. 649–666, Springer International Publishing, 2016.

[32] M. He, D. Chen, J. Liao, P. V. Sander, and L. Yuan, "Deep exemplar-based colorization," *CoRR*, vol. abs/1807.06587, 2018.

[33] B. Zhang, M. He, J. Liao, P. Sander, Y. Lu, B. Amine, and C. Dong, "Deep exemplar-based video colorization," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'19)*, IEEE Computer Society, 2019.

[34] H. Kato, Y. Ushiku, and T. Harada, "Neural 3d mesh renderer," *CoRR*, vol. abs/1711.07566, 2017.

[35] H.-T. D. Liu, M. Tao, and A. Jacobson, "Paparazzi: Surface editing by way of multi-view image processing," *ACM Transactions on Graphics*, 2018.

[36] A. Mordvintsev, N. Pezzotti, L. Schubert, and C. Olah, "Differentiable image parameterizations," *Distill*, 2018.

[37] J. Fišer, O. Jamriška, M. Lukáč, E. Shechtman, P. Asente, J. Lu, and D. Sỳkora, "Stylit: illumination-guided example-based stylization of 3d renderings," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, pp. 1–11, 2016.

[38] J. Liu, Z. Lian, and J. Xiao, "Auto-colorization of 3d models from images," in *SIGGRAPH Asia 2017 Technical Briefs*, pp. 15:1–15:4, ACM, 2017.

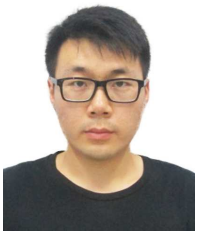
[39] S. Bi, X. Han, and Y. Yu, "An l1 image transform for edge-preserving smoothing and scene-level intrinsic decomposition," *ACM Trans. Graph.*, vol. 34, pp. 78:1–78:12, July 2015.

[40] R. Mechrez, I. Talmi, and L. Zelnik-Manor, "The contextual loss for image transformation with non-aligned data," *CoRR*, vol. abs/1803.02077, 2018.

[41] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pp. 2672–2680, 2014.

[42] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[43] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *ACM Trans. Graph.*, vol. 28, pp. 24:1–24:11, July 2009.



Mohan Zhang received the BSc degree in software engineering from Sichuan University, China, in 2013. Currently, he is working toward the PhD degree at the State Key Lab of CAD&CG, Zhejiang University, China. His research interests include computer animation, physically based modeling and non-photorealistic rendering.

Jing Liao received the dual Ph.D. degrees from Zhejiang University and Hong Kong University of Science and Technology in 2014 and 2015 respectively. She was a researcher in MSRA from 2015 to 2018. She is currently an Assistant Professor with the Department of Computer Science, City University of Hong Kong. Her research interests include image and video processing, computational photography and non-photo-realistic rendering.

Jinhui Yu received the PhD degree in computer graphics from the University of Glasgow in 1999. He is a professor of computer science at the State Key Lab of CAD&CG, Zhejiang University, China. He is also a guest professor at the Department of Computer Science, Harbin Finance University, China. His research interests include image-based modeling, computer animation, and computer graphics art.