The Visual Computer (2022) 38:1283–1300
https://doi.org/10.1007/s00371-021-02219-x

ORIGINAL ARTICLE

A GAN-based approach toward architectural line drawing colorization prototyping

Qian Sun[1] · Yan Chen[1] · Wenyuan Tao[1] · Han Jiang[1] · Mu Zhang[1] · Kan Chen[2] · Marius Erdt[2,3]

Accepted: 6 June 2021 / Published online: 23 July 2021

Abstract
Line drawing with colorization is a popular art format and tool for architectural illustration. The goal of this research is toward generating a high-quality and natural-looking colorization based on an architectural line drawing. This paper presents a new Generative Adversarial Network (GAN)-based method, named *ArchGANs*, including *ArchColGAN* and *ArchShdGAN*. *ArchColGAN* is a GAN-based line-feature-aware network for stylized colorization generation. *ArchShdGAN* is a lighting effects generation network, from which the building depiction in 3D can benefit. In particular, *ArchColGAN* is able to maintain the important line features and the correlation property of building parts as well as reduce the uneven colorization caused by sparse lines. Moreover, we proposed a color enhancement method to further improve *ArchColGAN*. Besides the single line drawing images, we also extend our method to handle line drawing image sequences and achieve rotation animation. Experiments and studies demonstrate the effectiveness and usefulness of our proposed method for colorization prototyping.

Keywords Architectural illustration · GAN · Colorization · Line drawing

1 Introduction

As a saying from Paul Rudolph in [35]: "the architectural drawing is the most eloquent tool a professional has to communicate design ideas." For architects, architectural illustration acts as an essential medium to clarify, communicate, or document designs. As shown in Fig. 1, line drawing with colorization is a popular and important technique for architectural illustration. From the engineering perspective of the architect, it is expressive and can show the essential building information, for example, lighting effects, color, material, layout, and structure. Moreover, from the art perspective of the architect, architectural illustration using line drawing with colorization is a significant art format to portray artistic concepts in architecture and convey an architect's ideas. From the application perspective of the architect, such format is also beneficial in many applications for demonstrating the specific architecture design essence, for example, urban planning, marketing, competition, and design proposal.

For the purpose of exchanging ideas among designers and trying out various schemes of color, colorization prototyping is frequently needed, particularly in the early architectural design stage. For example, colorization is often done according to a line drawing, and watercolor is the common tool for colorization. This paper aims toward realizing watercolor alike colorization prototyping. Tedious efforts or high art skills are often required in the typical semi-manual/manual colorization methods [10]. Computer graphics (CG) methods, e.g., physics-based simulation [11] and non-photorealistic rendering (NPR) [2], often need to craft a specific technique or require an expensive computational cost, in order to generate the desired result. Current learning-based methods for colorization, e.g., style-transfer with examples [14], are mainly used for transferring general visual features, such as texture and color. These methods are hence more preferable for capturing the overall viewing perceptual similarities and appearances.

However, it is not easy for those methods to well preserve the essential underlying line features, which can convey the basic and key building structure and layout information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00371-021-02219-x.

✉ Kan Chen
kchen1@e.ntu.edu.sg

[1] Tianjin University, Tianjin, China

[2] Fraunhofer Singapore, Singapore, Singapore

[3] Nanyang Technological University, Singapore, Singapore

Fig. 1 Line drawing with watercolor colorization in real architectural illustration examples

for architectural illustration as well as exhibit representative drawing styles, e.g., in Fig. 1, the line features are important, because not only they can depict the important architectural parts and main shapes, but also line crossings often appear at the corners and such crossing is a typical style in a perspective drawing to be used as a coordinate reference.

Furthermore, the existing methods are mostly aiming at dealing with the type of input lines and results, which are in the 2D manga-character fashion [21]. But, unlike this type of lines, there are often larger blank areas and relatively sparser lines in the architectural line drawings. They are commonly used to represent building parts such as walls and roofs. When coloring the big empty areas, this limit of line information might cause fragmentation and ambiguity. In the existing methods, this can easily result in unnaturalness and unevenness in the big blank areas in the colorization results. Moreover, different from other art formats (such as portrait painting), one common property of architectural illustration is the strong correlation between building components, for example, one column of windows should be the same color in principle. Nevertheless, this property is not directly considered in the existing methods.

For designers, it is an advantage to have lighting effects added to an architectural illustration. This can help to portray a 3D building with various colors under various lighting conditions, but the existing 2D methods usually do not support 3D depiction for buildings.

Animating a static colored building (or moving its corresponding camera) can enhance the viewer's perceptual experience, e.g. offering the viewers a street walk-through around the building. It also provides additional options for designers to present the building, i.e., in architectural illustration, designers often want to illustrate the building in different viewing directions for the purpose of realizing a more comprehensive presentation. Animation in architectural illustration format such as line drawing with watercolor is also useful since it can present the essence and condensed information of a design idea as mentioned previously, it also provides a better perceptual and multi-view experience. Animation comprises a sequence of images, and these images may correspond to different camera views. In this case, the frame color coherency needs to be considered, e.g., the color of a wall should not change in different views, otherwise, the viewers can easily get confused. However, the change of views usually introduces more ambiguities and thus can increase the severity of the previously mentioned flaws, such as the unwanted blank areas and unclear features. Moreover, since there is some randomness when performing colorization each time, the same building parts can be colored differently in the frames. Unfortunately, the existing methods do not directly address this.

In this research, given a line drawing image as the input, a GAN-based method toward stylized watercolor alike colorization prototyping for architectural illustration is proposed. We also extend it to handle line drawing image sequences to generated building animation effects as an application of our method. We focus on the rotation animation effect, since the other common transformation operations, such as translation and scaling, are relatively straightforward to realize. The proposed method is well suited toward prototyping colorization effectively. It has the following main features.

(1) Toward achieving a stylized colorization based on a line drawing image of a building, a unified framework *ArchGANs* is proposed, which considers the line features and lighting effects. We explicitly realize the colorization and lighting effect using two generative adversarial networks (GANs), *ArchColGAN* and *ArchShdGAN*, respectively.
(2) *ArchColGAN* is proposed to generate the colorization. Based on a training dataset, it is able to learn and predict colors for the line drawing image input. In particular, by using its line-feature-aware network structure, *ArchColGAN* can preserve the line features in the resulting colored architectural illustration. The proposed approach can handle the line drawing images that contain sparse lines and generate the representative effect of line crossing at the corners.
(3) *ArchShdGAN* is proposed to generate the lighting effects. We utilized a simple and effective approach to represent and formulate the lighting effects to facilitate the generation.
(4) We propose a color enhancement method which is a complementary network to enhance the *ArchColGAN* results

in the attention fashion. We also propose a rotation animation effect generation network for the input of line drawing image sequences, which can ensure color consistency in the animated colorization. These methods are all GAN based and can be seamlessly integrated into *ArchGANs* framework.

(5) We conducted many experiments and studies to evaluate *ArchGANs* and the extensions. The results demonstrate that our method is effective, and compared to the conventional methods, it has improvements in overall quality.

Comparing to the earlier version [38], this extended version has the following new contributions.

(1) We propose a complementary color enhancement method by applying the attention idea in order to further reduce the unwanted blank areas and better preserve the line features if needed.
(2) We expand the input to handle line drawing image sequences and add a new effect: animation, which can be a helpful application. We propose incorporating Long Short Term Memory (LSTM) [18] networks to handle the frame color consistency.
(3) We add new results and studies to demonstrate the effectiveness and usefulness of our method.

2 Related work

The related work in stylized colorization based on Computer Graphics (CG) methods, Convolutional Neural Networks (CNNs), and Generative Adversarial Networks (GANs) is reviewed in this section, respectively.

2.1 Computer graphics-based methods

In this subsection, we mainly review the CG-based methods that are relevant to the watercolor alike colorization and line drawing.

There are many existing commercial solutions that have interactive colorization functionalities, for example, Corel©-Painter [10]. However, many similar solutions usually need a lot of manual work which can be tedious.

Visually realistic colorization results can be achieved using physical simulation-based methods, such as for the effects of oil painting and watercolor. Curtis et al. [11] performed a fluid simulation to model the water and pigment moving processes. Chu and Tai [8] as well as Van Laerhoven and Van Reeth [40] utilized the GPU computation for accelerating and generating realistic effects of watercolor. Nevertheless, the computational costs of such physical simulation-based methods are usually high.

Procedural colorization is another common approach. The typical approach is first processing and analyzing the input image to retrieve the information needed, and then applying various image filtering techniques to realize the brush stroke simulation. For example, a Sobel filter is used to simulate the darkening effects of the stroke edges [28], virtual painting knife [34] and content-dependent painterly rendering [19] are proposed for oil painting, and color scribbles with optimization are used for grayscale image colorization [13]. Please refer to [17] for a survey. Instead of simulating the physical process for watercolor, procedural colorization focuses on mimicking the watercolor effect appearance. In order to model the effect of watercolor, Bousseau et al. [2] combined a group of image filters and applied them to the results from 3D rendering. Luft and Deussen [31] proposed a method suitable for rendering plants with a watercolor effect. Luft et al. [32] applied a similar method to render CAD models with watercolor effect. These methods are basically aiming at rendering 3D models with non-photorealistic effects, however, designing a procedural to produce a particular effect is not always easy.

In general, CG-based methods can achieve realistic results, however, there are still some challenges that may be faced as mentioned above. On the other hand, our proposed method is data driven and GAN based, thus, some manual and procedural design efforts, as well as the run-time computational costs, can be reduced.

Line drawing is one of the non-photorealistic rendering effects, too. One of its applications is the tone shading (cel shading) used in games (e.g., [5]). Based on a 3D model, the common method for rendering its line drawing images is based on mathematically defining feature lines as points on the surface, which satisfy certain geometry constraints. Relief edges [27], ridge or valley lines [23,33], suggestive contours [12], photic extremum Lines [43], shadow abstraction [42], and silhouettes are some examples of the feature line definitions. In our training, we apply the splatting lines method [44] to generate line drawings of 3D buildings.

2.2 Convolutional neural networks-based methods

With the great advances in CNN, CNN has become to be the popular tool for many applications to solve synthesizing problems [37]. Gatys et al. [14] are pioneered in this, and they proposed the automatic art-to-image style transfer method. Then, a series of works extended this work. To enhance the image quality, Liao et al. [29] proposed using image analogy. Johnson et al. [22] introduced perceptual losses and Chen et al. [7] proposed Stylebank, for the purpose of improving the efficiency. A video style transfer extension was proposed by Chen et al. [6]. These approaches can handle style transfer tasks for many artistic styles, e.g., oil painting and watercolor. However, they often put the emphasis on transferring colors

and textures in a particular artistic style, at the same time trying to preserve the original image content, but not necessarily preserve the line features. Thus, these approaches cannot be directly used for architectural colorization, which includes a sense of the engineering ingredient and has imperative line features to be preserved.

2.3 Generative adversarial networks-based methods

Various GAN [15] methods have been proposed for image-to-image translation. The pix2pix approach by Isola et al. [21] trains with the image pairs to achieve convincing results for photo-to-map, photo-to-sketch, and photo-to-label translations. Zhu et al. [46] proposed an extension for multi-modal translation, namely BicycleGAN. Furthermore, many GAN methods have also been proposed to handle unpaired image translation, such as UNIT [30], CycleGAN [45], Disco-GAN [26], and MNUIT [20]. A number of methods were proposed to handle colorization for 2D manga style characters, e.g., CariGANs [4] and Tag2Pix [25]. However, different from 2D manga, architectural line drawing contains large empty regions (walls) that need to be evenly colored, as well as essential line features and sparse lines that need to be preserved. Convincing results can be achieved using those GAN-based methods, however, their main focus is basically still color or texture change, such as a horse to zebra. That is, the challenges raised by the properties of architectural line drawings are still not explicitly handled, which include the uneven colorization due to sparse lines as well as the lack of support for lighting effects and maintaining line features.

GAN can be also used for the task of video/frame prediction/generation, such as [9,36,39], recurrent neural networks such as LSTM can be also used for this task [3]. However, our work is different from these, because we focus on frame–frame architectural colorization consistency in the case of having different camera locations while realizing the even colorization and maintaining the line features.

3 Our method

3.1 Main structure and training dataset

3.1.1 Main structure

With the aim of effective colorization prototyping, the focus of this paper is toward automatically generating a natural-looking stylized colorization with user-specified light direction, from a given architectural line drawing as the input. We propose a new generative adversarial network (GAN) framework *ArchGANs* for tackling the following issues: the inadequately preserved line features and building part correlation, the undesired uneven colorization due to sparse lines, as well as the lack of plausible depiction of 3D lighting effects. The user can select the desired color and lighting direction from a set of pre-defined color schemes and lighting directions. Each color scheme with one lighting direction is trained as one model.

For many content generation tasks, the GAN-based methods [15]) have been proved to have good performance. As such, we adopt the GAN framework for our colorization generation. However, the learning-based colorization methods often require many example image pairs, for example, the pair of a line drawing and its colorization images. On the other hand, manually creating a number of such building image pairs under various conditions of lighting can be inefficient and tedious. Therefore, we propose decoupling the whole generation process into two branches of GAN networks with the similar architecture: stylized colorization (*ArchColGAN*) and lighting effect generation (*ArchShdGAN*) (Fig. 2). Our method can be more flexible thanks to this modular structure.

In the context of architectural illustration, *ArchGANs* learns a mapping $\phi : X \rightarrow Y$ from line drawing domain X to stylized colorization with lighting effects domain Y. Using this mapping, the input architectural line drawing $x \in X$ can be colored in the style with lighting effects as $y \in Y$. First, we train *ArchColGAN* to learn the translation from line drawing domain X to the domain of stylized colorization without lighting effect enhancement Y^C. That is, based on the input x, in order to generate a stylized colorization without lighting effect enhancement $y^C \in Y^C$, *ArchColGAN* learns the mapping: $\phi_C : X \rightarrow Y^C$.

In the results of *ArchColGAN*, few artifacts may happen, we hence propose a color enhancement network to further reduce the blank areas and enhance the line features. Moreover, we also propose a rotation animation generation network to extend *ArchColGAN* to handle line drawing image sequences while addressing the color consistency in frames. These two extension methods can be seamlessly integrated into *ArchColGAN*, however, they require extra steps and computations, they are essentially the complementary operations to meet the additional demands of the users. Their results can be combined with *ArchShdGAN*, too. Note that when we mention *ArchColGAN*, we refer to the *ArchColGAN* without enhancement.

ArchShdGAN learns a mapping $\phi_S : Y^C \rightarrow Y_S$. This mapping is used for the lighting effect of $y_S \in Y_S$ generation. Y_S is the lighting information domain for stylized colorization Y. The final result y can be obtained by integrating the output y_S with y^C.

3.1.2 Training dataset

Likewise, coloring each line drawing to create its corresponding stylized colorization pair is also inefficient and tedious. Furthermore, such image pairs with desired styles are rarely

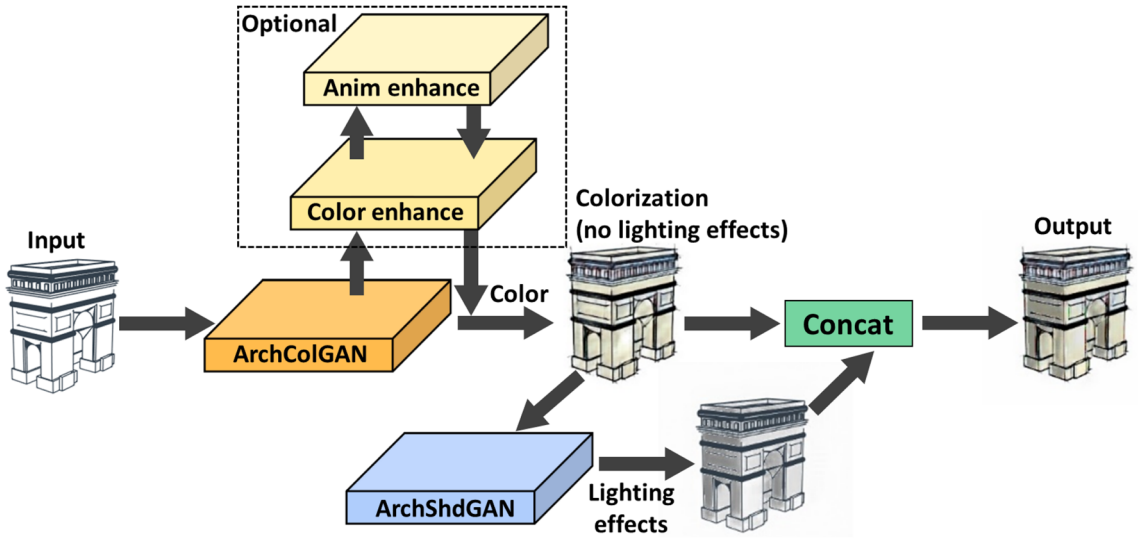


Fig. 2 Pipeline of the proposed *ArchGANs*

available. So constructing the training dataset is actually not easy. We have had the following observation based on our architectural illustration collection which contains a large number of images. The shapes of buildings usually can be constructed and represented with a group of representative and simpler elementary building parts, e.g., we can represent the building tower or body using a cylinder or box and represent the building roof using a hemisphere or cone or pyramid or prism accordingly.

Moreover, common and correlated patterns for building colors largely exist in many buildings. For example, the concrete color (gray) is commonly used to color the walls, brick color (red) is often used to color the roofs. In a conceptual manner, in order to train our network, the building can be constructed in a similar way as in "LEGO." As such, by composing those elementary building parts, we can construct an architectural illustration dataset that are created from simpler but representative building shapes, and the learning can be conducted based on this. In this way, we can efficiently construct the training dataset, while still maintaining the general representability.

In our implementation, the training datasets are created using 10 simpler but representative building shapes constructed from the simpler elementary building parts and the 10 color schemes that are the upmost representative based on the artist's opinion. We asked the artists to watercolor those elementary building parts (such as boxes) using these representative color schemes. We use a common 3D software (in our implementation, Autodesk 3ds Max) to render the building shapes from 100 directions and produce 1000 line drawings $sdata(X) = \{x_1, x_2, \ldots\} \subset X$ as well as respective 10000 colored images without lighting effects $sdata(Y^C) = \{y_1^C, y_2^C, \ldots\} \subset Y^C$ as the training dataset for *ArchColGAN*. Following the same way, we can also construct the training dataset for rotation animation generation, the image sequences corresponding to sequential 100 viewing directions are generated by continuously rotating the camera.

In the same way, 200 colored images from $sdata(Y^C)$ are selected, and their corresponding colored images with lighting effects with respect to different light positions can be automatically generated. We use 8 light directions and note that the light positions are above the buildings in our implementation. This process is realized using Autodesk 3ds Max via scripting. We adopt the standard point light and Phong model for the lighting model. By doing so, we can create $sdata(Y) = \{y_1, y_2, \ldots\} \subset Y$ as the training set for *ArchShdGAN*. For training *ArchShdGAN*, we use only the essential lighting effect information from $sdata(Y)$, which is actually $sdata(Y_S^C)$ and $sdata(Y_S)$, this formulation is introduced in Sect. 3.3 in detail.

In addition, in order to achieve desired effects, users can choose to adjust the colorization and lighting effects and then retrain the model.

3.2 Color translation

3.2.1 *ArchColGAN*

As shown in Fig. 3, the proposed *ArchColGAN* follows GAN architecture, including one generator network (G,) and two discriminator networks (D). The discriminators are local and global discriminators (LD and GD). The main features are as follows.

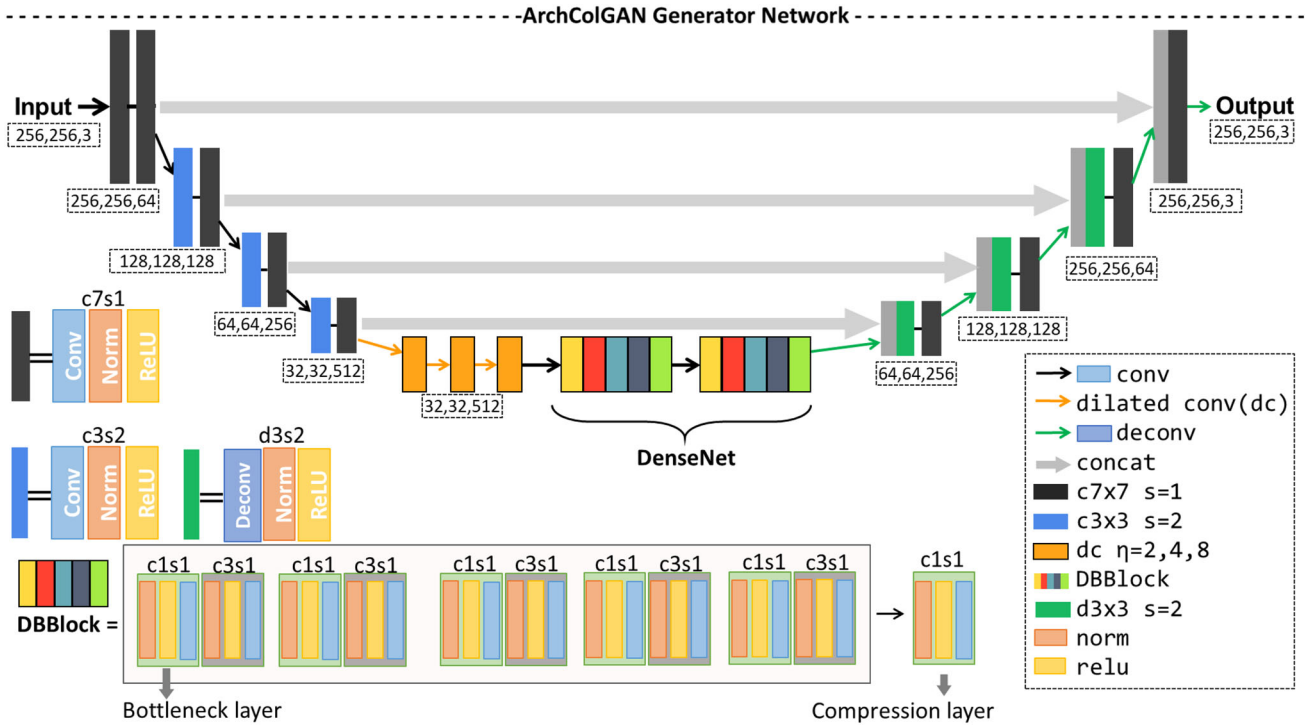


Fig. 3 Generator network of*ArchColGAN* c of $c7 \times 7$ refers to the convolution, d in $d3 \times 3$ refers to the deconvolution, s refers to the size of step, r refers to the dilation factor. These are the same in other Figs.

(1) The generator G is designed to achieve these two tasks: stylized colorization as well as inpainting. For an input line drawing image, we randomly cut a hole in it and apply the G to produce a stylized inpainted and colored output image. In this way, the capability of handling the building feature details, e.g., the corner features, and the capability of learning the building part correlation of the trained model can be strengthened. Conceptually speaking, we use the local feature synthesis operation (inpainting) to model the building part correlation (connecting), at the same time, we can also emphasize and thus better preserve the local features. The hole is set to match the essential feature size (e.g., the corner size). For the purpose of general coverage, its location is normally distributed in the image.

(2) We propose utilizing U-Net in the generator G. This facilitates the capture of important features, for the purpose of tackling the line feature maintaining problem. The feature reuse and upsampling can be also empowered with the concatenation function.

(3) Buildings may possess rich and vivid colors and complex shapes. There are also differences between the testing and training datasets. Furthermore, the training dataset may not have pixel-to-pixel matches and is not always fully paired (e.g., there might be missing corner features). Because of this, the generator model may encounter some difficulties. For the purpose of increasing the adaptiveness with respect to those variations, we design G by integrating cycle consistency (CycleGAN [45]) to the U-Net, in order to make G more robust and versatile while preventing the mode collapse. With $L_{cyc}(G, F)$ as the cycle consistency loss, $G : X \rightarrow Y^C$ is trained with its inverse mapping model $F : Y^C \rightarrow X$.

(4) In G, we propose utilizing a dilated convolution, its expanded receptive field can be helpful to reduce the unwanted colorization unevenness caused by sparse line. Instead of using ResNet [16] as in the conventional methods, we propose employing a DenseNet to be used as the transformer in the U-Net. By doing so, color and line feature generation can be enhanced. Furthermore, the feature reuse is increased and the number of parameters can be reduced.

(5) The discriminator D has global and local discriminators (GD and LD). GD is responsible for the overall output image from G, while LD aims to handle the region in the output image corresponding to the inpainted part. In this way, the global consistency and local features can be both preserved. Furthermore, rather than focusing on only the overall plausible colorization, adding LD can engage G to generate a better local colorization, thus the undesired uneven colorization can be reduced.

3.2.2 Loss

G^*, GD^*, and LD^* denote the weights of network respective. To this end, we want to solve this problem of minimization/maximization: G tends to minimize the objective $L(G, GD, LD)$ against the adversary LD and GD tries to maximize it, as follows:

$$(G^*, GD^*, LD^*) = \arg\min_{G} \max_{GD,LD} L(G, GD, LD),$$
$$L(G, GD, LD) = L_{adv}(G, GD, LD) + \lambda L_{cyc}(G, F).$$

We define the cycle consistency loss as:

$$L_{cyc}(G, F) = E_{x\sim sdata(X)}[\|F(G(x)) - x\|_1] +$$
$$E_{y^C\sim sdata(Y^C)}[\|G(F(y^C)) - y^C\|_1].$$

We define the adversarial loss as:

$$L_{adv}(G, GD, LD) =$$
$$E_{y\sim sdata(Y^C)}\Big[log\big(GD(y^C) + LD(y^C_{patch})\big)\Big] +$$
$$E_{x\sim sdata(X)}\Big[log\Big(1 - GD\big(G(x)\big) - LD\big(G(x_{patch})\big)\Big)\Big].$$

3.2.3 Implementation

The input to G is a line drawing image (256 × 256 pixel resolution) with a hole (40 × 40 pixel resolution). The center position of the hole is normally distributed within the image, with a 5 pixels padding margin to the image boundary.

Please refer to Fig. 3. U-Net in G begins with two Flatten layers. One Flatten layer contains a convolution (Conv) kernel of 7 × 7 with 1 as the step size, an instance normalization function (Norm), and a rectified linear unit (Relu) with a fixed size of the output feature map.

Then, three downsampling convolution blocks (encoding blocks) are followed. Each encoding block has a downsampling (Conv-Norm-Relu) and a flatten layer to compress and encode the image compressing and encoding. The important and useful image features can be abstracted for the later transformer. A 3 × 3 kernel with step size 2 is used in this downsampling, the number of feature channels is doubled after each step.

Afterward, the dilated convolution is applied. Without increasing the learnable weights, this step can help to expand the convolution kernel, thus, it enables to use more areas as the input at each layer. Specifically, for a 2D layer of C channel $h \times w$ mapping and a next layer of C' channel $h' \times w'$, the dilated convolution operator of each pixel is defined as:

$$y_{u,v} = \sigma\left(b + \sum_{i=-c_h^{'}}^{c_h^{'}} \sum_{j=-c_w^{'}}^{c_w^{'}} W_{c_h^{'}+i, c_w^{'}+j} x_{u+\eta i, v+\eta j}\right),$$
$$k_h^{'} = \frac{c_h - 1}{2}, c_w^{'} = \frac{c_w - 1}{2},$$

where c_w and c_h (odd numbers) are the kernel width and height. η is the dilation factor. $x_{u,v} \in R_C$ and $y_{u,v} \in R_{C'}$ refer to the input and output pixel components of the layer. $\sigma(\cdot)$ is a nonlinear component-wise transfer function. $W_{i,j}$ are C'-by-C kernel matrices. $b \in R_{C'}$ is the vector of layer bias. Letting $\eta = 1$ will make the equation becoming the standard convolution. We use $\eta = 2, 4, 8$, in our implementation.

Consequently, we employ the DenseNet, which has two dense network blocks (DBBlock). After one block, a 1 × 1 Conv compression is applied with a 0.5 compression factor. Each block contains 5 layers, each layer has a Norm, a Relu, and a 3 × 3 Conv with a growth rate of 32. Beforehand, a bottleneck layer is applied. It involves a Norm function, a Relu, and a 1 × 1 Conv layer. In order to reduce the amount of input feature maps, thus reducing the computational cost, in our implementation, we let each 1 × 1 Conv generate $4k$ feature maps. In other words, we call the structure having 5 layers of Norm-Relu-Conv(1 × 1)-Norm-Relu-Conv(3 × 3) as the dense network block and use 2 structures of DBBlock-Conv(1 × 1) as the transformer of the generator.

Subsequently, three upsampling convolution blocks (decoding blocks) are applied to reconstruct and output the stylized colorization. Each block has an upsampling layer followed by a flatten layer. The upsampling layer is a deconvolution layer with a kernel of 3 × 3 and a step size of two (Deconv-Norm-Relu). We use the same flatten layer as in the encoding block. The number of feature channels is halved after each upsampling step. Finally, the final Conv layer with 3 × 3 kernel size is applied. Note that the output features of each encoding block and the output features of its corresponding decoding block are concatenated as the input for the next respective decoding block.

Then, the cycle consistency is applied, please refer to Fig. 4. The whole generated image (256 × 256 pixels) is the input for GD. GD has 4 downsampling layers (Conv 4 × 4, step size 2) and one Conv layer (Conv 4 × 4, step size 1). GD outputs a 16 × 16 matrix to compute differences to the real data. LD are similar to GD. For LD, the input is a 60 × 60 patch that is resized from the 40 × 40 patch at the inpainted hole position, and a 2 × 2 matrix is the output.

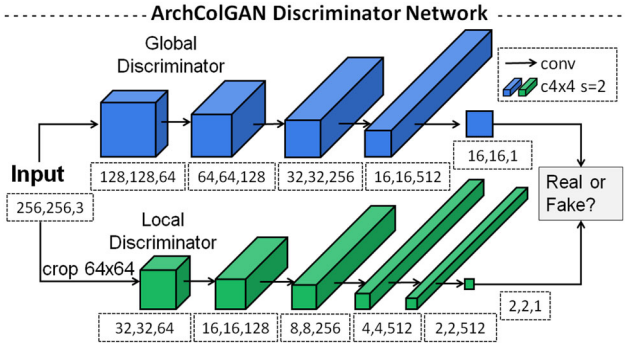


Fig. 4 Discriminator network of *ArchColGAN*

3.3 Lighting effect

3.3.1 *ArchShdGAN*

Our proposed *ArchShdGAN* is also based on the CycleGAN [45] architecture. Different to CycleGAN, which mostly emphasizes on image-to-image translation, the goal of *ArchShdGAN* is to handle effect-to-effect translation for building lighting effects. Our approach is based on the observation that the lighting effect perceived by the viewer can be plausibly depicted and represented using the Value channel of the HSV color space representation, which includes Hue, Saturation, and Value channels. We therefore proposed formulating this effect-to-effect translation for the lighting effect as the Value-to-Value translation. That is, $Y_S \approx Y_V$, where Y_V refers to the Value channel.

The colored building image (y^C) is converted from RGB to HSV format. Its Value attribute is then fetched and used as the input for *ArchShdGAN*, that is, $y_S^C \approx y_V^C$. As shown in Fig. 5, the generator of *ArchShdGAN* includes the following steps. We first apply one Flatten layer and two layers of downsampling convolution encoding. Subsequently, we use ResNet [16] as the transformer. After this, we apply upsampling with two convolution layers and then one Flatten layer, in order to generate the Value attribute. This Value attribute generated from the generator is used as the input to the discriminator, which includes four downsampling convolution layers. Then, we compute and evaluate the loss, as well as incorporate it with the discriminator to realize the Value attribute adversarial generation.

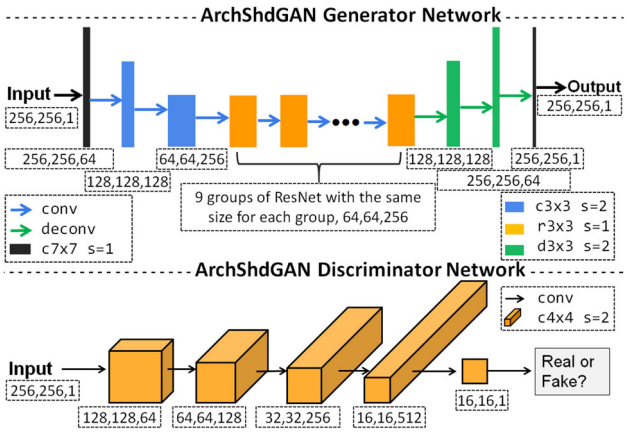


Fig. 5 Generator and discriminator networks of *ArchShdGAN*. Note that *r* means ResNet [16] here

3.3.2 Loss

We denote the weights of generator and discriminator of *ArchShdGAN* as G_S^* and D_S^*, respectively. The loss function is denoted as L_S. Similarly, we would like to solve the minimization/maximization problem of G_S trying to minimize the objective $L_S(G_S, D_S)$ against an adversary D_S that tries to maximize it, as follows:

$$(G_S^*, D_S^*) = \arg\min_{G_S} \max_{D_S} L_S(G_S, D_S)$$
$$L_S(G_S, D_S) = L_{S_{adv}}(G_S, D_S) + L_{S_{cyc}}(G_S, F_S).$$

We define the adversarial loss $L_{S_{adv}}$ as:

$$L_{S_{adv}}(G_S, D_S) = E_{y_S \sim sdata(Y_S)}[log(D_S(y_S))] + E_{y_S^C \sim sdata(Y_S^C)}[log(1 - D_S(G_S(y_S^C)))].$$

We define the bidirectional cycle consistency loss $L_{S_{cyc}}$ as:

$$L_{S_{cyc}}(G_S, F_S) = E_{y_S^C \sim sdata(Y_S^C)}[\|F_S(G_S(y_S^C)) - y_S^C\|_1] + E_{y_S \sim sdata(Y_S)}[\|G_S(F_S(y_S)) - y_S\|_1].$$

3.3.3 Concatenation

The final step is concatenating y_S output with the y^C Hue and Saturation attributes and then converting it back into the RGB format, which is the final result y with the stylized colorization and lighting effect (Fig. 6).

4 Colorization enhancement

On top of *ArchColGAN*, in order to further enhance the colorization results, we propose incorporating the attention idea to reduce the unwanted blank areas and better preserve the features. This step acts as one complementary operation, if the user is satisfied with the *ArchColGAN* result, this step can be skipped. The basic idea and realization of adding attention are as follows:

(1) In the architecture illustration, the build itself attracts most of the viewer's attention, as such we focus on syn-

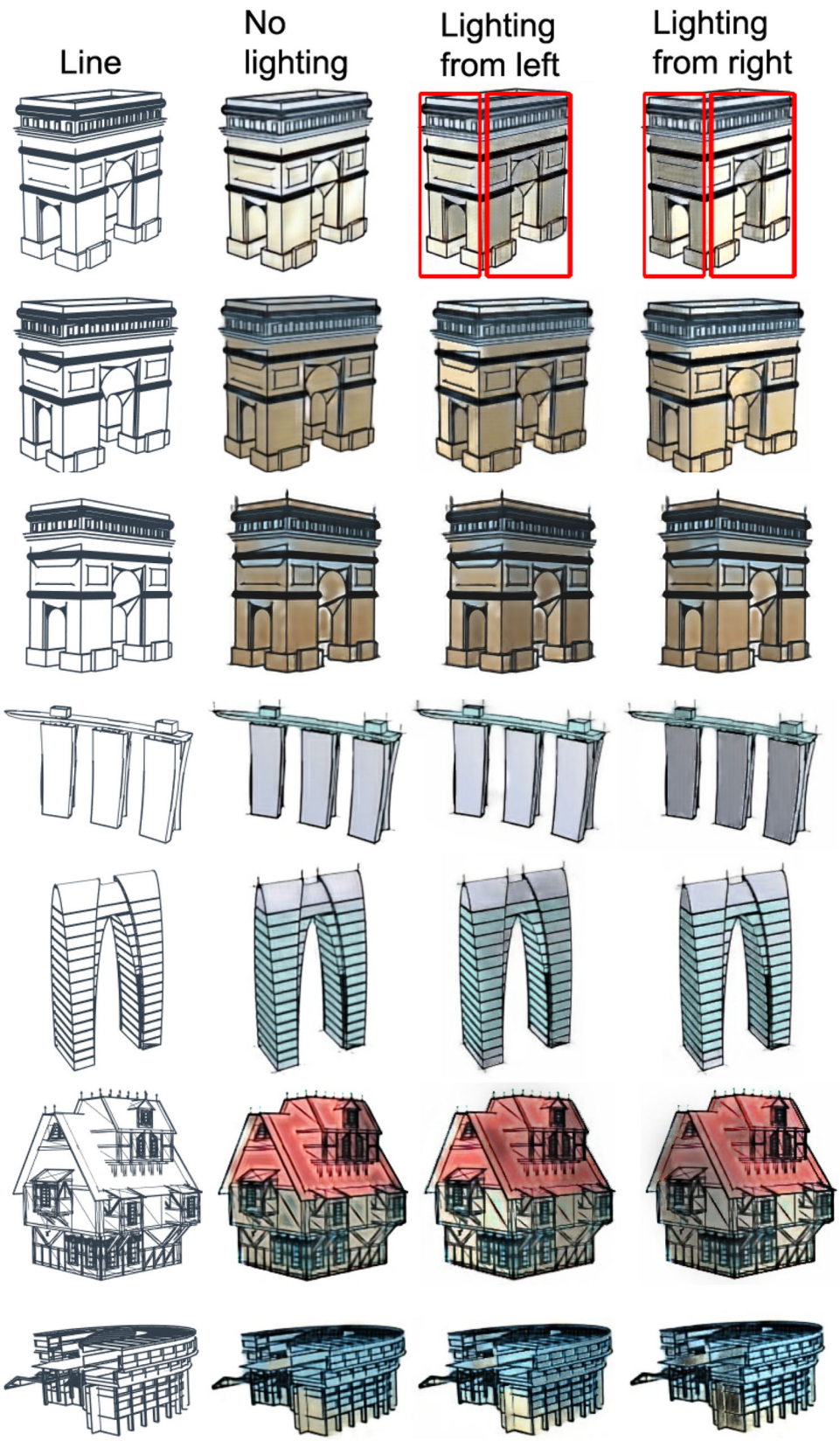


Fig. 6 Adding lighting effects. Note that the illumination changes are highlighted in the first row as an example

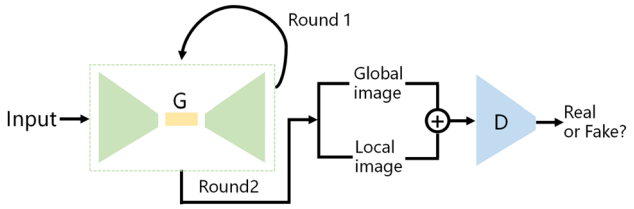


Fig. 7 Structure for the colorization enhancement

thesizing and maintaining the information of the building. So, instead of applying a simple random cut within the whole image region as in *ArchColGAN* (1), we restrict the cut in the building region of the image.

(2) We pay more attention to the unwanted blank areas to reduce them, after the G of *ArchColGAN* we add an additional round of generation using the same G (Fig. 7). Analogously, this is similar to the real-world coloring, that is, performing a secondary makeup after the main coloring. This second round G is conducted based on the result of the first round G. Moreover, in the generated local cut region from the first round G, we first apply a high-pass filter to exaggerate the blank area, to attract the networks' attention to inpaint this. Secondly, if this cut contains a line segment, we add it back to the cut, as such during the following G, the line feature can be more emphasized and better preserved with coherency with surrounding coloring.

(3) Furthermore, on top of *ArchColGAN*, we follow the attention map and define a mask with 0s at the blank areas the 1s at the rest. We concatenate and input them to the downsampling convolution layer. In the later steps of the network, the image and mask will be convoluted together, respectively. Using the mask, we can determine the local image (non-1s). In this case, the local image can have a soft boundary, which means a better connection with the global image to improve the global coherency. By applying the masking, we can derive LD based on GD, we therefore can also combine their computation to reduce the network complexity.

All the other parts are the same as *ArchColGAN*.

5 Rotation animation effect

We consider the rotation animation effect as a sequence of images generated by rotating the building. Recurrent neural networks such as LSTM [18] have been proved to achieve state-of-the-art results on dealing with such time series data while preserving the frame coherency. As such, we adopt the LSTM into our *ArchColGAN* framework to realize this effect.

Given a line drawing image sequence, in order to improve the frame coherency, we design our pipeline as follows.

Similar to the colorization enhancement, the main idea is that in an additional generation round, we utilize the prediction power of the LSTM to incorporate information inherited (or memorized) from the previous frames to enhance the current frame in terms of the coherency with its previous frames. It bears the same concept as the cut inpainting in our previous approaches. Here, it can be thought of as inpainting in time-space (predicting one frame).

In the first round, we use G from *ArchColGAN* to generate the colored images for the current frame and its previous 3 consecutive images.

In the second round, we input these 4 images with cut holes as in the colorization enhancement, into the generator again. We follow the same generator network structure of *ArchColGAN* as shown in Fig. 8. We adjust this the second round G to have additional downsampling and upsampling layers, so the feature map becomes $16 \times 16 \times 1024$ after the downsampling (encoding). We also replace the transformer DenseNet with the LSTM net. The previous 3 colored frames are encoded and only used in this LSTM to generate (or predict) the feature map of the current frame. On the other hand, the colored current frame is encoded and used for concatenation in decoding this feature map generated using LSTM. Explaining this in detail, the 3 encoded $16 \times 16 \times 1024$ feature maps of the 3 previous frames are transformed to 256×1 vectors to input into LSTM.

In our implementation, we engage the standard LSTM by calling the *BasicLSTMCell* method from Tensorflow [1] with 256 as the *num_units*, and using it as the *cell* of the *dynamic_rnn* method. Its output is convoluted to $16 \times 16 \times 1024$ and used as the feature map of the current frame. As mentioned, it will then go through the decoding which is based on concatenating the encoded current frame. All the other parts are the same as *ArchColGAN*.

6 Result and discussion

We implement our method using the TensorFlow [1] framework. We use a Desktop PC with Intel Core i7-7700K CPU and two Nvidia GeForce GTX1070 GPUs for the implementation and experiments. The training rate of the method is $2e^{-4}$, and the number of epochs is 1000. Using this system setup, the model training times are as follows: *ArchColGANs*: 8 hours, *ArchShdGANs*: 3 hours, color enhancement: 8 hours, animation enhancement: 8 hours. The model running time is 2 to 5 seconds for generating one 256×256 image/frame for all the models.

We applied *ArchGANs* on a number of representative building line drawings, using some user-defined color schemes. As shown in Figs. 6, 9, 10, and in the supple-

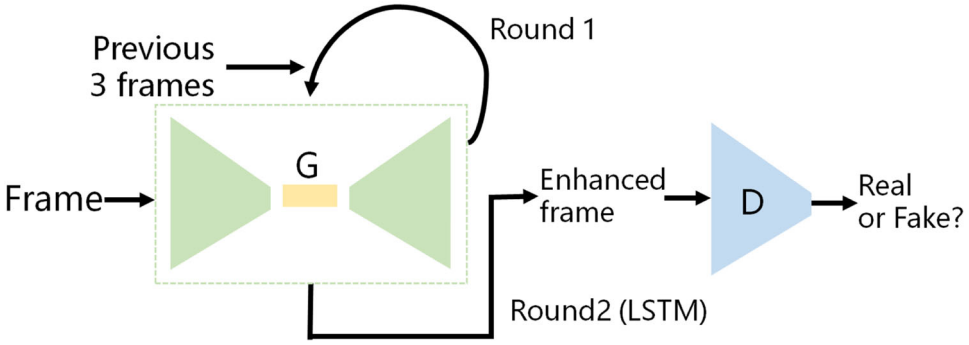


Fig. 8 Structure for the animation enhancement

mentary material of more *ArchGANs* results, *ArchGANs* can generate stylized colorization that are in general visually plausible as well as can maintain line features, reduce unwanted uneven colorization, and augment the colorization with lighting effects. Note that, for the examples in these figures, *ArchColGAN* with and without color enhancement has the same performance. However, there are few cases that *ArchColGAN* may not perform well. In these rare cases, our newly proposed color enhancement can help to improve the results as shown in Sect. 6.2.

6.1 Evaluating *ArchGANs*

As shown in Fig. 9, to evaluate our proposed method, we compare our results using *ArchGANs* with the results generated using the representative state-of-the-art GAN-based methods including VGG [15], pix2pix [21], DualGAN [41], and CycleGAN [45]. We also compare our results using *ArchGANs* with the ground truth colorization which is produced by a professional artist. We use the same color scheme as the ground truth.

Particularly, we evaluate *ArchGANs* from 3 important aspects: lines, colors, and lighting effects with a subjective user study. We evaluate the lines from three perspectives: maintaining the line structure, stylizing, and repairing the lines. We evaluate the colors from three perspectives: evenly coloring of large walls, color consistency of windows, and distinction between the main building and the background.

6.1.1 Line evaluation

As shown in the column two of Fig.9, the VGG network in general cannot preserve well the line structure in their results. Mainly, this is due to that its performance largely relied on the comprehensiveness of the training dataset.

But in most circumstances, we have to face the situation that there can be relatively large differences between the training and test datasets. Hence, using VGG would result in results that are often unexpected. VGG usually performs better for the task of transferring colors and textures, but it does not perform very well for preserving the structure of lines. Moreover, it usually tends to preserve high-level general information and thus can be more suitable for artistic stylized effects. Unlike VGG, other methods can generate satisfying outputs in terms of maintaining the structure of lines.

In the first row of Fig. 9, the triumphal arch model is used to demonstrate the stylized line drawing transformation. The input line drawings are expected to be transformed into stylized line drawings with crossing features at the corners. Such corner crossings are frequently used for perspective references and can be commonly found in many architectural illustration images. In the results of CycleGAN and our method (as shown in the figure), the corner crossings are much clearer. This indicates that both two methods can achieve the better-stylized transformation of the lines. However, in the results of other methods, the corner crossing features can be hardly seen (less stylized line drawings). This is because CycleGAN and our method in general perform better when dealing with such local features.

6.1.2 Color evaluation

The outputs at the first, second, and third rows of Fig. 9 show the capability for dealing with colorization for large-area walls.

For the larger wall area of building, e.g., in the first row, for the pillar faces of the triumphal arch, and in the second row, for the external walls of the house, there are larger blank areas in the center of the walls in the colorization results of pix2pix, DualGAN, and CycleGAN methods. And, there is an unwanted red color for the wall areas (third row) in the results of pix2pix and CycleGAN. This kind of colorization unevenness is unnatural, and the overall perception experiences may be degraded. Different from them, our method can produce more even colorization results for large-area walls and reduce the unwanted blank area.

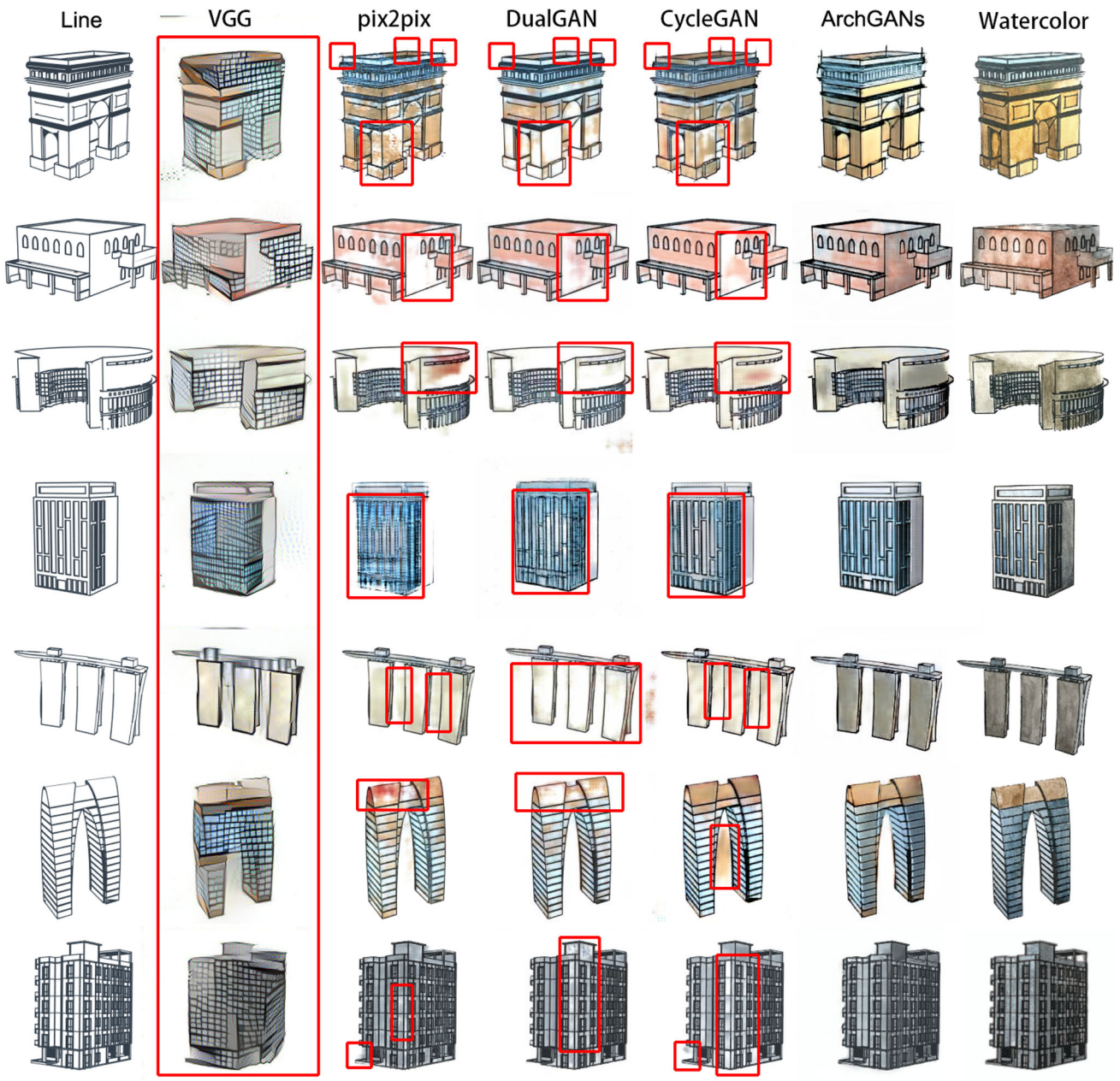


Fig. 9 Comparing *ArchGANs* with different methods. From left to right: the line drawing input images, VGG, pix2pix, DualGAN, CycleGAN, *ArchGANs*, and watercoloring by an artist. Note that some artifacts and features are highlighted

Table 1 Mean opinion scores of the user feedback for evaluating *ArchGANs* and the animation extension of *ArchColGANs*

VGG	pix2pix	DualGAN	CycleGAN	ArchColGANs Animation
1.214	2.170	2.523	2.995	3.621
ArchGANs without LD	**ArchGANs without DC**	**ArchGANs without Lighting**	**ArchGANs**	**ArchColGANs Animation with enhancement**
2.840	3.254	3.869	4.261	4.062

In the fourth and sixth rows of Fig. 9, we can observe the capability to deal with window color consistency. Higher color consistency (blue) can be observed in our results, comparing with the others.

As shown in the fifth and sixth rows of Fig. 9, in the results of other methods, the empty region between building pillars has leaked color, this is in general unwanted for colorization.

6.1.3 Lighting effect evaluation

As shown in Fig. 6, in our results, the direction of the light source and the 3D effect of the building can be plausibly represented. Our *ArchShdGAN* module is helpful in generating such lighting effects.

6.1.4 Ablation study

We also conducted an ablation study, in order to further evaluate the effectiveness of *ArchGANs*. The results in the following cases are compared in this ablation study: *ArchGANs* without LD, *ArchGANs* without dilated convolution (DC), *ArchGANs* without Shading, and *ArchGANs*, using different color schemes (Fig. 10a) and building models (Fig. 10b).

The following benefits of *ArchGANs* modules can be observed based on this study: our model with GD and LD can help to deal with local feature details, e.g., the features in the bottom of the building in Fig. 10b row one, adding the dilated convolution can be helpful in reducing the uneven colorization in the large walls, and adding the lighting effects can enhance the building depiction in 3D.

6.1.5 User study

In order to evaluate our method, we conducted a user study based on the mean opinion score (MOS). We invited 17 participants with art backgrounds to give their opinion scores regarding various colorization results. They are invited to evaluate based on the perceptual experience and visual quality. For reference, we also provide 20 ground truth colorization by artists to the participants. The opinion scores are ranging from 1 to 5: 1 (Very bad), 2 (Bad), 3 (Average), 4 (Good), 5 (Very good). The results are generated using 8 methods: VGG, pix2pix, DualGAN, CycleGAN, *ArchGANs* without LD, *ArchGANs* without dilated convolution (DC), *ArchGANs* without Lighting, and *ArchGANs*. We generated 62 different colorization results for each case. Each participant evaluated 496 images, and we have 8432 scores in total. Table 1 left shows the MOS of the results using those methods.

From the user study, we learned that the state-of-the-art CycleGAN (MOS 2.995) performs better than *ArchGANs* without LD (MOS 2.840), but, after incorporating LD, dilated convolution, and Shading, the results can be greatly improved (MOS 4.261), and the viewer can have a better visual experience. In terms of the components, compared to adding Shading effect (MOS from 3.869 to 4.261), adding dilated convolution (MOS from 3.254 to 4.261) and LD (MOS from 2.840 to 4.261) are relatively more important components for improving results. This shows that compared with the lighting effect, users usually are more sensitive to the color effect, e.g., the evenness of the colorization.

6.2 Evaluating color and animation enhancement

We evaluate the newly extended enhancement methods by comparing the enhancement results with the results using only *ArchColGANs*.

6.2.1 Color enhancement

In our experiments, we found that in most cases, *ArchColGAN* can already produce satisfactory results. In some rare cases when the color enhancement is needed, as shown in Fig. 11, our new complementary color enhancement method can improve the results of *ArchColGAN* by reducing the unwanted blank areas and maintaining the line features. Adding an additional generation round can be helpful, however, it requires some additional computational costs.

6.2.2 Animation enhancement

Please refer to the supplementary video, we compare the proposed animation enhancement by solely applying *ArchColGANs* at each image. Some frames of the video are shown in Fig. 12. It can be seen that the additional animation enhancement can improve the frame color consistency. The ambiguities and randomness in the frame-by-frame colorization can cause and magnify the color inconsistency in animation. However, in our proposed animation enhancement, considering the previous frames can be helpful in reducing such color inconsistency, thus a better animation effect can be produced.

Additionally, we also present a straightforward zooming effect using our method. The zooming starts from 256×256 resolution and zooms out (smaller) then in (bigger) until 256×256. This is done using a post-processing scaling. We currently focus on 256×256 images, we will discuss this in the limitation section.

As a qualitative evaluation, we also conduct a user study, as shown in the right of Table 1. Similar to the previous one, we ask for the user opinions regarding additional 9 animation clips generated using *ArchColGAN* with and without animation enhancement. From the evaluation results, we get to learn that adding the animation enhancement can improve the MOS from 3.621 to 4.062. This indicates the effectiveness and usefulness of the proposed animation enhancement.

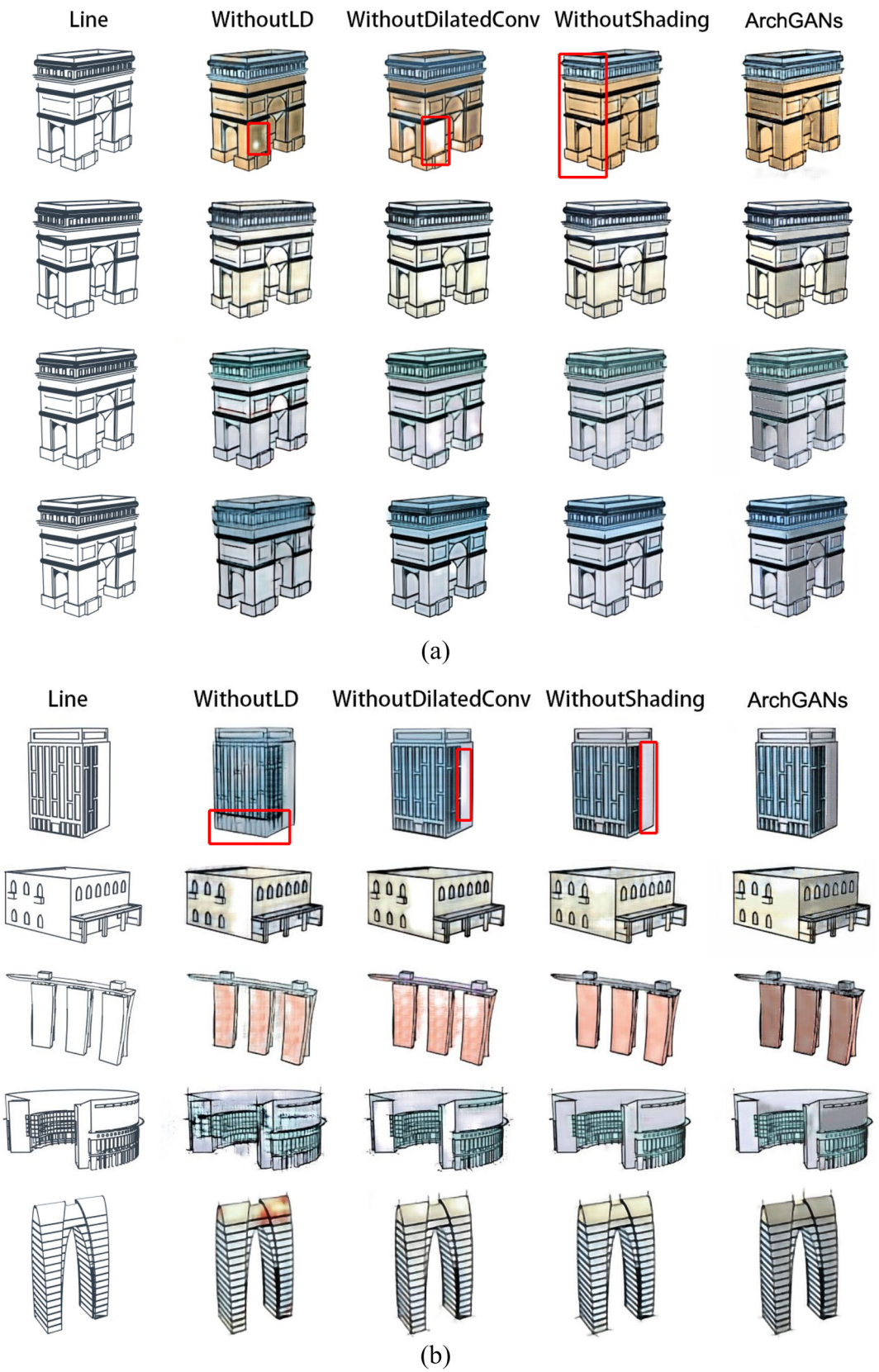


Fig. 10 Ablation study for ArchGANs. (a) Different color schemes. (b) Different building models. Note that the main differences are highlighted in the first row as an example

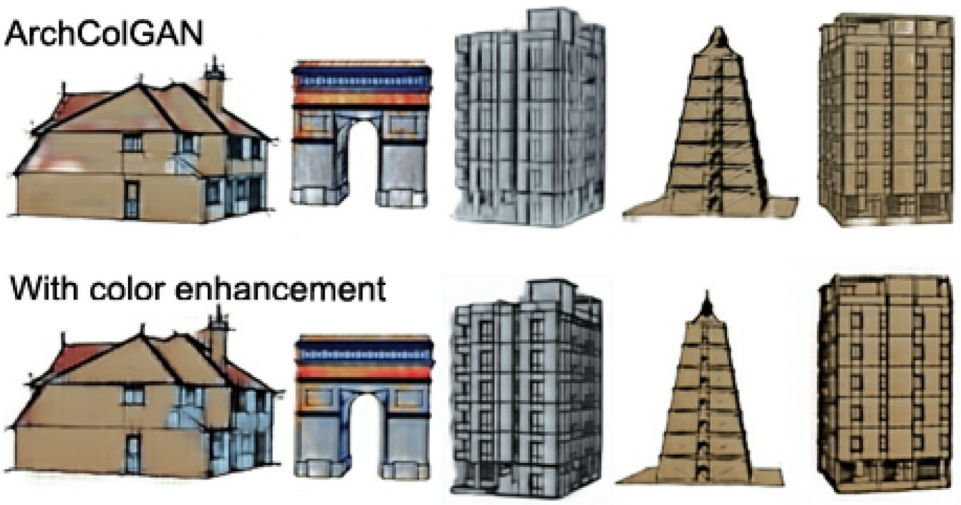


Fig. 11 Color enhancement for *ArchColGAN*

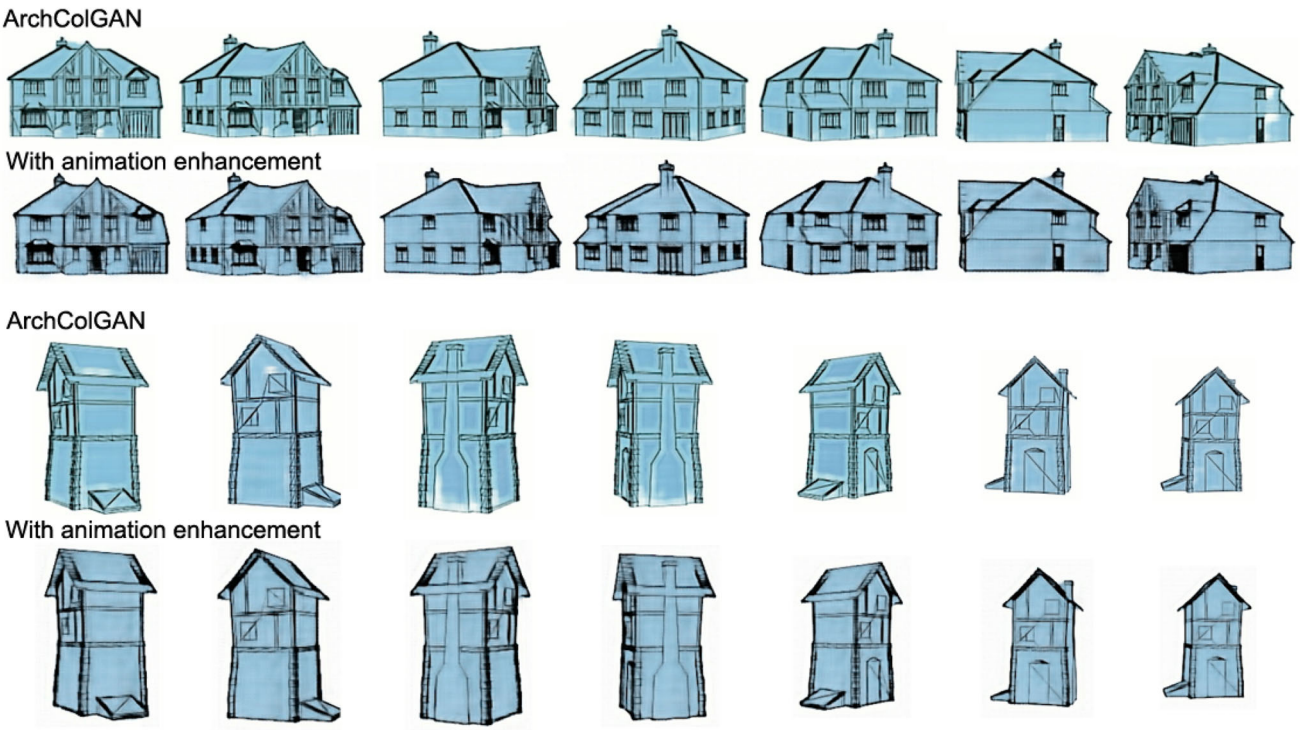


Fig. 12 Animation enhancement for *ArchColGAN*

6.3 Limitations

The main limitations are as follows:

(1) The construction of the training dataset involves some manual and human efforts to achieve visually plausible results, although we have attempted to reduce such efforts, such as proposing the "LEGO" manner approach. Currently, we invite expert opinions in designing the shape and color schemes and ensuring the quality of the training dataset, which is considered by us as the quality of the ground truth. However, this can be prone to human errors and subjective. As a result, the richness and diversity of the results can be affected and limited by the dataset generation. In the future, we plan to explore fully automatic data synthesis and data augmentation methods to generate a more comprehensive dataset and further reduce the manual efforts, such as using texture synthesis methods. We also plan to apply image quality assessment methods to quantitatively access the dataset quality.

(2) The training is computationally expensive. Due to our current computational power constraint, our model is trained and tested with a low resolution of 256 × 256. We plan to investigate methods to expedite the training process and consider progressive growing [24] to progressively add details to our current 256 × 256 image to generate a high resolution image.
(3) We currently focus on only the lighting effects, more effects like shadow and reflection can be added by considering the idea of screen space ambient occlusion and the material property of the building parts.
(4) Our animation enhancement works mainly on the gradual changes (continuous frames), not the large changes. To handle the large changes, the animation techniques such as inbetweening can be considered.

7 Conclusion and future work

In this research, a novel GAN-approach, *ArchGANs* has been proposed for effective prototyping stylized architectural line drawing colorization. It consists of two main parts, *ArchColGAN* and *ArchShdGAN*.

ArchColGAN is designed to conduct both stylized colorization and inpainting tasks. We realize the stylized colorization by utilizing U-Net and incorporating two-stage discriminators (local and global), dilated convolution, and cycle consistency. *ArchShdGAN* can add lighting effects. Different from the existing methods, *ArchGANs* has better support for lighting effects, even colorization, and handling line features. Furthermore, we also proposed an extension for complementary color enhancement and adding rotation animation effect in *ArchColGAN*. The effectiveness of *ArchGANs* has been demonstrated in our results and evaluation.

As future work, besides those mentioned in the limitation section, we also want to apply our method to handle other scenarios, e.g., industrial, car, and CAD design. Another possible future work is to extend *ArchGANs* to deal with other objects, e.g., sky, streets, and vegetation.

Acknowledgements We gratefully thank the reviewers for their constructive comments.

Funding Open Access funding enabled and organized by Projekt DEAL. This research is supported by NSFC Grants (61702363, 51978441), China and the National Research Foundation, Singapore under its International Research Centres in Singapore Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

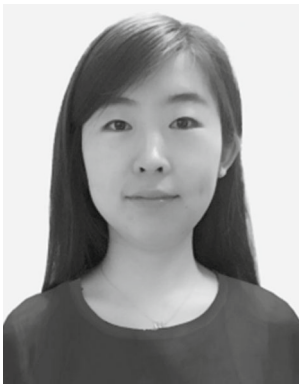
Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, DG., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zheng, X.:TensorFlow: A system for large-scale machine learning. In: OSDI'16: Proceedings of the 12th USENIX symposium on operating systems design and implementation, pp 265–283(2016)
2. Bousseau, A., Kaplan, M., Thollot, J., Sillion, FX.: Interactive watercolor rendering with temporal coherence and abstraction. In: NPAR'06: Proceedings of the 2006 international symposium on non-photorealistic animation and rendering, pp 141–149(2006)
3. Byeon W, Wang Q, Kumar Srivastava R, Koumoutsakos P (2018) ContextVP: Fully context-aware video prediction. In: ECCV'18: Proceedings of the European conference on computer vision, pp 753–769
4. Cao, K., Liao, J., Yuan, L.: CariGANs: unpaired photo-to-caricature translation. ACM Trans. Graph. **37**(6), 1–14 (2018)
5. Capcom (2008) Street fighter iv
6. Chen, D., Liao, J., Yuan, L., Yu, N., Hua, G.: Coherent online video style transfer. In: ICCV'19: Proceedings of the IEEE international conference on computer vision, pp 1105–1114 (2017a)
7. Chen, D., Yuan, L., Liao, J., Yu, N., Hua, G.: Stylebank: An explicit representation for neural image style transfer. In: CVPR'17: Proceedings of the 2017 IEEE conference on computer vision and pattern recognition, pp 1897–1906 (2017b)
8. Chu, N.S.H., Tai, C.L.: MoXi: real-time ink dispersion in absorbent paper. ACM Trans. Graph. **24**(3), 504–511 (2005)
9. Clark, A., Donahue, J., Simonyan, K.: Adversarial video generation on complex datasets. (2019).arXiv preprint p arXiv:1907.06571
10. Corel (2011) Painter 12. www.corel.com
11. Curtis, CJ., Anderson, SE., Seims, JE., Fleischer, KW., Salesin, DH.: Computer-generated watercolor. In: SIGGRAPH'97: proceedings of the 1997 annual conference on computer graphics and interactive techniques, pp 421–430(1997)
12. DeCarlo, D., Finkelstein, A., Rusinkiewicz, S., Santella, A.: Suggestive contours for conveying shape. ACM Trans. Graph. **22**(3), 848–855 (2003)
13. Fang, L., Wang, J., Lu, G., Zhang, D., Fu, J.: Hand-drawn grayscale image colorful colorization based on natural image. The Vis. Comput. **35**(3), 1667–1681 (2013)
14. Gatys, LA., Ecker, AS., Bethge, M.:Image style transfer using convolutional neural networks. In: CVPR'16: proceedings of the 2016 IEEE conference on computer vision and pattern recognition, pp 2414–2423 (2016)

15. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.:Generative adversarial networks. In: NIPS'14: Proceedings of the 2014 international conference on neural information processing systems, pp 2672–2680 (2014)
16. He, K., Zhang, X., Ren, S., Sun, J.:Deep residual learning for image recognition. In: CVPR'16: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778 (2016)
17. Hertzmann, A.: Tutorial: a survey of stroke-based rendering. IEEE Comput. Graph. Appl. **23**(4), 70–81 (2003)
18. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
19. Huang, H., Fu, T.N., Li, C.F.: Painterly rendering with content-dependent natural paint strokes. The Vis. Comput. **27**(9), 861–871 (2011)
20. Huang, X., Liu, MY., Belongie, S., Kautz, J.:Multimodal unsupervised image-to-image translation. In: ECCV'18: Proceedings of the 2018 european conference on computer vision, pp 172–189 (2018)
21. Isola, P., Zhu, JY., Zhou, T., Efros, AA.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1125–1134(2017)
22. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV'16: Proceedings of the European conference on computer vision, pp 694–711(2016)
23. Judd, T., Durand, F., Adelson, EH.:Apparent ridges for line drawing. ACM Transactions on Graphics 26(3):19–es (2007)
24. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of GANs for improved quality, stability, and variation.(2017) arXiv preprint p arXiv:1710.10196
25. Kim, H., Jhoo, HY., Park, E., Yoo, S.:Tag2Pix: Line art colorization using text tag with secat and changing loss. In: ICCV'19: Proceedings of the ieee international conference on computer vision, pp 9056–9065 (2019)
26. Kim, T., Cha, M., Kim, H., Lee, JK., Kim, J.: Learning to discover cross-domain relations with generative adversarial networks. In: ICML'17: Proceedings of the 2017 international conference on machine learning, pp 1857–1865(2017)
27. Kolomenkin, M., Shimshoni, I., Tal, A.: On edge detection on surfaces. In: CVPR'09: Proceedings of the 2009 IEEE conference on computer vision and pattern recognition, pp 2767–2774(2009)
28. Lei, SIE., Chang, CF.:Real-time rendering of watercolor effects for virtual environments. In: PCM'04: Proceedings of the 2004 Pacific Rim conference on Advances in multimedia information processing, pp 474–481 (2004)
29. Liao, J., Yao, Y., Yuan, L., Hua, G., Kang, SB.: Visual attribute transfer through deep image analogy. (2017). arXiv preprint p arXiv:1705.01088
30. Liu, MY., Breuel, T., Kautz, J.:Unsupervised image-to-image translation networks. In: NIPS'17: Proceedings of the 2017 international conference on neural information processing systems, pp 700–708 (2017)
31. Luft, T., Deussen, O.:Real-time watercolor illustrations of plants using a blurred depth test. In: NPAR'06: Proceedings of the 2006 international symposium on non-photorealistic animation and rendering, pp 11–20 (2006)
32. Luft, T., Kobs, F., Zinser, W., Deussen, O.: Watercolor illustrations of cad data. In: Computational Aesthetics'08: Proceedings of the 2008 Eurographics conference on computational aesthetics in graphics, visualization and imaging, pp 57–63(2008)
33. Ohtake, Y., Belyaev, A., Seidel, H.P.: Ridge-valley lines on meshes via implicit surface fitting. ACM Trans. Graph. **23**(3), 609–612 (2004)
34. Okaichi, N., Johan, H., Imagire, T., Nishita, T.: A virtual painting knife. The Vis. Comput. **24**(7), 753–763 (2008)
35. Schaller, T.W.: The art of architectural drawing: imagination and technique. Wiley (1997)
36. Shahroudy, A., Ng, T.T., Gong, Y., Wang, G.: Deep multimodal feature analysis for action recognition in rgb+d videos. IEEE Trans. Pattern Anal. Mach. Intell. **40**(5), 1045–1058 (2017)
37. Simonyan, K., Zisserman, A.:Very deep convolutional networks for large-scale image recognition. (2014) .arXiv preprint p arXiv:1409.1556
38. Tao, W., Jiang, H., Sun, Q., Zhang, M., Chen, K., Erdt, M.:ArchGANs: stylized colorization prototyping for architectural line drawing. In: CW'20: Proceedings of the 2020 international conference on cyberworlds, pp 33–40 (2020)
39. Tulyakov, S., Liu, MY., Yang, X., Kautz, J.:MoCoGAN: Decomposing motion and content for video generation. In: CVPR'18: Proceedings of the 2018 IEEE conference on computer vision and pattern recognition, pp 1526–1535 (2018)
40. Van Laerhoven, T., Van Reeth, F.: Real-time simulation of watery paint: natural phenomena and special effects. Comput. Animat. Virtual Worlds **16**(3–4), 429–439 (2005)
41. Yi, Z., Zhang, H., Tan, P., Gong, M.:DualGAN: Unsupervised dual learning for image-to-image translation. In: ICCV'17: Proceedings of the 2017 IEEE conference on computer vision and pattern recognition, pp 2849–2857 (2017)
42. Zang, Y., Huang, H., Li, C.F.: Artistic preprocessing for painterly rendering and image stylization. The Vis. Comput. **30**(9), 969–979 (2013)
43. Zhang, L., He, Y., Seah, H.S.: Real-time computation of photic extremum lines (PELs). The Vis. Comput. **26**(6–8), 399–407 (2010)
44. Zhang, L., Sun, Q., He, Y.:Splatting Lines: An efficient method for illustrating 3d surfaces and volumes. In: I3D'14: Proceedings of the 2014 ACM SIGGRAPH symposium on interactive 3D graphics and games, pp 135–142 (2014)
45. Zhu, JY., Park, T., Isola, P., Efros, AA.:Unpaired image-to-image translation using cycle-consistent adversarial networks. In: ICCV'17: Proceedings of the 2017 IEEE conference on computer vision and pattern recognition, pp 2223–2232 (2017a)
46. Zhu, JY., Zhang, R., Pathak, D., Darrell, T., Efros, AA., Wang, O., Shechtman, E.:Toward multimodal image-to-image translation. In: NIPS'17: Proceedings of the 2017 international conference on neural information processing systems, pp 465–476 (2017b)

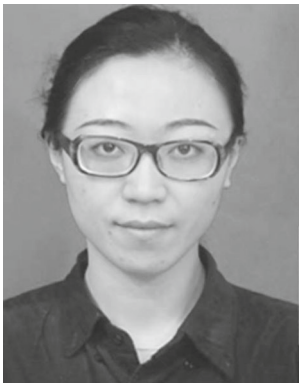


Qian Sun received the Ph.D. degree in Computer Science from Nanyang Technological University , Singapore. She is currently an Asso ciate Professor in the College of Intelligence and Computing, Tianjin University, China. Her current research interests include human-computer interaction and computer graphics.

Yan Chen is a graduate student from Tianjin University. She received an M.Eng. in Software Engineering from Tianjin University. Her research interests include computer vision and computer graphics.

Wenyuan Tao received the B.S. degree in Information Science and Technology from the Department of Information and Control, Xi'an Jiaotong University, China, in 1992 and the M.S. degree in Engineering in Control Theory and Application from Beijing University of Technology, Beijing, China, in 1998. He received the Ph.D. degree in School of Management, Tianjin University, in 2002. In 2015, he was the research fellow at the Drexel University, USA. He is currently the Full Professor, Vice Dean of College of Intelligence and Computing, and Dean of School of Computer Software in Tianjin University, China. His research interest includes virtual reality, technology of digital media content, and Internet of things.

Han Jiang is a software development engineer at JD Logistics. She received a master's degree from the College of Intelligence and Computing at Tianjin University. Her research interests include deep learning, computer graphics, and software development.



Mu Zhang received a Ph.D. degree in Urban & Rural Planning from Tianjin University. Her research interests include cinema tic architecture and animation design.

Kan Chen is a research fellow at Fraunhofer Singapore. He received a B.Comp. (Honors) in Computer Science from National University of Singapore and an M.Eng. and a Ph.D. in Computer Engineering from Nanyang Technological University. His research interests include computer graphics, computer vision, and human-computer interaction.

Marius Erdt is Deputy Director of Fraunhofer Singapore where he is also Head of AI Image Analysis and Data Visualisation. He is also an Adjunct Assistant Professor at the School of Computer Science and Engineering at Nanyang Technological University in Singapore. His research interests are in visual and medical computing, image analysis, and artificial intelligence.