# Reference-Based Deep Line Art Video Colorization

Min Shi†, Jia-Qi Zhang†, Shu-Yu Chen, Lin Gao*, Yu-Kun Lai and Fang-Lue Zhang

**Abstract**—Coloring line art images based on the colors of reference images is a crucial stage in animation production, which is time-consuming and tedious. This paper proposes a deep architecture to automatically color line art videos with the same color style as the given reference images. Our framework consists of a color transform network and a temporal refinement network based on 3U-net. The color transform network takes the target line art images as well as the line art and color images of the reference images as input and generates corresponding target color images. To cope with the large differences between each target line art image and the reference color images, we propose a distance attention layer that utilizes non-local similarity matching to determine the region correspondences between the target image and the reference images and transforms the local color information from the references to the target. To ensure global color style consistency, we further incorporate Adaptive Instance Normalization (AdaIN) with the transformation parameters obtained from a multiple-layer AdaIN that describes the global color style of the references extracted by an embedder network. The temporal refinement network learns spatiotemporal features through 3D convolutions to ensure the temporal color consistency of the results. Our model can achieve even better coloring results by fine-tuning the parameters with only a small number of samples when dealing with an animation of a new style. To evaluate our method, we build a line art coloring dataset. Experiments show that our method achieves the best performance on line art video coloring compared to the current state-of-the-art methods.

**Index Terms**—Line Art Colorization, Color Transform, Temporal Coherence, Few Shot Learning

## 1 Introduction

The process of animation production requires high labor input. Coloring is one of the critical stages after the line art images are created, which is a time-consuming and tedious task. Usually, "inbetweeners" colorize a series of line art images according to several reference color images drawn by artists. Some commercial devices and software can be used to speed up the workflow of line art image coloring. Nevertheless, it still needs much repetitive work in each frame. Therefore, automatic methods for coloring line art images based on reference color images are highly demanded, significantly reducing animation production costs.

Early research on line art image coloring mostly relies on manually specifying colors, which are then spread out to similar regions [1], [2], [3]. However, the efficiency of the coloring stage in animation production cannot be significantly improved using the above methods. Inspired by the success of generative models on image synthesis tasks in recent years, researchers have used deep convolutional neural networks (CNNs) to automatically color line art images [4], [5], [6]. However, user interactions are required to achieve final satisfactory coloring results. To encourage temporal consistency in the colored animation, Thasarathan et al. [7] input the previous frame together with the current frame to a discriminator to improve the colorization of neighboring frames. However, the model cannot fully guarantee the consistency between the color styles of their results and the reference image, which is important for the color quality of the animation.

Coloring the whole sequence of line art videos based on a few reference colored sample frames is challenging. Unlike real-life videos, animation videos do not hold pixel-wise continuity between successive frames. For example, lines corresponding to limbs of an animated character may jump from one shape to another to depict fast motion. As such temporal continuity cannot be directly used to guide the learning process to maintain the color consistency between regions in adjacent frames. Furthermore, line art images only consist of black and white lines, which lack rich texture and intensity information than grayscale images, making colorization harder. Moreover, when coloring a new animation video from line art images, only a few examples are available. When applied to color animation videos with a new color style, this requires the model to be trainable with a small number of samples.

This paper proposes a deep architecture to automatically color line art videos based on reference colored frames. In order to avoid possible error accumulation when continuously coloring a sequence of line art images, we determine the region correspondences between the target image and reference images by using a distance attention layer, which is able to match similar region features extracted by the convolutional encoders from the target and reference images, and use this to transform the local color information from the references to the target. Then, to ensure the global color consistency, the Adaptive Instance Normalization (AdaIN) [8] parameters are learned from the embeddings extracted for describing the global color style of the reference images. We

† *Authors contributed equally.*
* *Corresponding Author is Lin Gao (gaolin@ict.ac.cn).*

- *M. Shi is with North China Electric Power University, Beijing 102206, China. E-mail:shi_ min@ncepu.edu.cn*
- *J.-Q. Zhang is with Beihang University, Beijing 100191, China. E-mail:zhangjiaqi79@buaa.edu.cn*
- *S.-Y. Chen and L. Gao are with the Beijing Key Laboratory of Mobile Computing and Pervasive Device, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China and also with the University of Chinese Academy of Sciences, Beijing 100190, China. E-mail:{chenshuyu, gaolin}@ict.ac.cn*
- *Y.-K. Lai is with the School of Computer Science & Informatics, Cardiff University, Wales, UK. Email: LaiY4@cardiff.ac.uk*
- *F.-L. Zhang is with the School of Engineering and Computer Science, Victoria University of Wellington, New Zealand. E-mail:fanglue.zhang@ecs.vuw.ac.nz*

further use a 3U-net based 3D convolutional network to refine the temporal color consistency between the coloring result and the reference images. It is the first time to utilize such a global and local network architecture to color line art images. Moreover, our model can achieve even better coloring results by fine-tuning the parameters with only a small number of samples when dealing with animations of a new color style.

Our technical contributions include:

1) We propose a deep learning-based line art colorization method which is suitable for practical applications where artists need to color several sample frames, and the entire video can be colored with a consistent style.
2) We propose a distance attention layer that can transform colors from the reference images to the target line art image based on pixel-level similarity, which learns how to spatially match the features of reference line art images with the target image.
3) We introduce a joint solution for sketch colorization, which applies several effective modules to solve the line art colorization problem, such as style learning by AdaIN, attention-based line feature matching and temporal convolution.

## 2 Related Work

### 2.1 Sketch colorization without references

Early research on line drawing colorization [1], [2], [3] allows the user to specify color using brushes and then propagates the color to similar areas. The range of propagation can be determined by finding the region similarity or specified by the user. Qu et al. [1] use a level-set method to find similar regions, and propagate users' scribbles to those regions. Orzan et al. [9] require the users to set the gradient range of curves to control the range of their scribbles. Given a set of diffusion curves as constraints, the final image is generated by solving the Poisson equation. Those methods require a lot of manual interactions to achieve desired coloring results. Zhu et al. [10] proposed a toon tracking method, which can color the animation videos by the region correspondence between animation frames. However, when line art images have special effects (such as glare), blurred contour lines, too complicated line structures and a large number of segmentation regions, the final results will be significantly affected. With the development of deep learning, researchers have used it to achieve automatic or interactive coloring [11], [12]. Yoo et al [13] proposed a memory network structure, which matches the features of the input gray-scale image with the features of the training set, and then the AdaIN [8] was used to transfer color after obtaining the corresponding color feature vector. However, a lot of manual interaction is still required to obtain reliable coloring results.

### 2.2 Sketch colorization with references

To reduce manual workload, colorization methods with reference images have been proposed to achieve the specified color style for the target sketch. Sato et al. [18] represent the relationships between the regions of line art images using a graph structure, and then solve the matching problem through quadratic programming. However, it is difficult to accurately segment complex line art images. Deep learning methods are proposed to avoid the requirement of accurate segmentation. Zhang et al. [5] extract VGG features from the reference image as their description, but their


Fig. 1: Comparison of different line art image extraction methods. (a) original color image; (b) Canny [14]; (c) XDoG [15]; (d) Coherent Line Drawing [16]; (e) SketchKeras [17]; (f) distance field map from SketchKeras results.

results have blurred object boundaries and mixed colors, probably due to the different characteristics of line art and normal images. To further refine the results, Zhang et al. [19] divide the line art image colorization problem into the drafting stage and refinement stage, and users can input manual hints or provide a reference image to control the color style of the results [20]. However, these methods still require user interactions to achieve satisfactory coloring results. Hensman et al. [6] use cGANs (conditional Generative Adversarial Networks) to colorize gray images with little need for user refinement, but the method is only applicable to learning relationships between grayscale and color images, rather than line art images. Chen et al. [21] use active learning [22] to infer the color of uncolored areas. In addition, Liao et al. [23] use the PatchMatch [24] algorithm to match the high-dimensional features extracted from the reference and target images, and realize style conversion between image pairs. It can be used for the line art image coloring task, but it does not match images with different global structures very well.

### 2.3 Video colorization

In video coloring research, animation coloring is much less explored than natural video coloring. Normal grayscale video coloring work [25], [26], [27], [28] learns temporal color consistency by calculating optical flow. Using the image analogy method, Jamriška et al. [29] used image color, foreground object binary mask, position of SIFT Flow (Scale-invariant feature transform Flow) [30], and the edge information of foreground objects as guidance to achieve video stylization [31]. Iizuka et al. [32] proposed a single end-to-end framework to tackle black-and-white vintage film remastering. The key idea of their network structure is introducing source-reference attention to guide the transfer of the colors from the reference image into the target image.

However, animated video frames do not hold pixel-level continuity in general, causing optical flow algorithms to fail to get good results. Sýkora et al. [33] propose a method which uses path-pasting to colorize black-and-white cartoons for continuous animation. However, when given sketch images contain lines which are not continuous, path-pasting is hard to find accurate correspondence. Automatic Temporally Coherent Video Colorization

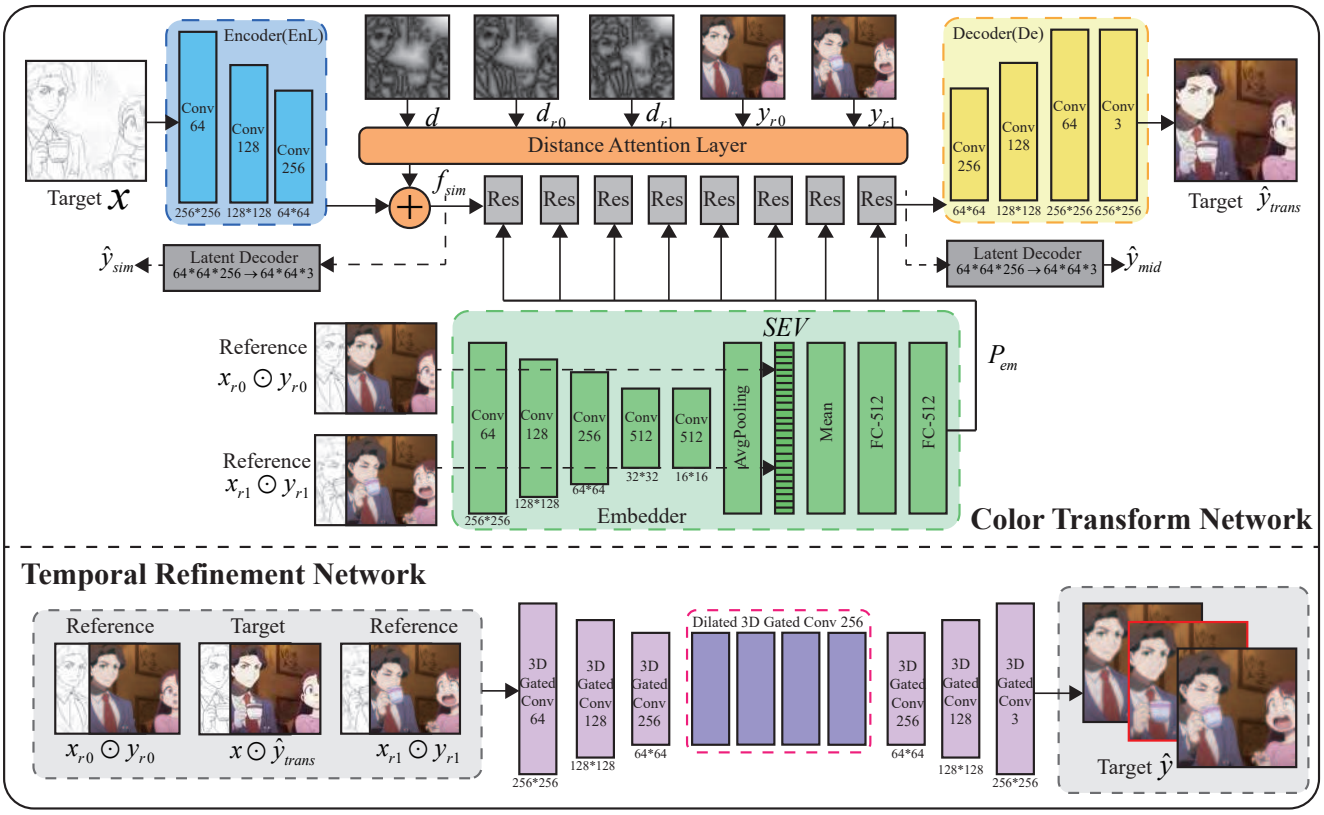



Fig. 2: The overall structure of the our network. The color transform network combines the latent features of global color style extracted from reference color images with latent features extracted from the input line art image to be colored. Finally, the temporal refinement network ensures the coloring result to be temporally consistent with the reference images.

(TCVC) [7] inputs the previous colored frame into the generator as a condition for the current line art image, and inputs both the previous and current colored frames into the discriminator to enforce temporal coherence. However, this method will cause error propagation in the generated coloring results, as errors introduced in one frame affect coloring results of all future frames. Instead, we use a method based on region matching with reference images to find the color features of the target image. Our method can better process long sequences of line art images without color error accumulation.

## 3 METHOD

In animation production, coloring in-between line art frames according to the key frames drawn and colored by the original artist is a necessary step. For automating this process, we design a generative network structure. One of our design goals is to maintain the local color consistency between the corresponding regions in the target and the reference images. We expect that our model will estimate accurate dense color transformations by matching local features between the target and reference line art images. It involves a distance attention layer after the convolutional feature encoding stage. However, due to the possible large deformations of the same object in the line art images, it is insufficient to obtain accurate coloring results by just matching local features of line art images. Therefore, we also expect the network to be capable of controlling the color style globally and correcting the coloring errors caused by inaccurate local similarity results. We thus propose to use a sub-module, Embedder, to extract the embeddings representing the global color style of the input reference images. The embedding vectors are used to ensure the global color style consistency between the generated coloring result and reference images. Finally, we add a 3D convolution network to improve the coloring quality by enhancing the temporal color consistency.

### 3.1 Data Preparation

Since there is no public dataset for evaluating line art animation coloring, we built such a dataset to train our model and evaluate it. We collect a set of animation videos and divide them into video shots using the color difference between the successive frames. The line art images corresponding to all the color frames of each video shot are extracted. See more details in Sec. 4.1.

In order to obtain high-quality line art images from color animation, we tried various line extraction methods. The line art images extracted by the traditional Canny edge detection algorithm [14] are pretty different from the line drawing images drawn by cartoonists. Other extractors, like XDoG edge extraction operator [15] and the Coherent Line Drawing [16] method, are too sensitive to user-specified parameters. Sketch-Keras [17] is a deep line drafting model that uses a neural network trained to not only adapt to color images with different quality but also to extract lines for the important details. Therefore, here we use the sketch-Keras to generate line art images, which have overall best lines, contain rich details and have the most similar style with that drawn by cartoonists. Due to the data sparsity of lines in line art images, inspired by SketchyGAN [34], we also convert line art images into distance field maps to improve matching between images when training our model. The comparisons of different methods are shown in Fig. 1.

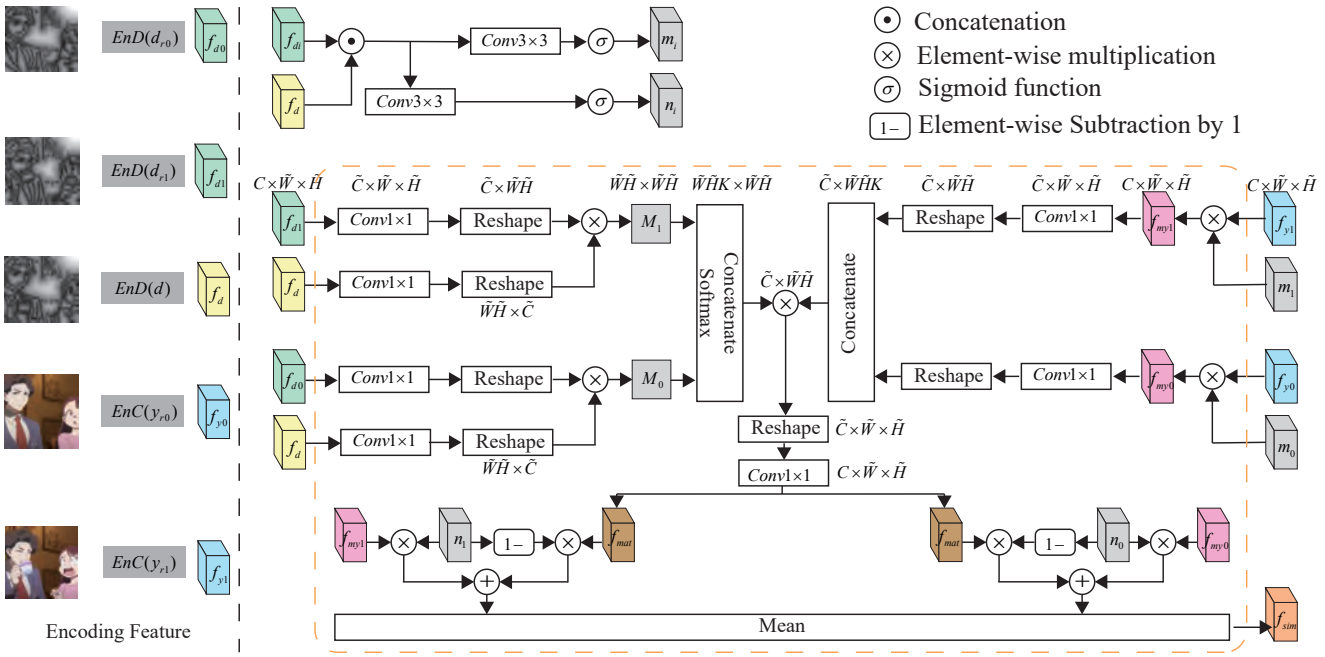



Fig. 3: Distance attention layer. It first extracts the feature maps of the input distance field maps of the target and reference images through the encoder $EnD$. Then the similarity map is calculated for transforming the color features of the reference images to those of the target image $f_{mat}$. Finally, the matching-based transform feature $f_{mat}$ and the color feature of the reference images $f_{my_i}$ ($i = 0, 1$ refers to two reference images) are dynamically combined to obtain the local color feature map of the target image $f_{sim}$.

### 3.2 Overall Network Architecture

Our network consists of two sub-networks, color transform network and refinement network. The color transform network consists of a conditional generator $G$ and an adversarial discriminator $D$. Inspired by the few-shot unsupervised image-to-image translation framework (FUNIT) [35], our generator learns global style information through Embedder, and provides local color features through the distance attention layer. Thus, our generator $G$ takes a target line art image $x$, a target distance field map $d$ and two reference images (including the line art, distance field map and color image for each reference image) $\{x_{r_0}, x_{r_1}; d_{r_0}, d_{r_1}; y_{r_0}, y_{r_1}\}$ as input and produces the target color image $\hat{y}_{trans}$ via

$$\hat{y}_{trans} = G(x, d, \{x_{r_0}, x_{r_1}; d_{r_0}, d_{r_1}; y_{r_0}, y_{r_1}\}). \quad (1)$$

Here, $r$ represents the reference object. $\hat{y}_{trans}$ is the preliminary coloring result generated by the color transform network, where subscripts 0 and 1 represent the beginning and end of the video sequence, respectively.

Figure 2 shows the architecture of the color transform network, which is composed of six parts: encoders, a distance attention layer $Sim$, a group of middle residual blocks $Mid$, a style information Embedder $Em$, a decoder $De$, and a discriminator $D$.

First, we use the encoders to extract the feature maps of the following images: the target line art image $EnL(x)$, the target distance field map $EnD(d)$, the reference distance field maps $EnD(d_{r_0}, d_{r_1})$, and the reference color images $EnC(y_{r_0}, y_{r_1})$. $EnL$, $EnD$, and $EnC$ are three identically constructed encoders that extract the features of line art images, field distance maps, and color images, respectively. The extracted feature maps of the above distance field maps and color images are then fed into the distance attention layer to obtain the local color features $f_{sim}$ of the target line art image, which provides a local colorization for the target image.

After concatenating the line art images and the reference color images, we also feed them separately into an Embedder module [35] to get intermediate latent vectors for reference images, and then compute the mean of their intermediate latent vectors to obtain the final style embedding vector (SEV) $Em(x_{r_0} \odot y_{r_0}, ..., x_{r_1} \odot y_{r_1})$. The SEV is used to adaptively compute the affine transformation parameters $P_{em}$ for the adaptive instance normalization (AdaIN) residual blocks [8] via a two-layer fully connected network, which learns the global color information for the target image. Then, with the SEV controlling the AdaIN parameter, the output features of the $Sim$ layer (capturing similarity matching between the target and references) and $EnL$ (capturing local image information of the target) are added to get the input to the $Mid$ module, whose output then forms the input to the decoder $De$ to generate the final target color image $\hat{y}_{trans}$. Eq. 1 is factorized to

$$\begin{aligned} f_{sim} =& Sim(EnD(d), \{EnD(d_{r_0}), EnD(d_{r_1}); \\ & EnC(y_{r_0}), EnC(y_{r_1})\}), \\ P_{em} =& Em(x_{r_0} \odot y_{r_0}, x_{r_1} \odot y_{r_1}), \\ \hat{y}_{trans} =& De(Mid(EnL(x) \oplus f_{sim}, P_{em})), \end{aligned} \quad (2)$$

where $\odot$ means concatenation, and $\oplus$ means element-wise summation. The design of our color transform network ensures that the coloring process considers both the local similarity with the reference image regions and the global color style of the reference images.

The color transform network discriminator $D$ takes as input an image pair of line art image $x$ and corresponding color image, either real color image $y$ from the training set or $\hat{y}_{trans}$ produced by the generator $G$. It is trained to solve a binary classification task to determine whether the color image is real or fake (generated). The discriminator is trained in an adversarial way to make the

generated images indistinguishable from real color animation images.

We apply a 3D convolutional generation adversarial network to make the coloring result temporally coherent to learn the temporal relationship between the target coloring image and the reference images. The generator takes the reference and target line art and color image pairs as its input and generates the target coloring results in $\hat{y}$ as well as the beginning and the end reference frames. The input and output are both put in chronological order. The discriminator is trained to perform a binary classification to determine whether the color image sequence is real or fake. In order to reduce the training time and parameter amount of 3D convolution, we use the learnable gated temporal shift module [36] (LGTSM) based on the temporal shift module (TSM) in both the generator and discriminator. The LGTSM structure uses 2D convolution to achieve the effect of 3D convolution with guaranteed performance.

### 3.3 Distance attention layer

To cope with larger changes between adjacent frames in line art sketches, existing work [7] learns the matching of adjacent frames by increasing the number of intermediate residual blocks in the generator network. However, our method needs to match references with target line art drawing, which can have even more significant differences. Therefore, we utilize the global relationship learned by non-local neural networks [37] used in the work of video colorization [27].

Compared with matching features from grayscale images [27], matching features from line art images do not directly obtain satisfactory results because of the large deformations of lines of objects. We expect the network to learn the confidence of the correspondences simultaneously. Therefore, we make the distance attention layer to additionally learn masks of the features of reference images, which adaptively select the positions in the reference color images with the highest matching confidence. In the last stage of this module, we dynamically combine the matching-based transform feature from the whole reference images and the color features, where the color features are selected by the masks based on the matching confidence, and the matching-based transform feature provides features learned from the whole reference images.

The overall structure of the distance attention layer is shown in Fig. 3. We calculate the similarity of the high-dimensional features extracted from the target line art image and the reference line art images at a global scale. The module uses two learned internal masks $m_i$ and $n_i$:

$$m_i = \sigma(Conv(f_{d_i} \odot f_d)) \tag{3}$$

$$n_i = \sigma(Conv(f_{d_i} \odot f_d)) \tag{4}$$

where $f_d$ is the features from the target, and $f_{d_i}$ is the $i$-th reference image ($i = 0, 1$). The mask $m_i$ is used to select new features $f_{my_i} = f_{y_i} \otimes m_i$ from the reference color image features $f_{y_i}$, and $n_i$ is used to combine the feature matching information $f_{mat}$ with the reference color image features $f_{my_i}$.

The matching-based transform feature $f_{mat}$ is obtained by estimating the similarity between the features of the target and reference line art images. To make matching more effective, all the input line art images are first turned into distance field maps, which are used for extracting feature maps using the encoder $EnD$. Similarly, the color information of reference images $y_{r_0}$ and $y_{r_1}$ are extracted through encoders $EnC$. Let $W$ and $H$ be the width and height of the input images, and $\tilde{W} = W/4$, $\tilde{H} = H/4$, and $C$ are the width, height, and channel number of the feature maps. The feature map size is reduced to $1/4$ of the input image size to make a reasonable computation resource demand of similarity calculation. To reduce the complexity of global feature matching, we apply $1 \times 1$ convolutions to reduce the number of channels for feature maps to $\tilde{C} = C/8$, and reshape them to size $\tilde{W}\tilde{H} \times \tilde{C}$. The similarity map $M_i$ ($\tilde{W}\tilde{H} \times \tilde{W}\tilde{H}$) measures the similarity between features at different locations of the feature maps of the target and the $i$-th reference image. It is obtained through matrix multiplication: $M_i = f_d \cdot f_{d_i}^T$. We concatenate the matrices for all the reference images and apply softmax to $\{M_i\}$ to form the matching matrix $\tilde{M}$ of size $\tilde{W}\tilde{H}K \times \tilde{W}\tilde{H}$, $K = 2$, represents the number of reference images. Similarly, we apply $1 \times 1$ convolutions to reduce the channels of the color feature map of the reference images $f_{my_i}$, and reshape and concatenate them to form reference color matrix $f_C$ of size $\tilde{C} \times \tilde{W}\tilde{H}K$. The output of the module, $f_{mat} = f_C \cdot \tilde{M}$, represents the matching-based transform feature which transforms the color information from the reference images to the target based on the local similarity.

Overall, we use the following approach to get the final color similarity feature $f_{sim}$:

$$f_{sim} = \frac{1}{2}\sum_{i=0}^{1}((1 - n_i) \otimes f_{mat} + f_{my_i} \otimes n_i) \tag{5}$$

### 3.4 Loss Function

We define our loss function based on the following loss terms to enforce the color coherence between similar regions and penalize the color style difference with the reference images. Meanwhile, the generation result is divided into multiple images in the refinement network, and the following loss terms are applied. Finally, the calculated loss values of multiple images are averaged to obtain the final loss value.

**L1 loss**. To encourage the generated results to be similar to the ground truth, we use the pixel level L1 loss measuring the difference between the network generated result $\hat{y}$ and the ground truth color image $y$:

$$\mathcal{L}_{L1} = \|y - \hat{y}\|_1 \tag{6}$$

**Perceptual loss**. In order to ensure that the generated results are perceptually consistent with ground truth images, we use the perceptual loss introduced in [39]:

$$\mathcal{L}_{perc} = \sum_{i=1}^{5} \frac{1}{N_i}\|\Phi_{\hat{y}}^i - \Phi_y^i\|_2^2 \tag{7}$$

where $\Phi^i$ ($i = 1, \ldots, 5$) represents the feature map extracted at the ReLU $i\_1$ layer from the VGG19 network [40]. For our work, we extract feature maps from the VGG19 network layers ReLU 1_1; ReLU2_1; ReLU 3_1; ReLU 4_1 and ReLU 5_1. $N_i$ represents the number of elements in the $i$-th layer feature map.

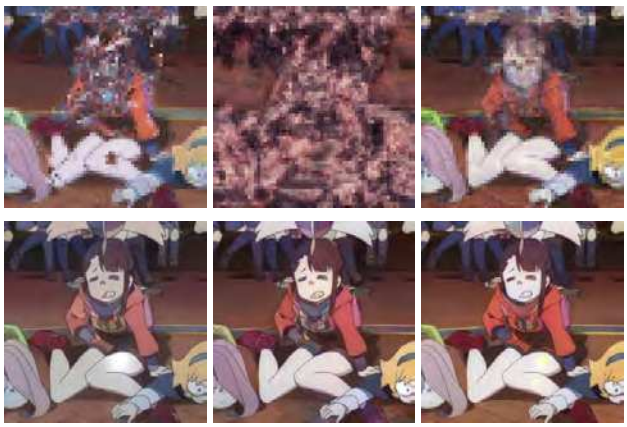
**Style loss**. Similar to image style transfer work [7], [41], [42], we calculate the similarity of the Gram matrices on the high-dimensional feature maps of the generated image and the ground truth to encourage the generated result to have the same style as the ground truth.

$$\mathcal{L}_{style} = \sum_{i=1}^{5} \|G(\Phi_{\hat{y}}^i) - G(\Phi_y^i)\|_1, \tag{8}$$


Fig. 4: Network model component analysis.

(a) (b) (c) (d) (e) (f) (g)

Fig. 5: Intermediate results. (a) Reference color image 1, (b) Reference color image 2, (c) Latent decoder image $\hat{y}_{Sim}$, (d) Latent decoder image $\hat{y}_{Mid}$, (e) Color image without the refinement network, (f) Final color image, (g) Ground truth.



(a) (b) (c)

Fig. 6: $Sim$ module reconstruction results. The first row presents the $Sim$ module reconstruction results, and the second row shows the network generation results. (a) All convolutional layers use spectral normalization; (b) None of convolutional layers use spectral normalization; (c) Only Encoders and Decoder(but not embedder) use spectral normalization.

where $G(\cdot)$ calculates the Gram matrix for the input feature maps. We use the VGG19 network to extract image feature maps from the same layers for calculating the style loss.

**Latent constraint loss**. In order to improve the stability of the generated effect, inspired by [5], [43], in addition to constraining the final generation results, we introduce further constraints on intermediate results of the network. Specifically, we add multi-supervised constraints to the distance attention layer output $f_{sim}$ and $Mid$ module output $f_{mid}$.

To make perceptual similarity measured more easily, $f_{sim}$ and $f_{mid}$ first pass through latent decoders (involving a convolution layer) to output 3-channel color images $\hat{y}_{sim}$ and $\hat{y}_{mid}$. We then use L1 loss to measure their similarity with the ground truth as follows:

$$\mathcal{L}_{latent} = \|y - \hat{y}_{sim}\|_1 + \|y - \hat{y}_{mid}\|_1 \tag{9}$$

**Adversarial Loss**. The adversarial loss promotes correct classification of real images ($y$) and generated images ($\hat{y}$).

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_{(x,y)}[\log D(x, y)] + \mathbb{E}_{(x,\hat{y})}[\log(1 - D(x, \hat{y}))] \tag{10}$$

**Overall objective loss function**. Combining all of the above loss terms together, we set the optimization goal for our model:

$$\mathcal{L}_x = \lambda_{perc}\mathcal{L}_{perc} + \lambda_{style}\mathcal{L}_{style} + \lambda_{latent}\mathcal{L}_{latent} + \lambda_{GAN}\mathcal{L}_{GAN} + \lambda_{L1}\mathcal{L}_{L1} \tag{11}$$

where $\lambda$ controls the importance of terms. We set $\lambda_{perc} = 1$, $\lambda_{style} = 1000$, $\lambda_{latent} = 1$, $\lambda_{GAN} = 1$, $\lambda_{L1} = 10$. Since the resulted style loss value is relatively small in our experiments, we set its weight as 1000, to make its contribution comparable with the GAN loss.

### 3.5 Implementation details

Our network consists of two sub-networks, color transform network and refinement network. We first train the color transform network to ensure the network generates plausible coloring results. Then we fix the color transform network parameters and the optimize refinement network parameters to refine the temporal consistency between the coloring result and the reference images.

The color transform network is composed of a generator network and a discriminator network. The generator network consists of encoders, a distance attention layer, middle residual

(a) (b) (c) (d) (e) (f) (g) (h)

Fig. 7: Comparison with existing methods. (a) the reference image; (b) results of cGAN-based [6]; (c) results of Deep Image Analogy [23]; (d) results of Two-Stage method [19]; (e) results of ARDSC [38] (f) results of TCVC [7]; (g) our results; (h) Ground truth.


Fig. 8: Video sequence coloring result comparison. We compare the method with cGAN-based [6], Deep Image Analogy [23], Two-Stage [19], TCVC [7] on a long sequence of coloring results. We show several frames from the long sequence at an equal interval.

convolution blocks, a Decoder, and an Embedder. The encoders for line art images, distance field maps and color images share the same network architecture. They are composed of 3 convolution layers and use instance normalization since colorization should not affect the samples in the same batch. They utilize the ReLU activations. We have 8 middle residual blocks [8] with AdaIN [44] as the normalization layer and ReLU activations. The Decoder comprises 4 convolutional layers with instance normalization and ReLU activation. Before the convolution of Decoder, we use the nearest neighbor upsampling to enlarge the feature map by 2 times along each spatial dimension. It will eliminate the artifacts of the checkerboard pattern in the generated results of GAN [45]. The Embedder consists of 5 convolution layers followed by a mean operation along the sample axis. Specifically, it maps multiple reference images to latent vectors and then averages them to get the final style latent vector. The affine transformation parameters are adaptively computed using the style latent vector by a two-layer fully connected network. Meanwhile, encoders and Decoder apply spectral normalization [46] to their convolutional layers to increase the stability of training. The discriminator network structure is similar to Embedder, consisting of 5 layers of convolutions. At the same time, in addition to the last layer of convolution, the discriminator adds spectral normalization [46] to other convolutional layers. It utilizes the Leaky LeakyReLU activations.

The temporal refinement network is composed of a generator and a patch discriminator. The generator comprises an Encoder with 3 3D gated convolutional layers and a Decoder with 4 dilated 3D gated convolutional layers and 3 3D gated convolutional layers. In the last 3 convolutional layers of Decoder, we use the nearest neighbor upsampling to enlarge feature maps by 2 times along each spatial dimension. The patch discriminator is composed of 5 3D convolutional layers. The spectral normalization is applied to both the generator and discriminator to enhance training stability.

## 4 Experiments

We evaluate the line art video coloring results on our line art colorization dataset. We show that our model outperforms other

| Setting | MSE↓ | PSNR↑ | SSIM↑ | FID↓ |
|---|---|---|---|---|
| No Sim | 0.0220 | 17.27 | 0.77 | 48.37 |
| No Emb | 0.0149 | 19.92 | 0.86 | 26.97 |
| No Line Art | 0.0198 | 18.71 | 0.77 | 37.06 |
| No Distance | 0.0126 | 20.46 | 0.84 | 30.75 |
| No Perc. Loss | 0.0122 | 20.57 | 0.83 | 33.65 |
| No Latent Loss | 0.0125 | 20.51 | 0.84 | 30.86 |
| No Style Loss | 0.0123 | 20.56 | 0.84 | 34.41 |
| No Temporal | 0.0111 | 21.60 | 0.86 | 27.67 |
| Full | **0.0101** | **22.81** | **0.87** | **26.92** |

TABLE 1: Ablation studies for different components on the coloring results, using mean MSE, PSNR, SSIM and FID.

| Method | MSE↓ | PSNR↑ | SSIM↑ | FID↓ |
|---|---|---|---|---|
| FUNIT [35] | 0.0946 | 10.54 | 0.35 | 83.17 |
| FCMAN [13] | 0.0797 | 11.69 | 0.57 | 87.31 |
| FVI [36] | 0.0379 | 15.49 | 0.62 | 106.17 |
| cGAN [6] | 0.0366 | 15.07 | 0.72 | 63.48 |
| TCVC [7] | 0.0426 | 14.95 | 0.73 | 50.75 |
| Two-stage [19] | 0.0352 | 14.91 | 0.65 | 42.08 |
| Deep IA [23] | 0.0478 | 15.36 | 0.67 | 38.22 |
| ARDSC [38] | 0.0356 | 19.21 | 0.80 | 34.47 |
| Ours | **0.0101** | **22.81** | **0.87** | **26.92** |

TABLE 2: Quantitative comparison with [6], [7], [13], [19], [23], [35], [36], [38] using mean MSE, PSNR, SSIM and FID. Some examples of visual comparison are shown in Fig. 7.

methods both quantitatively and qualitatively. In the following, we first introduce the data collection details. Then we analyze the effectiveness of the various components of the proposed method, and report the comparison between our method and the current state-of-the-art methods both qualitatively and quantitatively.

### 4.1 Data Collection and Training

We extract video frames from selected animations and extract the line art images to form our dataset. We calculate a 768-dimensional feature vector of histograms of R, G, B channels for each frame. The difference between frames is determined by calculating the mean square error of the feature vectors, which is used for splitting the source animations into shots. When the difference between the neighboring frames is greater than 200, it is considered to belong to different shots. In order to improve the quality of the data, we remove shots in which the mean square errors between all frame pairs are less than 10 (as they are too uniform), and the shot with a length less than eight frames. Then we filter out video frames that are too dark or too faded in color. Finally, we get a total of 1096 video sequences from 6 animations, with 29,834 images. Each video sequence has 27 frames on average. These six animation videos include 1) Little Witch Academia; 2) Dragon Ball; 3) Amanchu; 4) SSSS.Gridman; 5) No.6; 6) Soul Eater. All the images are scaled to $256 \times 256$ size.

The number of frames in each video sequence can vary from $8$ to more frames. Thus we randomly extract eight successive frames from videos for network training. The first and last frames are used as reference images, and the intermediate frames are used as the target images. When testing other methods that only take one reference image, the first frame of the entire sequence is selected as the reference image to color the other frames. To evaluate the versatility, we choose the data of animation 1, a total of 416 video sequences, 11,241 images for training and testing the network. Specifically, $50\%$ of the animation 1 is used as the training data, and the remaining $50\%$ is used as the test data. Other animation data is mainly used for testing, apart from using a few sequences for fine-tuning.

We use the Adam optimizer and set the generator learning rate to $1 \times 10^{-4}$, the discriminator learning rate to $1 \times 10^{-5}$, $\beta_1$ to 0.5, $\beta_2$ to 0.999, batch size 4. The color transform network trained 40 epochs; temporal refinement network trained 10 epochs. The experiment is performed on a computer with an Intel i7-6900K CPU and a GTX 1080Ti GPU, and the training time is about two days. It takes 71ms to color a line art image.

### 4.2 Ablation Studies

We perform ablation studies to evaluate the contribution of each module and loss term. The quantitative results comparing our full pipeline with one component disabled are reported in Table 1, using standard metrics PSNR, MSE, SSIM, Fréchet Inception Distance (FID) [47]. This shows that the distance attention layer, Embedder, usage of distance field maps, perceptual loss, latent loss, style loss, and refinement network are all essential to the performance of our method. When testing, we use the first and last frames of the video sequences as reference images to color all the intermediate frames. The visual comparison of an example is shown in Figure 4. Figure 4 shows that the $Sim$ module dramatically contributes to the generation result. The $Emb$ module constrains the generation result to match the color style of the reference images (see the region of the hair of the character). The temporal refinement network ensures that the colors of the resulting images are more chronologically coherent. We conducted a further experiment to validate the importance of each part of the network by coloring the target frame using features learned with only specific sub-modules. As shown in Figure 5, when the target image and the reference images have large deformation, just using features from $Sim$ is insufficient to generate accurate results, and adding $Emb$ preserves the global color style of reference images very well.

In experiments, we found that adding spectral normalization to the network can improve the stability of the training, but adding the spectral normalization to the Emb module of the color transform network will cause the generation results to be dim and the colors not bright. Figure 6 shows the results of $Sim$ reconstruction and network generation in the following three cases. The color transform network all adds spectral normalization, none adds spectral normalization, and only the $Emb$ module does not add it. We can see from Figure 6 that the final result of adding the spectral normalization network to the $Emb$ module is dim, and $Sim$ cannot learn specific color information without adding it. In addition, note that the output size of the Sim module is $1/16$ of the size of the original image. When visualizing the reconstructed image from Sim in the same size as the original image in Figure 6, we resize the result by 16 times, which makes the first row of images look blocky.

### 4.3 Comparisons on Line Art Colorization Method

We compare our approach to the state-of-the-art line art image coloring methods, cGAN-based [6], Deep Image Analogy (IA) [23], Two-Stage method [19], Augmented-self Reference and Dense Semantic Correspondence (ARDSC) [38] and TCVC [7]. For fairness, we use the training set described in Sec. 4.1 for all methods and then evaluate them on the same test set. Only one reference image is used for each sequence, as other methods cannot take multiple references. In order to conduct a fair comparison, for the two reference images needed in our network, we both use the first frame of a video sequence. Figure 7 shows the coloring results


Fig. 9: Comparison with colorization methods for grayscale images, including FUNIT [35], FCMAN [13], and FVI [36].

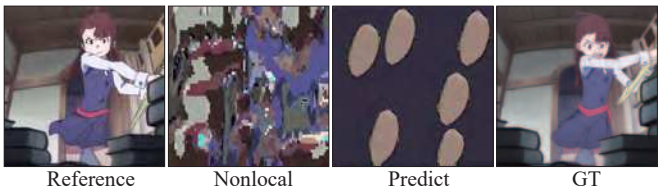



Fig. 10: Coloring results of DEVC [27]. We show the intermediate result of the nonlocal network and the final prediction of the whole network with reference images.

of all the methods. We further use several standard metrics to quantitatively evaluate the coloring results, which are presented in Table 2. The experimental results show that our method not only gets better coloring results but also keeps the style consistent with the reference image.

### 4.4 Comparisons on Grayscale Image Colorization Method

We compare our method with other colorization methods for cartoon animation, as shown in Figure 8. For a fair comparison, we select 32 sequences from animation 2 to fine-tune the models of all the methods and then test them on the remaining sequences. When coloring each sequence using cGAN-based [6], Deep Image Analogy [23], and Two-Stage Generation [19] methods, the first frame is selected as the reference image to color the remaining images in the sequence. For TCVC [7], the first frame of a sequence is used as the input condition for coloring the second frame, and then the generated result is used as the input condition of the following frame for all the subsequent frames. We can see that TCVC keeps propagating color errors to subsequent frames, leading to error accumulation in frames, while our method can effectively reduce the errors caused by color propagation. The cGAN-based and Deep Image Analogy methods have obvious color matching errors when there are large motions and shape deformations. Although our method might generate color bleeding in large deformed areas, it can still provide an acceptable preliminary coloring result. Also, note that the results of the Two-Stage methods do not well maintain the color consistency with the reference image, since they are not originally designed for animation coloring generation.

Our model trained on only one animation also generalizes well to new animations. When their color styles are different, our method benefits from using a small number of sequences from the new animation to fine-tune the parameters. To demonstrate it, we apply the network trained on animation 1 to colorize animation 2 (basic model) with and without fine-tuning using 32 sequences from animation 2 (fine-tuned model). We compare the results with TCVC [7] using the same settings. As shown in Table 4, our method achieves better coloring results after fine-tuning the network with only a small number of new animation data. Table 3 shows the quantitative testing results of our method and TCVC on other 4 animation videos where the models are fine-tuned by 32 video sequences. Our method outperforms TCVC [7] by a large margin in all settings.

| animation | Method | MSE↓ | PSNR↑ | SSIM↑ | FID↓ |
|---|---|---|---|---|---|
| 3 | [7] | 0.0345 | 15.23 | 0.66 | 76.34 |
| | Ours | **0.0105** | **20.91** | **0.83** | **51.99** |
| 4 | [7] | 0.0412 | 14.52 | 0.66 | 99.13 |
| | Ours | **0.0090** | **22.16** | **0.85** | **56.30** |
| 5 | [7] | 0.0455 | 14.67 | 0.59 | 95.68 |
| | Ours | **0.0095** | **21.91** | **0.82** | **58.27** |
| 6 | [7] | 0.0585 | 13.14 | 0.62 | 91.38 |
| | Ours | **0.0113** | **20.97** | **0.83** | **54.52** |

TABLE 3: Quantitative evaluation of different animation videos, using 32 video sequences to fine tune network parameters.

| Method | MSE↓ | PSNR↑ | SSIM↑ | FID↓ |
|---|---|---|---|---|
| Basic [7]- | 0.0523 | 13.32 | 0.63 | 78.64 |
| Fine-tuned [7]- | 0.0282 | 16.17 | 0.71 | 72.12 |
| Basic ours- | 0.0132 | 19.57 | 0.85 | 66.72 |
| Fine-tuned ours- | **0.0073** | **23.03** | **0.89** | **33.71** |
| Basic [7]+ | 0.0951 | 10.71 | 0.56 | 86.04 |
| Fine-tuned [7]+ | 0.0584 | 12.94 | 0.59 | 77.81 |
| Basic ours+ | 0.0197 | 18.06 | 0.82 | 68.23 |
| Fine-tuned ours+ | **0.0138** | **20.50** | **0.86** | **37.79** |

TABLE 4: Quantitative evaluation with [7] using mean MSE, PSNR, SSIM and FID on animation 2. "-" indicates a short sequence of length 8 is used for testing; "+" indicates the entire sequence is tested, regardless of the length of the sequence.

We compared our method with existing grayscale image colorization methods, including the limited data colorization method (FCMAN) [13], the deep exemplar-based video colorization method (DEVC) [27], the reference baseline method FUNIT [35] and the free-form video inpainting method (FVI) [36]. For fair comparisons, we use the same dataset to train the models of all the methods and use the first frame of each sequence as the reference image when testing. The numerical comparison results are shown in Table 2, and some colorization results are shown in Fig. 9. The coloring result of FUNIT [35] lost the curvilinear information and messed up the positions of color regions, which means that it fails to match the line art features. The FCMAN [13] method neither transfers correct colors to the line art image very well nor preserves the edges of the input line art image. The result of FVI [36] simply copies the colors of the reference image, rather than coloring regions considering line movements. We re-trained the network parameters using the code provided by the authors of DEVC [27], and Fig. 10 shows their nonlocal network output and final coloring result. The network does not generate effective coloring results, which may be because VGG19 trained by normal images cannot cope well with line art images. In addition, accurate colorization is difficult for DEVC to achieve since the intermediate feature matching results are poor.

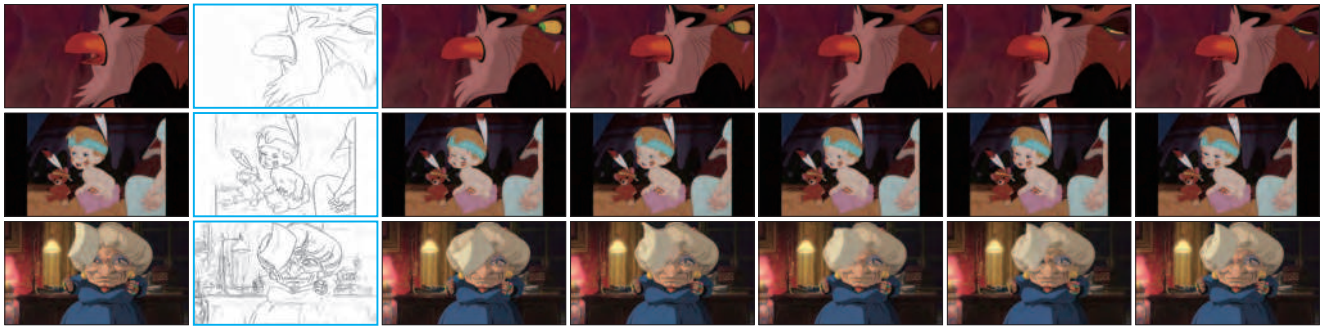



Fig. 11: Comparison with video interpolation methods DAIN [48] and DAVI [49]. Note: The input line drawings with blue borders are only inputted to our method.

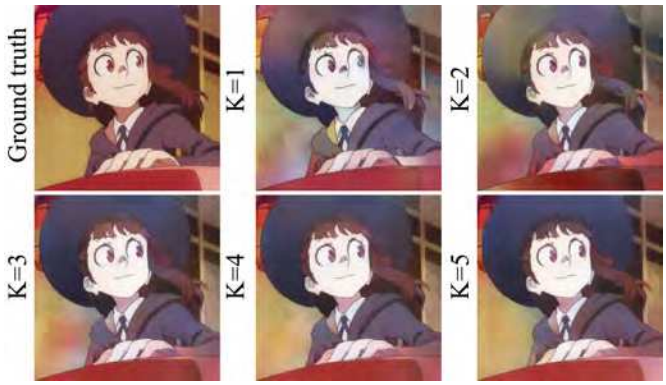



Fig. 12: Comparison of the coloring results with different numbers of reference images $K$.

| $K$ | MSE↓ | PSNR↑ | SSIM↑ | FID↓ |
|---|---|---|---|---|
| 1 | 0.0170 | 19.87 | 0.83 | 31.09 |
| 2 | 0.0111 | 21.59 | 0.86 | 27.67 |
| 3 | 0.0084 | 22.76 | 0.88 | 24.18 |
| 4 | 0.0069 | 23.44 | 0.89 | 22.20 |
| 5 | **0.0062** | **23.83** | **0.89** | **21.79** |

TABLE 5: Quantitative comparison of different numbers of reference images $K$ on the coloration of video sequences.

### 4.5 Comparisons on Video Frame Interpolation

Although our goal and required input are not the same as the video intermediate frame generation task, we can make the results of our method and animation video interpolation methods comparable with suitable input. It is not feasible to test interpolation methods using two key frames at the two ends of a given cartoon sequence as it is likely that changes within the sequence are too much for interpolation. So we alternatively applied our method to the cartoon interpolation task to evaluate the capability of our network to learn to transfer colors. We compared with the methods of Depth-aware video frame interpolation (DAIN) [48] and Deep Animation Video Interpolation (DAVI) [49]. We trained and tested our method and the above two methods using the interpolation dataset, ATD-12K [49], where we used the extracted sketch images as our line art input when testing. The final visual comparison results and quantitative comparison results are shown in Figure 11. We can see that our results are qualitatively and quantitatively better than other methods. Despite that the comparison is not fully fair for the other two methods since their input does not include the line art images as guidance, it still demonstrates that our solution is more suitable to colorize the in-between frames when the line art images are accessible.

### 4.6 More Discussions

**Number of reference images.** Our method can be easily generalized to use multiple reference images. Here we test the effect of feeding different numbers of reference images to train the model for long coloring sequences. We divide a video sequence into multiple segments according to different numbers of reference images to achieve this. We use the frames where the sequence is divided as the reference images to color the in-between frames. We set the reference image number as $K = 1, 2, \ldots, 5$ respectively. The coloring results are shown in Figure 12, where the quality is improved with an increasing number of reference images. The quantitative evaluation is shown in Table 5. We can see that more reference images generally lead to better results, but when $K$ exceeds 3, the improvement of the coloring results starts to saturate. To balance the reference image numbers and the coloring quality, we choose $K$ as 3 when applying our method to long sequence colorization.

**Model fine-tuning.** For new animations, if the style is similar to the training dataset, we can get high-quality color results. Nevertheless, for animation with a different style, a small amount of data is needed to fine-tune the network to learn new styles better. To fully evaluate the effects of fine-tuning a trained model using the data from the target animation, we further conducted the following experiments: We trained a baseline model with a larger diversity using the animations of the training set of ATD-12K [49]. Then we tested the models fine-tuned by different numbers of sequences (Seq-Num) to achieve satisfactory coloring results in the new animation's style, where the number of sequences is set to 8, 16, 24, 32, 40, respectively. The network generally produces better coloring results with increasing Seq-Num, and the results stabilize when Seq-Num exceeds 32. Thus, to balance the coloring results and required training data amount, we set Seq-Num to 32 by default. In addition, we tested the fine-tuned model by modifying the color distribution of the reference images. Specifically, the reference image is converted to the HSV color space. Then its H-channel is added by 90/180/270 (treated as a circular quantity) to change the tone of the reference image. Qualitative and quantitative evaluations are shown in Fig. 13 and Table 6, respectively. We can see that the color distribution that has never been seen during training can be accurately transferred to the target images. It demonstrates that the model successfully learned the desired color matching information, rather than overfitting the spatial relationship between color and local features.

Moreover, we found the model trained on a dataset with a larger diversity is better than trained using a dataset from the same animation. Table 7 shows the quantitative comparison results of


Fig. 13: Comparison of coloring results of the network after using different numbers of images to fine-tune the network with reference images of different tones. The baseline model is trained using ATD-12K.

| Setting | Hue=0 | | Hue=90 | | Hue=180 | | Hue=270 | |
|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ | PSNR↑ | SSIM↑ |
| Baseline | 20.26 | 0.791 | 20.01 | 0.794 | 20.04 | 0.793 | 19.96 | 0.794 |
| Seq-Num8 | 20.56 | 0.823 | 20.06 | 0.826 | 20.25 | 0.825 | 20.24 | 0.828 |
| Seq-Num16 | 21.39 | 0.836 | 20.78 | 0.837 | 20.89 | 0.834 | 20.76 | 0.837 |
| Seq-Num24 | 21.79 | 0.841 | 20.90 | 0.841 | 21.21 | 0.843 | 21.04 | 0.84 |
| Seq-Num32 | 21.79 | 0.848 | **21.02** | 0.847 | **21.31** | 0.846 | 21.02 | 0.847 |
| Seq-Num40 | **21.90** | **0.851** | 21.01 | **0.847** | 21.21 | **0.847** | **21.09** | **0.850** |

TABLE 6: Quantitative evaluation of fine-tuning with different sequence numbers using reference images of different tones. The baseline model is trained using ATD-12K.

| Setting | MSE↓ | PSNR↑ | SSIM↑ |
|---|---|---|---|
| Baseline | 0.0204 | 18.19 | 0.78 |
| Seq-Num8 | 0.0204 | 18.19 | 0.78 |
| Seq-Num16 | 0.0157 | 19.44 | 0.80 |
| Seq-Num24 | 0.0147 | 19.91 | 0.81 |
| Seq-Num32 | **0.0127** | 20.21 | **0.83** |
| Seq-Num40 | 0.0128 | **20.55** | 0.83 |

TABLE 7: Quantitative evaluation of fine-tuning with different sequence numbers, where the baseline is trained on animation 1.

fine-tuning the model trained on animation 1 using data from other animations. The generalizability of the fine-tuned model originally trained using ATD-12K is better than those initially trained using only one animation. One reason is that the model trained with large diversity learns the matching relationship between the target image and the reference image better, especially in sophisticated areas. For example, the hair of animation characters has different styles in different animations, where the model trained on a diverse range of animations can correctly identify the corresponding area and make a better matching across animation frames.

### 4.7 User Study and Hand-drawn Line Art Colorization

To better evaluate our method, we conduct a user study to subjectively assess the visual quality of colorized line art images and the effectiveness of our method. In our experiment, we randomly selected 10 animation sequences and only used one reference color image to color the line art images. We first invite ten men and ten women with an average age of 24. Then the videos colored by different methods are randomly shuffled and displayed to the user. The end user evaluates the results of different colorization methods in the following dimensions:

1) *Content consistency*: The consistency between the color of results and the reference image.
2) *Interpolation quality*: The smoothness of the transition between successive frames of colored results.
3) *Content quality*: The amount of expressed details.
4) *Boundary consistency*: The consistency between the lines of the colored results and the input line art images.
5) *Overall quality*: The overall plausibility of colored results.

In this survey, users were asked to give each result a score between 0 and 1 for each dimension. Figure 15 shows the radar chart of different methods. We can see that the results of our method can get a higher score than other methods in all the evaluation dimensions.

We invited professional cartoonists to provide hand-drawn sketches of colored cartoon images to test our method on real-world data. We also provide several colorization results on the line art images created by the manga line extraction method (MLE) [50]. Since the sizes of the collected line art images are not uniform, the images to be colored are uniformly reduced to the


Fig. 14: Comparison using different types of input sketches. We automatically extract sketches by MLE [50], and also invite cartoonists to draw line art images as our input. Here, the original color images are all from ATD-12K [49] dataset.

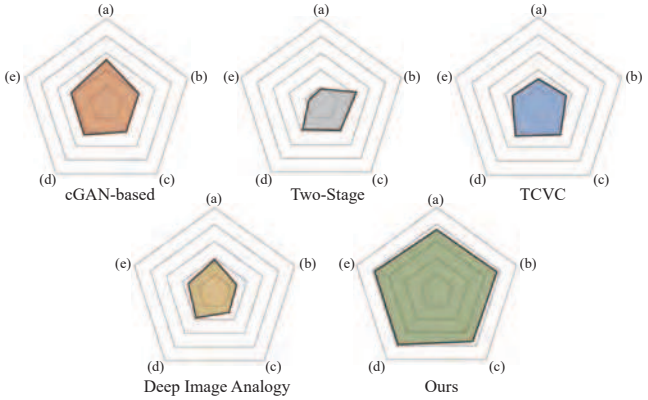



Fig. 15: Visualization of the user study result. (a) Color consistency; (b) Smoothness; (c) Details; (d) Line Consistency; (e) Plausibility.

network input size before being input to the network. Figure 14 shows the coloring results of our method with two different line art image inputs. Here, all the original cartoon images are from the ATD-12K [49] dataset.

We also search for the line art images drawn by professional cartoonists from the image search engine, and then invited professional colorization workers to match the color of the key frames of line art videos. We first use the colored key frames to fine-tune the pre-trained model and then use the fine-tuned model to color other line art images. Since the size of the collected line art images is not uniform, the image to be colored is uniformly reduced to the network input size before being input to the network. Then colored images are restored to their original size through the animation image super-resolution method proposed by Chao et al. [51]. Please see our coloring results in the supplementary materials.

## 5 Limitations

Our method performs global matching on image features to achieve long-distance feature matching. However, if there is a new character showing up in the frames to be colored, our method cannot handle it well, as shown in the first row of Fig. 16. In addition, if the motion of the character is too fast, our method could generate area matching errors and color overflow, as shown in the second row of Fig. 16. Further research is needed to enhance the temporal consistency for fast-moving objects where discontinuity exists between successive frames. In addition, our colorization examples could have some artifacts if the model cannot well learn the correspondences between largely deformed lines caused by the character's actions, especially when only a small number of images are available for model fine-tuning. Although our method outperforms the state-of-the-art methods, it may still not reach the production requirement.

## 6 Conclusions

In this paper, we propose a new line art video colorization method with a few reference images. The architecture exploits both local similarity and global color styles to colorize the line art images and adopt a 3D convolutional module to refine the temporal consistency of the final result. Our method does not use sequential propagation to consecutive color frames, avoiding error accumulation. We collect a dataset for evaluating our method of line art colorization. Extensive experiments show that our method performs well for coloring video sequences from the same animation as the training data and generalizes well to new animation videos. Better results can be obtained by fine-tuning with a small number of reference images of the new animation.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61972379, No. 62102403 and No. 61872440), the Science and Technology Service Network Initiative, Chinese Academy of Sciences (No. KFJ-STS-QYZD-2021-11-001), Royal Society Newton Advanced Fellowship (No. NAF\R2\192151), Royal Society (No. IES\R1\180126), the Youth Innovation Promotion Association CAS and the Marsden Fund Council managed by Royal Society of New Zealand (No. MFP-20-VUW-180).


Fig. 16: Failure cases. The first frame (t=0) and the last frame (t=1) are used as reference images. Here, we show 4 color frames (t=0.2,...,0.8) from the entire colored video.

## REFERENCES


[1] Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng. Manga colorization. *ACM Trans. Graph.*, 25(3):1214–1220, July 2006.

[2] Daniel Sýkora, John Dingliana, and Steven Collins. Lazybrush: Flexible painting tool for hand-drawn cartoons. *Computer Graphics Forum*, 28(2):599–608, 2009.

[3] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Trans. Graph.*, 23(3):689–694, August 2004.

[4] Chie Furusawa, Kazuyuki Hiroshiba, Keisuke Ogaki, and Yuri Odagiri. Comicolorization: Semi-automatic manga colorization. In *SIGGRAPH Asia 2017 Technical Briefs*, SA '17, pages 12:1–12:4, New York, NY, USA, 2017. ACM.

[5] Lvmin Zhang, Yi Ji, Xin Lin, and Chunping Liu. Style transfer for anime sketches with enhanced residual u-net and auxiliary classifier gan. In *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 506–511. IEEE, 2017.

[6] Paulina Hensman and Kiyoharu Aizawa. cgan-based manga colorization using a single training image. *CoRR*, abs/1706.06918, 2017.

[7] Harrish Thasarathan, Kamyar Nazeri, and Mehran Ebrahimi. Automatic temporally coherent video colorization. *CoRR*, abs/1904.09527, 2019.

[8] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–189, 2018.

[9] Alexandrina Orzan, Adrien Bousseau, Pascal Barla, Holger Winnemöller, Joëlle Thollot, and David Salesin. Diffusion curves: A vector representation for smooth-shaded images. *Commun. ACM*, 56(7):101–108, July 2013.

[10] Haichao Zhu, Xueting Liu, Tien-Tsin Wong, and Pheng-Ann Heng. Globally optimal toon tracking. *ACM Trans. Graph.*, 35(4), July 2016.

[11] Yuanzheng Ci, Xinzhu Ma, Zhihui Wang, Haojie Li, and Zhongxuan Luo. User-guided deep anime line art colorization with conditional adversarial networks. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, pages 1536–1544, New York, NY, USA, 2018. ACM.

[12] Domonkos Varga, Csaba Attila Szabó, and Tamás Szirányi. Automatic cartoon colorization based on convolutional neural network. In *Proceedings of the 15th International Workshop on Content-Based Multimedia Indexing*, CBMI '17, pages 28:1–28:6, New York, NY, USA, 2017. ACM.

[13] Seungjoo Yoo, Hyojin Bahng, Sunghyo Chung, Junsoo Lee, Jaehyuk Chang, and Jaegul Choo. Coloring with limited data: Few-shot colorization via memory augmented networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11283–11292, 2019.

[14] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[15] Holger WinnemöLler, Jan Eric Kyprianidis, and Sven C Olsen. Xdog: an extended difference-of-gaussians compendium including advanced image stylization. *Computers & Graphics*, 36(6):740–753, 2012.

[16] Henry Kang, Seungyong Lee, and Charles K. Chui. Coherent line drawing. In *Proceedings of the 5th International Symposium on Non-photorealistic Animation and Rendering*, NPAR '07, pages 43–50, New York, NY, USA, 2007. ACM.

[17] lllyasviel. Sketchkeras. https://github.com/lllyasviel/sketchKeras, 2018. Accessed April 4, 2019.

[18] Kazuhiro Sato, Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa. Reference-based manga colorization by graph correspondence using quadratic programming. In *SIGGRAPH Asia 2014 Technical Briefs*, SA '14, pages 15:1–15:4, New York, NY, USA, 2014. ACM.

[19] Lvmin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji, and Chunping Liu. Two-stage sketch colorization. *ACM Trans. Graph.*, 37(6):261:1–261:14, December 2018.

[20] lllyasviel. style2paints. https://github.com/lllyasviel/style2paints, 2018. Accessed April 3, 2019.

[21] Shu-Yu Chen, Jia-Qi Zhang, Lin Gao, Yue He, Shihong Xia, Min Shi, and Fang-Lue Zhang. Active colorization for cartoon line drawings. *IEEE Transactions on Visualization and Computer Graphics*, 28(2):1198–1208, 2022.

[22] Lin Gao, Yan-Pei Cao, Yu-Kun Lai, Hao-Zhi Huang, Leif Kobbelt, and Shi-Min Hu. Active exploration of large 3d model repositories. *IEEE transactions on visualization and computer graphics*, 21(12):1390–1402, 2014.

[23] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *ACM Trans. Graph.*, 36(4):120:1–120:15, July 2017.

[24] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24:1–24:11, July 2009.

[25] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Trans. Graph.*, 23(3):689–694, August 2004.

[26] Varun Jampani, Raghudeep Gadde, and Peter V Gehler. Video propagation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 451–461, 2017.

[27] Bo Zhang, Mingming He, Jing Liao, Pedro V Sander, Lu Yuan, Amine Bermak, and Dong Chen. Deep exemplar-based video colorization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8052–8061, 2019.

[28] Chenyang Lei and Qifeng Chen. Fully automatic video colorization with self-regularization and diversity. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[29] Ondřej Jamriška, Šárka Sochorová, Ondřej Texler, Michal Lukáč, Jakub Fišer, Jingwan Lu, Eli Shechtman, and Daniel Sýkora. Stylizing video by example. *ACM Trans. Graph.*, 38(4), July 2019.

[30] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):978–994, 2011.

[31] Keyu WU, Lingchen YANG, Hongbo FU, and Youyi ZHENG. ihairrecolorer: deep image-to-video hair color transfer. *Science China Information Sciences*, 64(11), November 2021.

[32] Satoshi Iizuka and Edgar Simo-Serra. Deepremaster: Temporal source-reference attention networks for comprehensive video enhancement. *ACM Trans. Graph.*, 38(6), November 2019.

[33] Daniel Sýkora, Jan Buriánek, and Jiří Žára. Unsupervised colorization of black-and-white cartoons. In *Proceedings of the 3rd International Symposium on Non-photorealistic Animation and Rendering*, NPAR '04, pages 121–127, New York, NY, USA, 2004. ACM.

[34] Wengling Chen and James Hays. Sketchygan: Towards diverse and realistic sketch to image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9416–9425, 2018.

[35] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. *CoRR*, abs/1905.01723, 2019.

[36] Ya-Liang Chang, Zhe Yu Liu, Kuan-Ying Lee, and Winston Hsu. Free-form video inpainting with 3d gated convolution and temporal patchgan. *In Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[37] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

[38] Junsoo Lee, Eungyeup Kim, Yunsung Lee, Dongjun Kim, Jaehyuk Chang, and Jaegul Choo. Reference-based sketch image colorization using augmented-self reference and dense semantic correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5801–5810, 2020.

[39] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[41] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

[42] Pierre Wilmot, Eric Risser, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *CoRR*, abs/1701.08893, 2017.

[43] Hyunsu Kim, Ho Young Jhoo, Eunhyeok Park, and Sungjoo Yoo. Tag2pix: Line art colorization using text tag with secat and changing loss. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9056–9065, 2019.

[44] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[45] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[46] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[47] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[48] Wenbo Bao, Wei-Sheng Lai, Chao Ma, Xiaoyun Zhang, Zhiyong Gao, and Ming-Hsuan Yang. Depth-aware video frame interpolation. In *IEEE Conferene on Computer Vision and Pattern Recognition*, 2019.

[49] Li Siyao, Shiyu Zhao, Weijiang Yu, Wenxiu Sun, Dimitris Metaxas, Chen Change Loy, and Ziwei Liu. Deep animation video interpolation in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6587–6595, 2021.

[50] Chengze Li, Xueting Liu, and Tien-Tsin Wong. Deep extraction of manga structural lines. *ACM Transactions on Graphics (SIGGRAPH 2017 issue)*, 36(4):117:1–117:12, July 2017.

[51] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

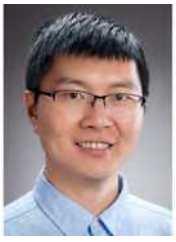

**Min Shi** is an associate professor in the school of Control and Computer Engineering, North China Electric Power University. She received her Ph.D. degree in computer science and technology from Chinese Academy of Sciences in 2013. Her research interests include cloth simulation, data analysis and visualization and virtual reality.

**Jia-Qi Zhang** received the BS degree in software engineering from North China Electric Power University. He is currently working toward the PhD degree in the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University. His research interests include virtual reality and computer vision.

**Shu-Yu Chen** received the PHD degree in computer science and technology from University of Chinese Academy of Sciences. She is currently working as a research associate in Institute of Computing Technology, Chinese Academy of Sciences. Her research interests include computer graphics.

**Lin Gao** received the bachelor's degree in mathematics from Sichuan University and the PhD degree in computer science from Tsinghua University. He is currently an Associate Professor at the Institute of Computing Technology, Chinese Academy of Sciences. He has been awarded the Newton Advanced Fellowship from the Royal Society and the Asia Graphics Association young researcher award. His research interests include computer graphics and geometric processing.

**Yu-Kun Lai** received his bachelor's and Ph.D. degrees in computer science from Tsinghua University, China, in 2003 and 2008, respectively. He is currently a Professor at the School of Computer Science & Informatics, Cardiff University. His research interests include Computer Graphics, Computer Vision, Geometry Processing and Image Processing. He is on the editorial boards of *Computer Graphics Forum* and *The Visual Computer*.



**Fang-Lue Zhang** is currently a Lecturer with Victoria University of Wellington, Wellington, New Zealand. He received the Doctoral degree from Tsinghua University in 2015. His research interests include image and video editing, computer vision, and computer graphics. He is a member of IEEE and ACM. He received Victoria Early-Career Research Excellence Award in 2019 and Marsden Fast-Start Grant from New Zealand Royal Society in 2020.