

# Deep Line Art Video Colorization with a Few References

Min Shi<sup>†</sup>, Jia-Qi Zhang<sup>†</sup>, Shu-Yu Chen, Lin Gao\*, Yu-Kun Lai, Fang-Lue Zhang

**Abstract**—Coloring line art images based on the colors of reference images is an important stage in animation production, which is time-consuming and tedious. In this paper, we propose 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 constraint network. The color transform network takes the target line art images as well as the line art and color images of one or more reference images as input, and generates corresponding target color images. To cope with larger differences between the target line art image and reference color images, our architecture utilizes non-local similarity matching to determine the region correspondences between the target image and the reference images, which are used to transform 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 style embedding vector that describes the global color style of the references, extracted by an embedder. The temporal constraint network takes the reference images and the target image as its input in chronological order, and learns the spatiotemporal features through 3D convolution 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 amount 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 state-of-the-art methods and other baselines.

**Index Terms**—line art colorization, color transform, 3D convolution, few shot learning

## 1 INTRODUCTION

THE process of animation production requires high labor input. Coloring is one of the important 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. But it still needs a lot of repetitive work in each frame. Therefore, automatic methods for coloring line art images based on reference color images are highly demanded, which can greatly reduce the costs in animation production.

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 line art videos based on given reference images is challenging. Firstly, 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 compared to grayscale images, making colorization a harder task. Also, in practice when coloring a new anime video from line art images, only a few examples are available. This requires the model to be trainable with a small number of samples, when applied to color animation videos with a new color style.

In this paper, we propose a deep architecture to automatically color line art videos based on reference images. 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 adopting a similarity-based color transform 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 further use a 3D convolutional network

<sup>†</sup> Both authors contributed equally to this research.

\* Corresponding Author.

- M. Shi and J. Zhang are with North China Electric Power University, Beijing 102206, China. E-mail: {zhangjiaqi, shi\_min}@ncepu.edu.cn
- S. 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: {chenshu, 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. Zhang is with School of Engineering and Computer Science, Victoria University of Wellington, New Zealand. E-mail: fanglue.zhang@ecs.vuw.ac.nz



Fig. 1: Comparison of different line art image extraction methods. (a) original color image; (b) Canny [12]; (c) XDoG [13]; (d) Coherent Line Drawing [14]; (e) SketchKeras [15]; (f) distance field map from SketchKeras results.

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 amount of samples when dealing with animations of a new color style.

## 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. With the development of deep learning, researchers have used it to achieve automatic or interactive coloring [10], [11]. 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. [16] 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 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. [17] 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 [18]. 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 learn relationships between grayscale and color images, rather than line art images. In addition, Liao et al. [19] use the PatchMatch [20] 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 [21], [22], [23], [24] learns temporal color consistency by calculating optical flow. However, animated video frames do not in general hold pixel-level continuity, causing optical flow algorithms to fail to get good results. Skora et al. [25] 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 (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 similarity-based color transform 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 not sufficient to obtain accurate coloring result 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.

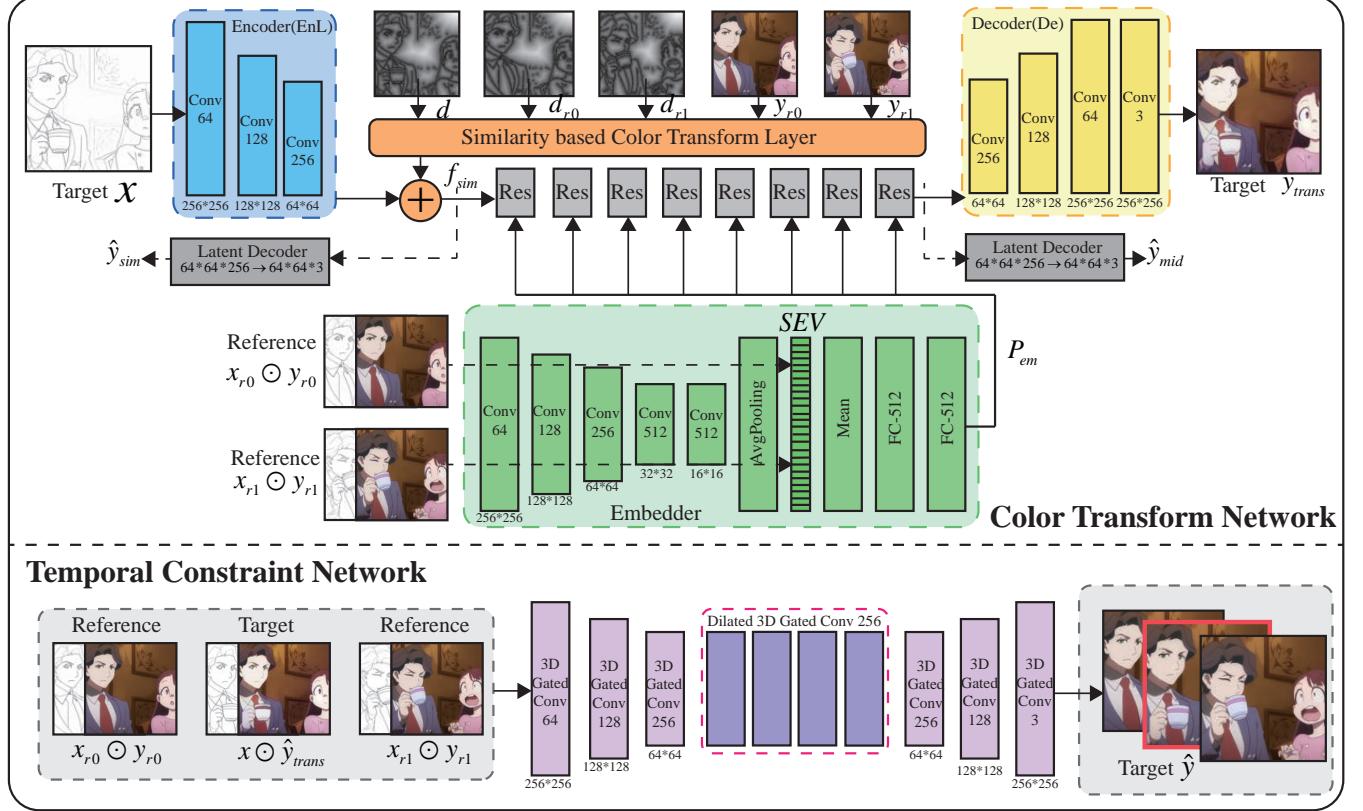


Fig. 2: The overall structure of the our network. The color transform network combines the latent features of global color style extracted from reference color images with latent features extracted from the input line art image to be colored. Finally, the temporal constraint network ensure the coloring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

### 3.1 Data Preparation

Since there is no public dataset for the evaluation of 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 section 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 [12] are quite different from the line drawing images drawn by cartoonists. Other extractors, like XDoG edge extraction operator [13] and the Coherent Line Drawing [14] method, are too sensitive to user-specified parameters. Sketch-Keras [15] 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 [26], 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.

### 3.2 Overall Network Architecture

Our network consists of two sub-networks, color transform network and temporal constraint network. The color transform network consists of a conditional generator  $G$  and an adversarial

discriminator  $D$ . Inspired by the image-to-image translation framework [27], our generator learns global style information through Embedder, and provides local color features through the similarity-based color transform 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)  $\{x_{r0}, x_{r1}; d_{r0}, d_{r1}; y_{r0}, y_{r1}\}$  as input and produces the target color image  $\hat{y}_{trans}$  via

$$\hat{y}_{trans} = G(x, d, \{x_{r0}, x_{r1}; d_{r0}, d_{r1}; y_{r0}, y_{r1}\}) \quad (1)$$

$r$  represents the reference object.  $\hat{y}_{trans}$  is the preliminary coloring result generated by the color transform network, where "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 mainly composed of six parts, encoders, a similarity-based color transform 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  $EnD(d_{r0}, d_{r1})$ , and the reference color images  $EnC(y_{r0}, y_{r1})$ .  $EnL$ ,  $EnD$ , 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 to the similarity-based color transform layer to obtain the local color features  $f_{sim}$  of the target line art image, which provide a local colorization for the target image. After concatenating the line art images and the reference images,

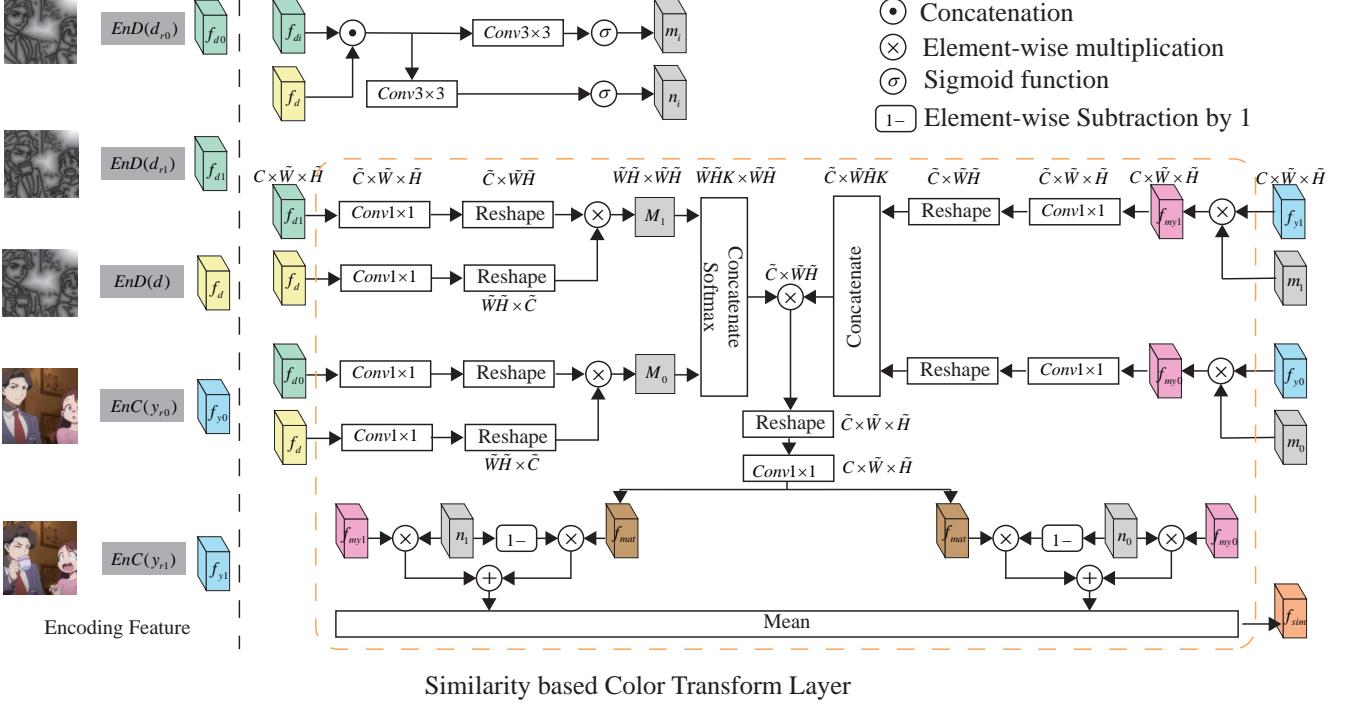


Fig. 3: Similarity-based color transform layer. It first extracts the feature maps of the input distance field maps of the target and reference images through the encoder  $EnD$ . Then the similarity map is calculated, which is for transforming the color features of the reference images to those of the target image  $f_{mat}$ . Finally, the matching-based transform feature  $f_{mat}$  and the color feature of the reference images  $f_{myi}$  are dynamically combined to obtain the local color feature map of the target image  $f_{sim}$ .

we also feed them separately into an Embedder module [27] 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)  $Em(x_{r1} \odot y_{r1}, \dots, x_{rK} \odot y_{rK})$ . 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

$$\begin{aligned} f_{sim} &= Sim(EnD(d), \{EnD(d_{r0}), EnD(d_{r1}); \\ &\quad EnC(y_{r0}), EnC(y_{r1})\}), \\ P_{em} &= Em(x_{r0} \odot y_{r0}, x_{r1} \odot y_{r1}), \\ \hat{y}_{trans} &= De(Mid(EnL(x) \oplus f_{sim}, P_{em})), \end{aligned} \quad (2)$$

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 try to make

the generated images indistinguishable from real color animation images.

To make the coloring result temporally coherent, we apply a 3D convolutional generation adversarial network 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 result  $\hat{y}$  as well as the reference beginning and the end 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 are real or fake. In order to reduce the training time and parameter amount of 3D convolution, we use the learnable gated temporal shift module [28] (LGTSM) based on temporal shift module (TSM) in both of the generator and discriminator. The LGTSM structure uses 2D convolution to achieve the effect of 3D convolution with guaranteed performance.

### 3.3 Similarity-based Color Transform Layer

To cope with larger changes between adjacent frames in 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. We therefore utilize the global relationship learned by non-local neural networks [29] used in the work of video colorization [23]. Compared with matching features from grayscale images [23], matching features from line art images does 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 similarity-based color transform 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 feature, 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 similarity-based color transform 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 on a global scale. The module uses two learned internal masks  $m_i$  and  $n_i$ :

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

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

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 features matching information  $f_{mat}$  with the reference color image features  $f_{my_i}$ .

The matching-based transform feature  $f_{mat}$  is obtained through estimating the similarity between the features of target and reference line art images. 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  $M_k$  ( $\tilde{W}\tilde{H} \times \tilde{W}\tilde{H}$ ) 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  $\tilde{W}\tilde{H}K \times \tilde{W}\tilde{H}$ . Similarly, we apply  $1 \times 1$  convolutions to reduce the channels of the color feature map of the reference images, 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.

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

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

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.

### 3.4 Loss

To enforce the color coherence between similar regions and penalize the color style difference with the reference images, we define our loss function based on the following loss terms. Meanwhile,

in the temporal constraint network, the generation result of the network is divided into multiple images and the following loss terms is 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  $y$ :

$$\mathcal{L}_{L1} = \|y - \hat{y}\|_1 \quad (6)$$

**Perceptual loss.** In order to ensure that the generated results are perceptually consistent with ground truth images, we use the perceptual loss introduced in [30]:

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

where  $\Phi^i$  ( $i = 1, \dots, 5$ ) represents the feature map extracted at the ReLU  $i\_1$  layer from the VGG19 network [31]. 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 conversion work [7], [32], [33], 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.

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

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 perceptual loss.

**Latent constraint loss.** In order to improve the stability of the generated effect, inspired by [5], [34], 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 similarity-based color transform 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:

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

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

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

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

$$\begin{aligned} \mathcal{L}_x = & \lambda_{perc} \mathcal{L}_{perc} + \lambda_{style} \mathcal{L}_{style} + \\ & \lambda_{latent} \mathcal{L}_{latent} + \lambda_{GAN} \mathcal{L}_{GAN} + \lambda_{L1} \mathcal{L}_{L1} \end{aligned} \quad (11)$$

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.



Fig. 4: Network model component analysis.

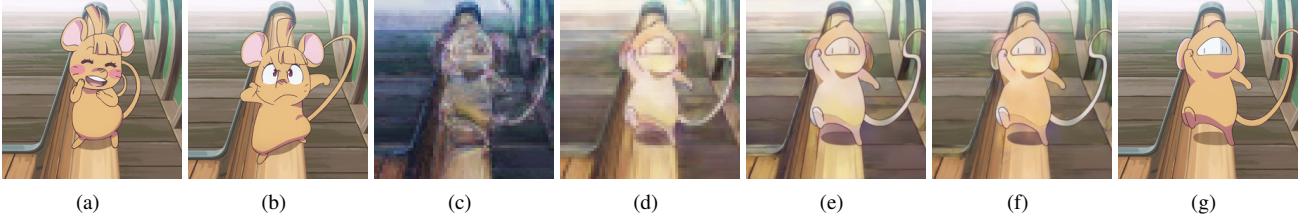
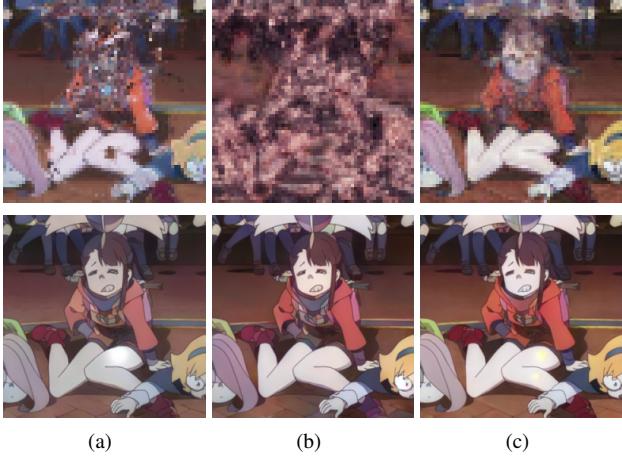
Fig. 5: Intermediate results. (a) Reference color image 1, (b) Reference color image 2, (c) Latent decoder image  $\hat{y}_{Sim}$ , (d) Latent decoder image  $\hat{y}_{Mid}$ , (e) No temporal color image, (f) Final color image, (g) Ground truth.

Fig. 6: Sim module reconstruction results. The first raw represents the network generation results, and the second raw represents the Sim module reconstruction results. (a) All convolutional layers use spectral normalization; (b) All convolutional layers do not use spectral normalization; (c) Only Encoders and Decoder use spectral normalization.

### 3.5 Implementation details

Our network consists of two sub-networks, color transform network and temporal constraint network. We first train the color transform network to ensure that the network generates plausible coloring results. Then the color transform network parameters are fixed, and the temporal constraint network parameters are

optimized 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 similarity-based color transform layer, middle residual 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 be affected by the samples in the same batch. They utilize the ReLU activations. We have 8 middle residual blocks [8] with AdaIN [35] as the normalization layer and ReLU activations. The Decoder is composed of 4 convolutional layers, also with the instance normalization and the ReLU activations. Before the convolution of Decoder, we use the nearest neighbor upsampling to enlarge the feature map by 2 times along each spatial dimension. This will eliminate the artifacts of checkerboard pattern in the generated results of GAN [36]. 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 [37] to their convolutional layers to increase the stability of training. The discriminator network structure is similar with Embedder, which consists of 5 layers of convolutions. At the same time, in addition to the last layer of convolution, the discriminator adds spectral normalization [37] to other convolutional layers. It utilizes the Leaky LeakyReLU

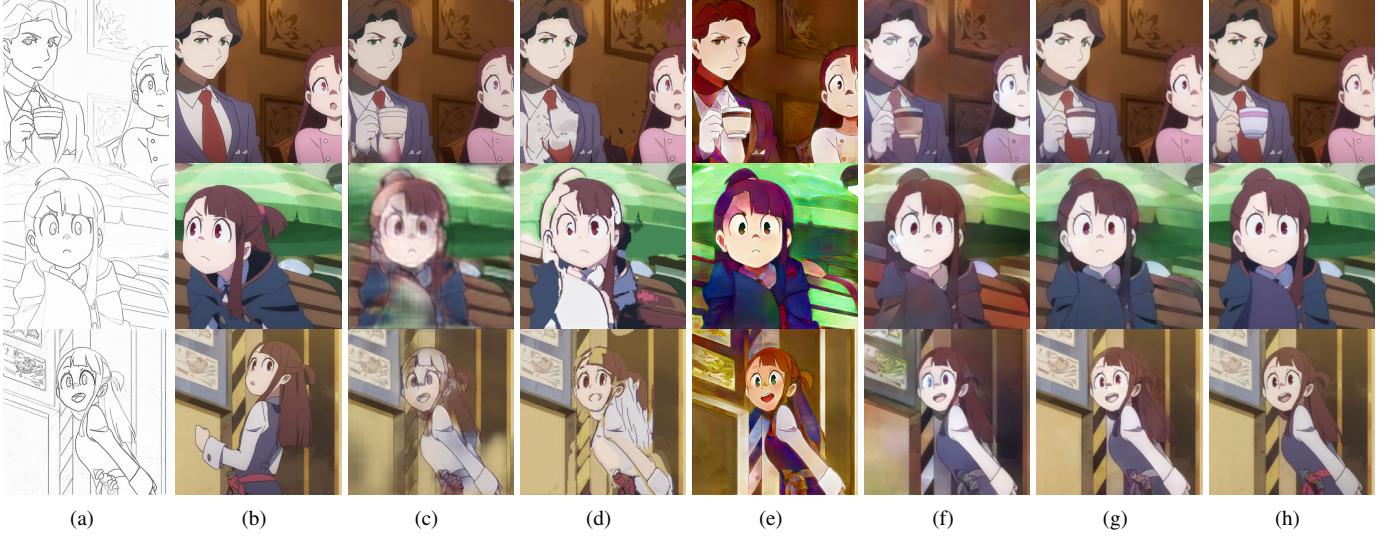


Fig. 7: Comparison with existing methods. (a) input sketch image; (b) the reference image; (c) results of cGAN [6]; (d) results of Deep Image Analogy [19]; (e) results of Two-stage method [17]; (f) results of TCVC [7]; (g) our results; (h) Ground truth.

activations.

The temporal constraint network is composed of a generator and a patch discriminator. The generator is composed of 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 the Decoder, we use 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 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 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 training 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 pairs of frames are less than 10 (as they are too uniform), and the shot with a length less than 8 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 a total of 29,834 images. Each video sequence has 27 frames on average. These 6 anime videos include: 1) Little

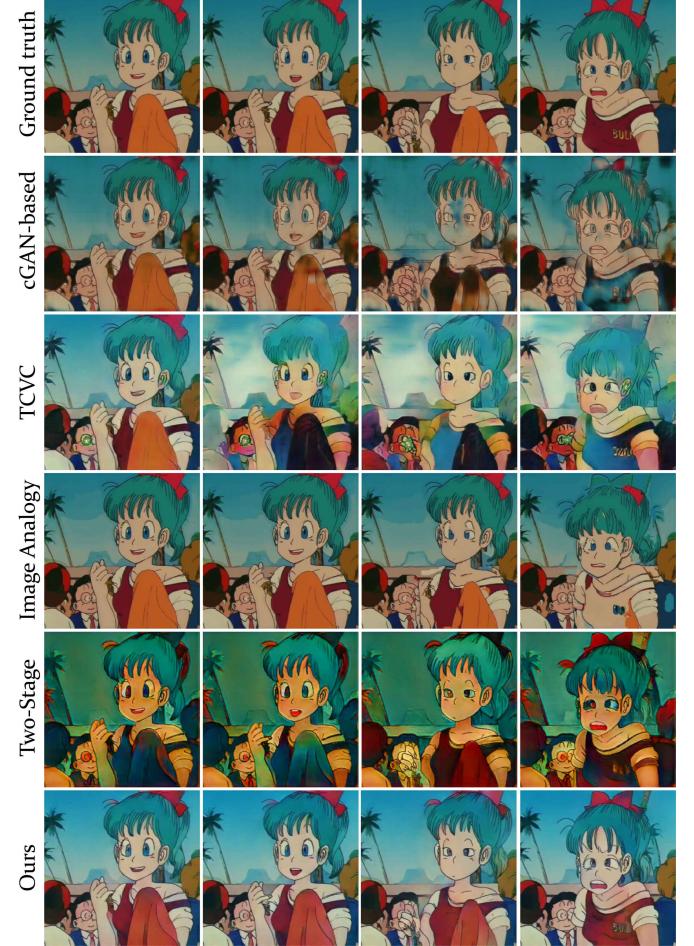


Fig. 8: Video sequence coloring result comparison. We compare the method with cGAN-based [6], Deep Image Analogy [19], Two-stage [17], TCVC [7] on a long sequence of coloring results. We show several frames from the long sequence at an equal interval.

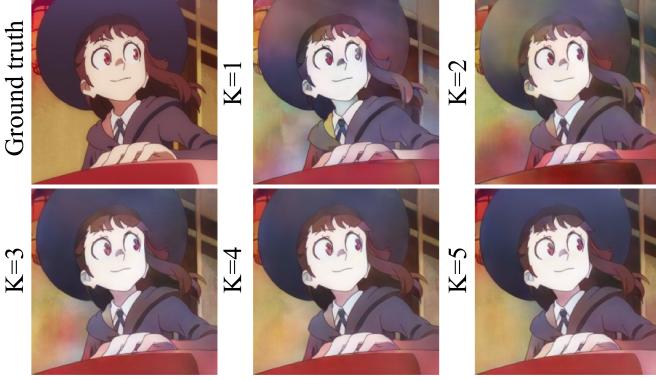


Fig. 9: Comparison of the coloring results with different numbers of reference images  $K$ .

Setting	MSE $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	FID $\downarrow$
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	<b>0.0101</b>	<b>22.81</b>	<b>0.87</b>	<b>26.92</b>

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

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.

To evaluate the versatility, we choose the data of Anime 1, a total of 416 video sequences, 11,241 images for training and testing the network. Specifically, 50% of the data in the Anime 1 is used as the training data, and the remaining 50% is used as the test data. Other anime 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 constraint 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 2 days. It takes 71ms to color a line art image.

The number of frames in each video sequence could vary from 8. Thus we randomly extract 8 successive frames from videos for network training, where the first and last frames are used as reference images, and the intermediate frames are used as the target images. When testing other methods, the first frame of the entire sequence is selected as the reference image to color the other frames.

## 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) [38]. This shows that the similarity-based color transform layer, Embedder, usage of distance field maps, perceptual loss, latent loss, style loss, and filter are all essential to the

Method	MSE $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	FID $\downarrow$
cGAN [6]	0.0366	15.07	0.72	63.48
TCVC [7]	0.0426	14.95	0.73	50.75
Two-stage [17]	0.0352	14.91	0.65	42.08
Deep IA [19]	0.0478	15.36	0.67	38.22
Ours	<b>0.0104</b>	<b>22.41</b>	<b>0.86</b>	<b>27.93</b>

TABLE 2: Quantitative comparison with [6], [7], [17], [19] using mean MSE, PSNR, SSIM and FID. Some examples of visual comparison are shown in Fig. 7.

Method	MSE $\downarrow$	PSNR $\uparrow$	SSIM $\uparrow$	FID $\downarrow$
Basic [7]-	0.0523	13.32	0.63	78.64
	0.0282	16.17	0.71	72.12
	0.0132	19.57	0.85	66.72
	<b>0.0073</b>	<b>23.03</b>	<b>0.89</b>	<b>33.71</b>
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+	<b>0.0138</b>	<b>20.50</b>	<b>0.86</b>	<b>37.79</b>

TABLE 3: Quantitative evaluation with [7] using mean MSE, PSNR, SSIM and FID on animation Dragon Ball. “-” indicates a short sequence of length 8 is used for testing; “+” indicates the entire sequence is tested, regardless of the length of the sequence.

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 greatly 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 human hair). The temporal constraint network ensures that the colors of 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 certain sub-modules. As shown in Figure 5, when the target image and the reference image 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 of them adds spectral normalization, and only the *Emb* module does not add it. It can be seen 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.

## 4.3 Comparisons of Line Art Image Colorization

We compare our approach to the state-of-the-art line art image coloring methods, cGan-based [6], Deep Image Analogy (IA) [19], Two-stage method [17], 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, and 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 of all the

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

TABLE 4: Quantitative evaluation of different anime videos, using 32 video sequences to fine tune network parameters.

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 maintains the style consistency with the reference image.

#### 4.4 Comparisons on Video Colorization

We compare our method with other colorization methods for animation coloring, as shown in Figure 8. We select 32 sequences from Anime 2 to fine-tune models of all methods and then test them on the remaining sequences. When coloring each sequence, the first image of the sequence is treated as the reference image, and the coloring results of the remaining images are evaluated. For the cGan-based [6], Deep Image Analogy [19], and Two-stage generation [17] methods, the selected reference image is used to color the remaining images in the sequence. For TCVC [7], the selected frame in a sequence is first used as the condition input to color the second frame, and the subsequent frames use the generation result of the previous frame as the condition input. It can be seen that TCVC [7] keeps propagating color errors to subsequent frames, leading to error accumulation, while our method can effectively avoid this problem. The cGan-based [6] and Deep Image Analogy [19] methods cannot handle the line art image coloring with large differences in shape and motion. The result of Two-stage generation [17] does not maintain consistency with the reference images well.

Our model trained on 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 for fine tuning. To demonstrate this, we apply the network trained on Anime 1 to color Anime 2 (basic model), with optional fine-tuning of the network using 32 sequences from Anime 2 (fine-tuned model). We compare the results with TCVC [7], using the same settings. As shown in Table 3, our method achieves better coloring results after fine-tuning the network with only a small amount of new animation data. Table 4 shows the quantitative testing results of our method and TCVC on other 4 anime videos where the models are fine-tuned by 32 video sequences. This demonstrates the strong generalizability of our network to unseen animation styles. Our method outperforms TCVC [7] by a large margin in all settings.

#### 4.5 More Discussions

**Number of reference images.** We test the effect of feeding different numbers of reference images to train the model for coloring long sequences. We divide a video sequence into multiple segments according to different numbers of reference images. We use the frames where the sequence is divided as the reference



Fig. 10: Comparison of coloring results of the network after inputting different numbers of images to fine tune the network.

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	<b>0.0062</b>	<b>23.83</b>	<b>0.89</b>	<b>21.79</b>

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

images to color the in-between frames. We set the reference image number as  $K = 1, 2, \dots, 5$  respectively. The coloring are shown in Figure 9, where the quality is improved with increasing number of reference images. The quantitative evaluation is shown in Table 5. It can be seen that more reference images generally lead to better results, but when  $K$  exceeds 3, the improvement of the coloring results starts to converge. To balance between the reference image numbers and the coloring quality, we choose  $K$  as 3 when applying our method to long time sequence colorization.

**Number of sequences for fine tuning.** For new animations, if the style is similar to the training dataset, we can get high quality color results. But for anime with a different style, a small amount of data is needed to fine tune the network to learn new styles. We test different numbers of sequences (Seq-Num) to fine tune the network parameters to achieve satisfactory coloring results in a new style, where Seq-Num is set to 8, 16, 24, 32, 40 respectively. The fine-tuning phase takes about 2 hours of training for 30 epochs. As shown in Figure 10 and quantitative comparison in Table 6, the network generally produces better coloring results with increasing Seq-Num, and the results stabilize when Seq-Num exceeds 32. Thus, to balance between the coloring results and required training data amount, we set Seq-Num to 32 by default.

Setting	MSE↓	PSNR↑	SSIM↑	FID↓
Seq-Num8	0.0204	18.19	0.78	85.81
Seq-Num16	0.0157	19.44	0.80	78.42
Seq-Num24	0.0147	19.91	0.81	77.12
Seq-Num32	<b>0.0127</b>	20.21	<b>0.83</b>	71.84
Seq-Num40	0.0128	<b>20.55</b>	0.83	<b>70.74</b>

TABLE 6: Quantitative evaluation of fine-tuning with different sequence numbers.

## 5 CONCLUSIONS

In this paper, we propose a new line art image video coloring method based on reference images. The architecture exploits both local similarity and global color styles for improved results. Our method does not use sequential propagation to color consecutive frames, which avoids error accumulation. We collect a dataset for evaluating line art colorization. Extensive evaluation shows that our method not only performs well for coloring on the same video used for training, but also generalizes well to new anime videos, and better results are obtained with fine-tuning with a small amount of data.

## REFERENCES

- [1] Yingge Qu, Tien-Tsin Wong, and Pheng-Ann Heng. Manga colorization. *ACM Trans. Graph.*, 25(3):1214–1220, July 2006.
- [2] Daniel Skora, 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 Hiroshima, 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] 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.
- [11] 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.
- [12] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.
- [13] 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.
- [14] 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.
- [15] Illyasviel. sketchkeras. <https://github.com/Illiyasviel/sketchKeras>, 2018. Accessed April 4, 2019.
- [16] 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.
- [17] 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.
- [18] Illyasviel. style2paints. <https://github.com/Illiyasviel/style2paints>, 2018. Accessed April 3, 2019.
- [19] 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.
- [20] 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.
- [21] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. *ACM Trans. Graph.*, 23(3):689–694, August 2004.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] 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.
- [26] Wengling Chen and James Hays. Sketchgan: Towards diverse and realistic sketch to image synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9416–9425, 2018.
- [27] 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.
- [28] 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.
- [29] 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.
- [30] 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.
- [31] 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.
- [32] 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.
- [33] Pierre Wilmot, Eric Risser, and Connelly Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *CoRR*, abs/1701.08893, 2017.
- [34] 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.
- [35] 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.
- [36] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.
- [37] 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.
- [38] 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.