ArchGANs: stylized colorization prototyping for architectural line drawing

Wenyuan Tao[1], Han Jiang[1], *Qian Sun[1], Mu Zhang[1], Kan Chen[2], Marius Erdt[2,3]
[1]*College of Intelligence and Computing, Tianjin University, Tianjin, China*
[2]*Fraunhofer Singapore, Singapore* [3]*Nanyang Technological University, Singapore*
*qian.sun@tju.edu.cn

Abstract

Architectural illustration using line drawing with colorization is an important tool and art format. In this paper, in order to generate a natural-looking and high quality watercolor-like colorization for architectural line drawing, we propose a novel Generative Adversarial Network (GAN) approach, namely *ArchGANs*. The proposed *ArchGANs* unifies a line-feature-aware stylized colorization network (*ArchColGAN*), which can learn, predict and generate the coloring based on a dataset, as well as a shading generation network (*ArchShdGAN*), which augments the illustration with controllable lighting effects for better depicting building in 3D. Specifically, *ArchColGAN* can preserve the essential line features and building part correlation property, it also tackles the uneven colorization problem caused by the sparse lines. Experimental results demonstrate our proposed method is effective and suitable for colorization prototyping.

Keywords: colorization; line drawing; GAN; architectural illustration

1. Introduction

"The architectural drawing is the most eloquent tool a professional has to communicate design ideas. – Paul Rudolph"[1]. Architectural illustration is an important medium to document, communicate or clarify designs for architects. Line drawing with colorization (as shown in Fig.1) is an essential and popular architectural illustration technique. Not only, it is illustrative and able to present the important information of buildings from the architect's engineering prospect, e.g., structure, layout, material, color and lighting effects. But also, from the art prospect, architectural illustration using line drawing with colorization is an important art format to express an architect's ideas and portray artistic architectural concepts. Application-wise, this is also helpful in demonstrating the essence of a particular building design for many purposes e.g., design proposal, competition, marketing and urban planning.

Colorization prototyping is often required in order to test different color schemes and exchange ideas among designers, especially during the early architectural design phase. For example, the colorization is usually achieved based on a line drawing, and watercolor is a popular colorization tool. In this paper, we focus on realizing the watercolor-like colorization prototyping. The conventional manual or semi-manual colorization approaches[2] usually require many time-consuming efforts or high art skills. Computer graphics techniques such as Non-photorealistic rendering (NPR)[3] and physics-based simulation[4] usually require to craft a particular method or to spend a high computational effort to achieve a desired effect. Existing learning-based colorization approaches, based on given examples or style-transfer[5], are mainly focus on transferring general visual attributes (e.g. color and texture), thus they are more suitable to capture the overall visual appearances and similarities. However, it is not easy for them to well preserve the essential underlying line features, which convey the basic and key building structure and layout information for architectural illustration as well as exhibit representative drawing style. For example, in Fig.1, the line features are important as they describe the basic shapes and essential building components, and note that the corners usually have line crossings, which is a typical style for the coordinate reference in a perspective drawing.

Besides, these existing approaches are mainly focusing on 2D manga-character-like input lines[6] and outputs. However, different to these, architectural line drawings usually contain relatively sparse lines and large empty regions, i.e. for the purposes to represent walls, roofs and etc.. Such limited line information can introduce ambiguity and fragmentation when coloring large empty regions. This often causes the colorization results uneven with large blank area and unnatural in the existing methods. Furthermore, unlike other types of art (e.g. portrait painting), architectural illustration usually exhibits stronger correlation property between parts, e.g. a row of windows usually share the same color. However, existing methods do not directly consider this property.

Moreover, these 2D approaches in general lacks the support of 3D illustration, however for architectural illustration, adding lighting effects, which helps to depict a 3D building with different colors under different lighting conditions, can be beneficial to the designers.

In this paper, for an input line drawing image, we propose a GAN-based stylized watercolor-like colorization method for architectural illustration. Our method is suitable for effective colorization prototyping. It has the following main features.

(1) We propose a unified framework *ArchGANs* for generating colored building images based on a line drawing with consideration of line features as well as lighting effects. It achieves

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

Figure 1: Real architectural illustration examples using line drawing with watercolor colorization.

both the stylized colorization and shading by explicitly modeling the color and lighting effect with two generative adversarial networks (GANs) respectively (namely *ArchColGAN* and *ArchShdGAN*).

(2) We propose *ArchColGAN* for stylized colorization, which can learn and predict colors based on the training data, specifically, *ArchColGAN* is able to preserve the line features well in the generated architectural illustration by our proposed line-feature-aware network structure. Our method is able to handle inputs with sparse lines as well as synthesize the typical line crossing effect at the corners.

(3) We propose *ArchShdGAN* for generating the lighting effects. To facilitate the generation of lighting effects, we introduce two major novel extensions: a diffuse equation representation of the lighting and a shading loss.

(4) Our proposed *ArchGANs* is interactive and effective. It allows user to adjust the color scheme and light direction for the generated architectural illustration. We evaluate our proposed method through various experiments and demonstrate the effectiveness and improvements of our method in overall quality comparing to the conventional methods.

2. Related work

In this section, we review the related work in stylized colorization based on Computer Graphics (CG) techniques, Convolutional Neural Networks (CNNs), and Generative Adversarial Networks (GANs) respectively.

2.1. CG-based methods

In this subsection, we mainly review CG methods related to watercolor-like colorization and line drawing. Some commercially available applications provide interactive colorization tools, such as Corel©Painter[2]. In general, many similar tools require tedious manual efforts.

Physics-based methods can achieve visually plausible colorization results for effects such as watercolor and oil painting. Curtis et al.[4] proposed to simulate the processes of water and pigment advection based on fluid simulation. Van Laerhoven and Van Reeth[7] and Chu and Tai[8] extended this approach and were able to created convincing watercolor effects in real-time with the help of GPU acceleration. However physics-based method requires high computational cost. The procedural methods to create colorization are mainly based on analyzing the image contents and applying many different types of image filter to simulate brush strokes, for example, Lei and Chang[9] used a Sobel filter to mimic the darkening effect of the stroke edges, please refer to the survey[10]. Procedural methods focus on modeling the appearance of watercolor effect instead of the physical process of watercolor. To create a watercolor-like effect for 3D models, Bousseau et al.[3] proposed to combine a series of image filters placing on 3D rendering results. Luft and Deussen[11] proposed a watercolor rendering approach especially for plants. Luft et.al[12] adopted a similar approach to create a watercolor effect of CAD data. These methods focus on producing non-photorealistic rendering effect for 3D modelings, it is not always easy to design a procedural for a specific effect.

Line drawing itself is also one NPR effect, i.e., cel-shading or tone-shading in games such as[13]. The common line drawing algorithm to render line drawing images from a given 3D model is based on mathematically defining feature lines as points on the surface which satisfy certain geometry constraints. There are many kinds of feature line definitions, e.g., silhouettes, suggestive contours[14], ridge or valley lines[15, 16] and relief edges[17].

2.2. CNN-based methods

With the rapid development in CNN, it has been proven to be a powerful technique for a variety of synthesizing tasks[18]. As the pioneering work, Gatys et al.[5] proposed a method to automatically transfer the style of an artwork to an image. Afterwards, a number of works have been proposed to extend the method. Liao et al. 2017[19] proposed image analogy to improve the image quality. To improve the efficiency, Johnson et al.[20] proposed perceptual losses and Chen et al.[21] proposed Stylebank. Chen et al.[22] proposed a extension for video style transfer. These methods are capable of handling style transfer for many artistic styles such as watercolor and oil painting. However, they usually focus on transferring textures and colors in a specific style while preserving the image content, they can not be directly applied to generate colorization while maintaining line features.

2.3. GAN-based methods

For the purpose of image-to-image translation, a number of GAN[23] methods have been proposed. Isola et al.[6] proposed the pix2pix approach that based on training with the images pairs to obtain plausible results for photo-to-label, photo-to-sketch, and photo-to-map translations. Zhu et al.[24] extended it to BicycleGAN for multi-modal translation. For unpaired image translation, a number of GAN methods have

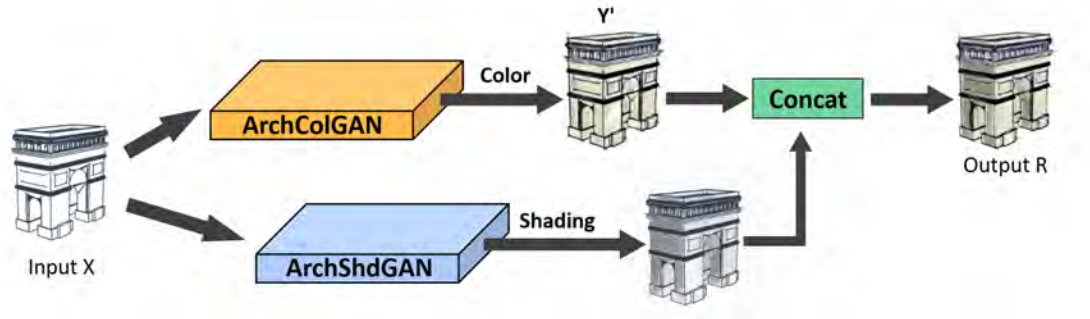

Figure 2: Main structure of the generator and discriminator networks in the proposed *ArchGANs*.

been proposed, such as CycleGAN[25], DualGAN[26], DiscoGAN[27], UNIT[28], and MNUIT[29]. Many methods such as CariGANs[30], Tag2Pix[31] were also proposed for handling 2D manga-like characters, however unlike these, building line drawings have sparse and important line features as well as large empty regions (walls) to be colored evenly. These GAN-based methods can achieve plausible results, however they mainly focus on color or texture changes only, such as horse to zebra, but not explicitly handle the line features, uneven colorization due to sparse lines as well as lighting conditions for building colorization.

3. Our Method

3.1. Main structure and training dataset

***Main structure*:** Based on an input architectural line drawing, our method aims to automatically generate a natural-looking stylized colorization with user specified light direction for architectural illustration. To tackle the problems of 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, we propose a new generative adversarial network (GAN) framework, namely *ArchGANs*.

We adopt the GAN framework[23]), as it is proven to be successful in content generation tasks. Learning-based colorization approaches are usually based on many example paired images of a line drawing with its corresponding colored painting. However, letting the designers manually create many such paired building images with different lighting conditions can be tedious and inefficient, we, therefore, propose to decouple the overall generation as two GAN branches of the similar architecture: stylized colorization (*ArchColGAN*) and lighting effect generation (*ArchShdGAN*), as shown in Fig.2. This modular structure makes our method more flexible.

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

ArchShdGAN learns a mapping $\phi_S : Y^C \rightarrow Y_S$, for generating the lighting effect $y_S \in Y_S$. Y_S denotes the lighting information domain for stylized colorization Y. By integrating the output y_S with y^C, the final result y can be obtained.

***Training dataset*:** Similarly, it is also tedious and inefficient for the designer to color every line drawings to generate paring stylized colorization, moreover, there is only a very limited number of such paired images with desired style available. As such, it is not easy to construct a training dataset for the network. Nevertheless, we collected a large number of architectural illustration images and observed that building shapes in general can be represented or constructed using a set of simpler and representative building elements: box/cylinder (e.g. building body/tower), prism/pyramid/cone/hemisphere (e.g. roof), and the building color schemes also have common patterns, e.g., many walls are colored in concrete color (grey) and the roofs are colored in brick color (dark red). Conceptually, for training purpose, we can construct the building in a “LEGO" manner. We, therefore, propose to conduct learning based on a set of simpler building shapes that are constructed by composing these building elements. By doing so, the training dataset for building architectural illustration can be more efficiently constructed without loss of general representability.

Our training datasets are built based on 10 representative simpler building shapes constructed from simple building elements and one most representative color schema. We invite the artists to paint the representative building elements (e.g box) using watercolor for the representative colors scheme. We employ 3D software (we use Autodesk 3ds Max in our implementation) to render the building shapes from 100 directions in order to generate 1000 line drawings $sdata(X) = \{x_1, x_2, \ldots\} \subset X$ as well as corresponding colored images without lighting effects $sdata(Y^C) = \{y_1^C, y_2^C, \ldots\} \subset Y^C$ as the training set for *ArchColGAN*, namely training set A.

Similarly, we select 200 colored images from $sdata(Y^C)$ and automatically generate corresponding colored images with lighting effects under different light directions. In our implementation, we use scripting in Autodesk 3ds Max and we choose 8 light directions, and the light positions are above the buildings. As the lighting model, we use point light and Phong lighting model. As such, we can build $sdata(Y) = \{y_1, y_2, \ldots\} \subset Y$ as the training set for *ArchShdGAN*, namely training set B.

For training *ArchShdGAN*, we utilize only the essential lighting effect information, which is actually $sdata(Y_S^C)$ and $sdata(Y_S)$.

In addition, to achieve desired effects, users can also specify or adjust the colorization and lighting effects.

3.2. Color translation

***ArchColGAN*:** Inspired by the GAN architecture, our proposed *ArchColGAN* includes one generator (G, as shown in Fig.3) as well as two discriminators (D, Fig.4), which are global discriminator (GD) and local discriminator (LD). It has the following features:

(1) We design the G to achieve both stylized colorization and inpainting tasks. This means, we let the G to take an input of a line drawing image with a randomly cut hole, and outputs a stylized colored inpainted image. By doing so, we increase the capability of the trained model to handle the detailed building features such as the corners as well as to learn the correlation between parts. Conceptually, we model the task of correlating parts as building connections through synthesizing (inpainting) local features, meanwhile, the local features can be emphasized, and hence be better preserved. We set the hole to fit that size of the important feature (corner) and has a normal distribution of locations for general coverage.

(2) The proposed G utilizes U-Net, which can capture the important features, in order to tackle the challenge of maintaining the line features. Its concatenation functionality improves the upsampling and the feature reuse, too.

(3) Buildings can have complex shapes, rich and vivid colors, differences also exist between the training and testing sets, and the training set is not fully paired (not pixel-to-pixel matches, some corner features can be missing). These often cause difficulties in the generator model. In order to be adaptive to such variations, on top of U-Net, we propose to integrate G with the cycle-consistency from CycleGAN[25] to enhance its robustness as well as prevent mode collapse. Essentially, we further train G with an inverse mapping model F with a cycle-consistency loss $L_{cyc}(G, F)$.

(4) The proposed G utilizes a dilated convolution to reduce the undesired uneven colorization due to sparse lines with the help of its expanded receptive field. Different from the conventional approach that using ResNet, our G employs a DenseNet as the transformer in the U-Net, which can enhance the generation of colors and line features. This also reduces the number of parameters and increases the feature reuse.

(5) The G consists of LD and GD. LD handles a patch from the output image from G at the inpainted hole position, while GD handles the whole output. By doing so, both the local features and global consistency can be preserved. Moreover, introducing LD can reduce the undesired uneven colorization by engaging G to generate better colorization locally instead of just overall plausible colorization.

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

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

The cycle-consistency loss is defined as:

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

The adversarial loss is defined as:

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

***Implementation*:** The input to G is a line drawing image of 256×256 pixel resolution, with a hole. The hole is of 40×40 pixel resolution, its centre position follows the normal distribution, its boundary has a 5 pixels padding margin to the image boundary.

As shown in Fig.3, the U-Net of G starts with two Flatten layers. Each Flatten layer uses a 7×7 convolution (Conv) kernel with a step size of one, an instance normalization function (IN), and a rectified linear unit (Relu) with a fixed size of the output feature map. Afterwards, three downsampling convolution blocks, namely encoding blocks, are applied, each consists of a Conv followed by a Flatten layer for image compressing and encoding. The useful and important local features can be fetched for the later translation. This downsampling uses a 3×3 kernel with a step size of two, we double the number of feature channels each step.

Then the dilated convolution, which allows more area can be used as input by expanding convolution kernel without increasing learn-able weights. For a 2D layer of C channel $h \times w$ mapping and a next layer of C' channel $h' \times w'$, the expanding convolution operator of each pixel is defined as:

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

where k_w and k_h are the kernel width and height (odd numbers), respectively, η is the dilation factor, $x_{u,v} \in R_C$ and $y_{u,v} \in R_{C'}$ are the pixel component of the input and output of the layer, $\sigma(\cdot)$ is a component-wise non-linear transfer function, $W_{s,t}$ are C'-by-C matrices of the kernel, and $b \in R_{C'}$ is the layer bias vector. When $\eta = 1$, the equation becomes the standard convolution operation. In our implementation, we use $\eta = 2, 4, 8$, respectively.

Afterwards, two dense network blocks (DenseNet-BC) are employed. There is a 1×1 Conv compression layer between blocks with a compression factor of 0.5. Furthermore, each

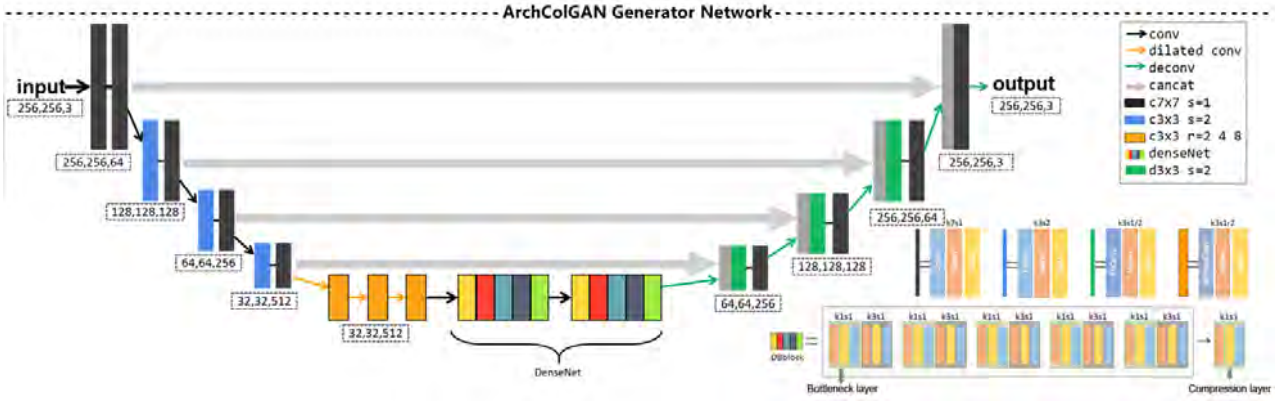

Figure 3: The generator network of *ArchColGAN*. c of $c7 \times 7$ is conv, d of $d3 \times 3$ is deconv, s is step size, r is dilation factor, same for the other Figures.

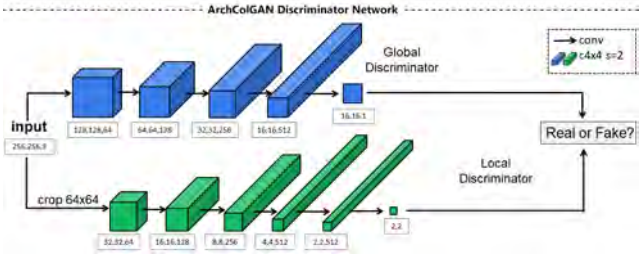

Figure 4: The discriminator network of *ArchColGAN*.

block has 5 layers: an IN, a Relu combined with a 3×3 Conv with a growth rate of $k = 32$ for each layer, and a bottleneck layer of the dense network applied beforehand which consists of an IN function and a Relu combined with a 1×1 Conv layer. In order to reduce the amount of input feature maps, thus to reduce the computational cost, in our implementation, we let each 1×1 Conv generate $4k$ feature maps. To this end, we name the structure that combines 5 layers of IN-Relu-Conv(1×1)-IN-Relu-Conv(3×3) as a dense network block (DB), and use 2 structures of DB-IN-Relu-Conv(1×1) as the transformer of the generator.

Then, we apply three upsampling convolution blocks, namely decoding block, each block is followed by a Flatten layer, to reconstruct and output the stylized colorization. The upsampling convolution is a deconvolution layer with a kernel of 3×3 and a step size of two. The Flatten layer is the same as the encoding block. At each upsampling step, we reduce the number of feature channels to half of its previous step. We then apply the final Conv layer with a kernel size of 3×3. We concatenate the output features of each encoding block and the output features of its corresponding decoding block as the input of the respective next decoding block. After this, the cycle-consistency is applied.

As shown in Fig.4, *GD* takes the whole generated image of 256×256 pixels as the input. It consists of 4 layers of down-sampling layers (Conv 4×4, step size 2) and a Conv layer (Conv 4×4, step size 1). It outputs a 16×16 matrix for computing differences to the real data. The *LD* has the same structure as the *GD*. For *LD*, we resize the 40×40 patch at the respective inpainted hole location to 60×60 as the input and it outputs a 2×2 matrix.

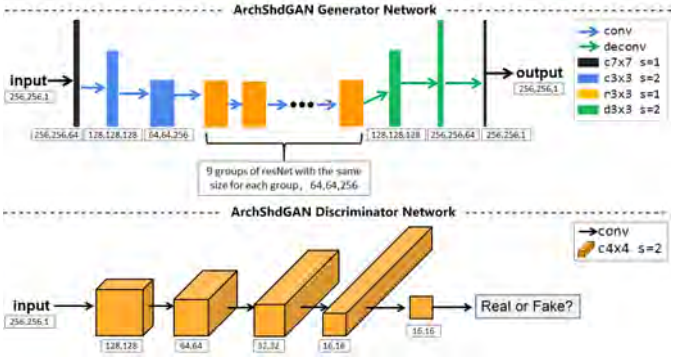

Figure 5: The generator and discriminator networks of *ArchShdGAN*. Note that r means ResNet here.

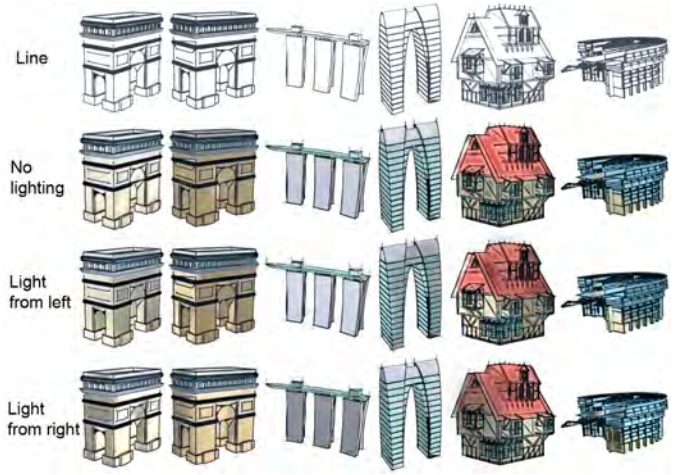

Figure 6: Adding lighting effects.

3.3. Lighting effect

***ArchShdGAN*:** Our method is based on the CycleGAN[25] architecture with generator and discriminator. Different from the CycleGAN, which mainly focuses on image-to-image translation, our proposed *ArchShdGAN* aims to handle effect-to-effect translation for lighting effects of buildings. We observe that the Value channel of the HSV (Hue, Saturation, Value) color space representation can plausibly depict and represent the general lighting effect perceived by the viewer. As such, we proposed to formulate this lighting effect-to-effect translation as a Value-to-Value translation: $Y_S \approx Y_V$, Y_V is the Value attribute.

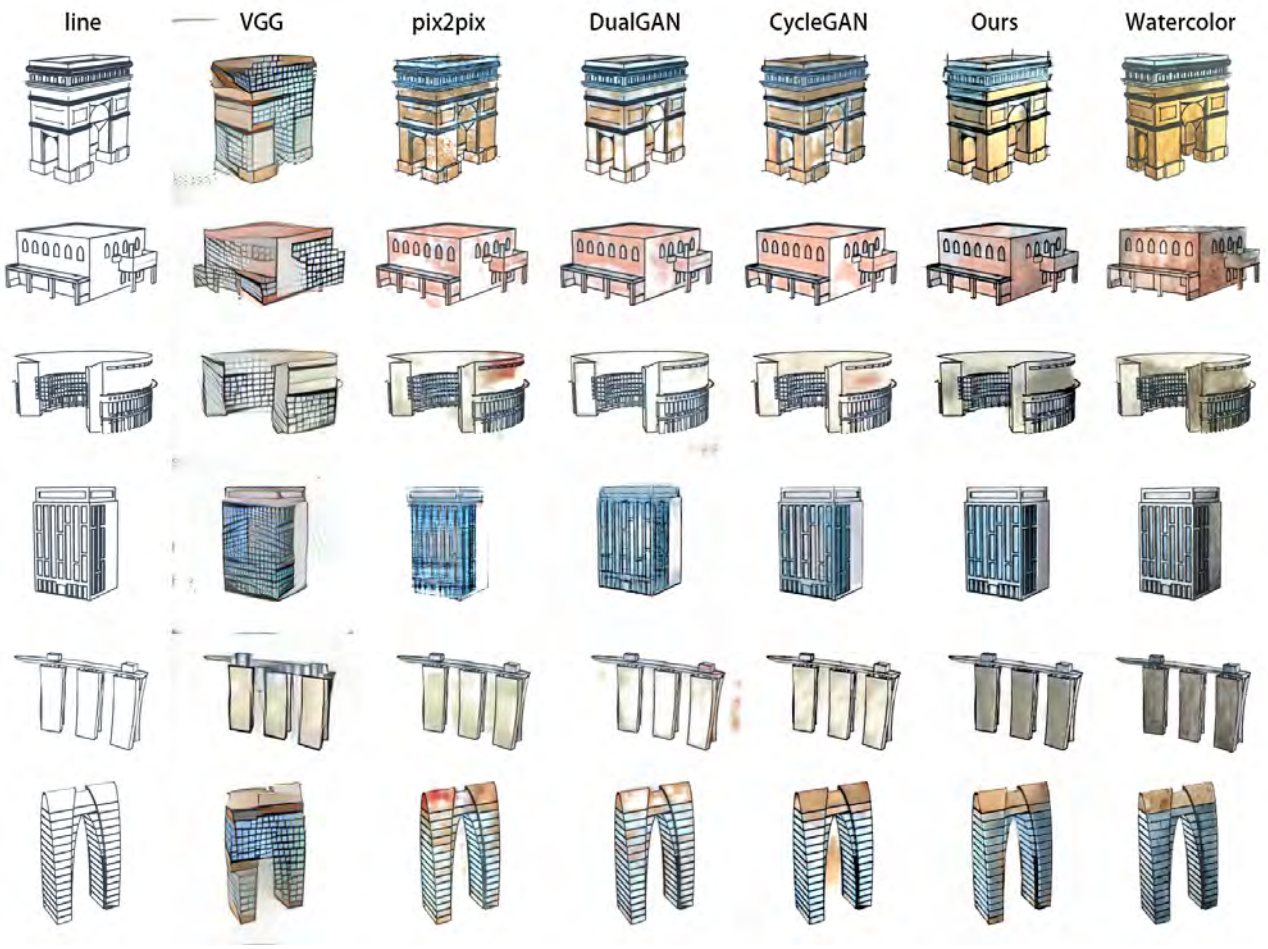

Figure 7: Comparison with different methods. From left column to the right: line drawing input images, results generated by VGG, pix2pix, DualGan, CycleGAN, our method, and an artist using watercolor.

We convert the colored building image (y^C) of RGB format into HSV format, and fetch its Value attribute as the input of *ArchShdGAN*, $y_S^C \approx y_V^C$. Its generator has following steps. After one flattening layer and two layers of downsampling convolution encoding, RasNet is subsequently applied for translation. Then, two layers of upsampling transposed convolution followed by one flattening layer are employed to synthesize the Value attribute. This synthesized Value attribute from the generator is then inputted to the discriminator, which consists of four downsampling convolution layers. The loss is computed and evaluated, incorporating with discriminator to achieve the adversarial generation for the Value attribute, as shown in Fig.5.

***Loss*:** G_S^*, D_S^* denote the weights of the generator and discriminator of *ArchShdGAN* respectively. L_S denotes the loss function. To this end, we aim to solve the minimization/maximization problem that G_S tries to minimize the objective $L_S(G_S, D_S)$ against an adversary D_S that aims to maximize it, as follows:

$$(G_S^*, D_S^*) = \arg\min_{G_S} \max_{D_S} L_S(G_S, D_S)$$

$$L_S(G_S, D_S) = L_{S_{adv}}(G_S, D_S) + L_{S_{cyc}}(G_S, F_S).$$

The adversarial loss $L_{S_{adv}}$ is defined as:

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

The bidirectional cycle-consistency loss $L_{S_{cyc}}$ is defined as:

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

***Concatenation*:** As the final step, the output y_S is concatenated with the Hue and Saturation attributes of y^C and convert to RGB format as the final resulting image y of stylized colorization and lighting effect.

4. Result and discussion

Our implementation of the proposed method is based on TensorFlow framework. We conducted the experiments on a Desktop PC with Intel Core i7-7700K CPU and two Nvidia GeForce GTX1070 GPUs.

We apply our method on a number of representative architectural line drawings, with several user-defined color schemas.

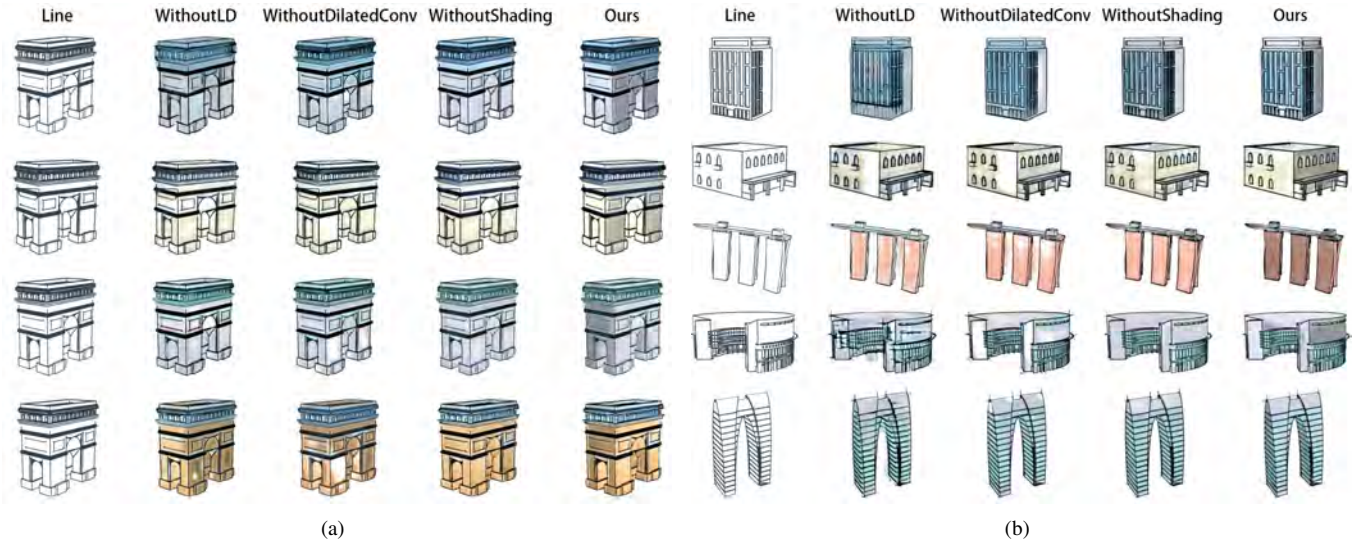

Figure 8: Ablation study. (a) Different color schemas. (b) Different building models.

As a result, in general, we are able to generate visually plausible stylized colorization, as shown in Figures 6, 7, 8, and in the supplemental material, our method is capable of maintaining line features, reducing unwanted uneven colorization while augmenting with lighting effects.

As shown in Fig.7 We evaluate our method by comparing the results with the ground truth colorization which is produced by a professional artist as well as comparing with colorization results generated using the state-of-the-art representative GAN methods, which are VGG[23], pix2pix[6], DualGAN[26], and CycleGAN[25]. For the input line drawings, we apply the same color schema as the ground truth. Moreover, we also conduct a user study for qualitative evaluation. Specifically, the evaluation is conducted from three important aspects, lines, colors as well as lighting effects. Line evaluation is mainly conducted in three perspectives: maintaining the line structure, stylizing and repairing the lines. color evaluation is mainly conducted in three perspectives: evenly coloring of large walls, color consistency of windows, and distinction between the main building and the background.

Line evaluation: In general, the line structure of the results by using VGG network can not be maintained well (Fig.7 column two). This is mainly because the performance of VGG is highly dependent on the comprehensiveness of training datasets. However, in most cases, we could encounter relatively large differences between the test and training datasets, thus, using VGG would lead to an unexpected result. VGG tends to perform better for the task of transferring textures and colors, however it can not well preserve the line structure. Different to VGG, the results of other methods can achieve satisfying results for maintaining line structure.

In Fig.7 first row, we use the triumphal arch model to demonstrate the stylized transformation of a line drawing. Note that the input line drawing should be transformed into a stylized line drawings with corner crossing features. In architectural drawings, these corner crossings are very common and are largely used as the references for perspective illustration. As shown in the figure, in the results of CycleGAN and our method, the corner crossing features are much clearer, this means better stylized transformation of the lines. However, in the results from other methods, these features are hardly visible, in other words, the line drawings are less stylized. This is mainly due to CycleGAN and our method can better handle such local features.

Color evaluation: The first, second and third rows of Fig.7 demonstrate the capability of handling the coloring for large area walls. It can be seen that, when the wall area is larger, such as the pillar faces of the triumphal arch in the first row, and the external walls of the house in the second row, the colorization results of using pix2pix, DualGAN and CycleGAN methods have larger blank areas in the center of the walls, pix2pix and CycleGAN may generate unwanted red color for the wall area (third row), such uneven colorization is unnatural and can degrade the overall perception experiences. On the other hand, in the result of our method, the colorization of large area wall is more even, the undesired blank area is reduced.

The fourth and sixth rows of Fig.7 demonstrate the capability of handling color consistency in windows. Different from other methods, in the result of our method, the window color (blue) has higher consistency.

The fifth and sixth rows of Fig.7 mainly assess the distinction between the foreground (the main building) and the background. Note that in other methods, the color may leak to the empty region between building pillars, which is undesired for colorization.

Lighting effect evaluation: Our results can plausible represent the 3D effect of the building and the direction of the light source, as shown in Fig.6, which is mainly with the help of the *ArchShdGAN* module in our network structure.

User study: We also conducted a user study using mean opinion score (MOS) to evaluate our method. We asked 17 participants with art background to evaluate various colorization results and give opinion scores. They are asked to evaluate based on the visual quality and perceptual experience. Twenty ground truth colorization by artists are provided to the participants for ref-

erence. The opinion scores are ranging from 5 to 1: 5 (Very good), 4 (Good), 3 (Average), 2 (Bad), 1 (Very bad). The results are generated using 8 methods: VGG, pix2pix, DualGAN, CycleGAN, ArchGANs without LD, ArchGANs without DilatedConv (DC), ArchGANs without Shading, and ArchGANs. For each method, we generated 62 different colorization results. Each participant evaluated 496 results and we got 8432 scores. The MOS of the results is shown in Fig.9.

VGG	pix2pix	DualGAN	CycleGAN	Ours without LD	Ours without DC	Ours without Shading	Ours
1.214	2.170	2.523	2.995	2.840	3.254	3.869	4.261

Figure 9: User feedback of the colorization results, in mean opinion score.

From the user study, we learned that ArchGANs without LD (MOS 2.840) performs worse than the state-of-the-art CycleGAN (MOS 2.995), however, incorporating LD, DilatedConv and Shading can largely improve the result (MOS 4.261), and give a better visual experience to the viewer. Component-wise, adding LD (MOS from 2.840 to 4.261) and DC (MOS from 3.254 to 4.261) are more important components for improving results comparing to adding Shading effect (MOS from 3.869 to 4.261), this indicates that the users usually are more sensitive to the color effect such as the evenness of the colorization than the lighting effect.

Ablation study: To further justify the effectiveness of our proposed method, we also conducted an ablation study. In this study, we compare the results from ArchGANs without LD, ArchGANs without DilatedConv (DC), ArchGANs without Shading, and ArchGANs, using different color schemas (Fig.8 (a)) and building models (Fig.8 (b)).

From this study, we can observe that our model with LD and GD is helpful to handle detailed local features such as the features in the bottom of the building in row one of Fig.8 (b), adding DilatedConv can further reduce the uneven colorization in the large walls. Adding the lighting effects can better depict the building in 3D.

5. Conclusion and future work

In this paper, we have presented *ArchGAN*, a novel GAN approach to generate stylized colorization for architectural line drawing. As the fist component, we have proposed *ArchColGAN*, through designing it to perform both stylized colorization and inpainting tasks, utilizing U-Net, and incorporating cycle consistency, dilated convolution, and two-stage discriminators (local and global), we can well achieve stylized colorization. As the second component, we have proposed *ArchShdGAN* for lighting effect augmentation. Different from the existing methods, the proposed method can well support the handling of line features, even colorization, and lighting effects. The results and evaluation have demonstrated the effectiveness of our proposed method.

In the future, we plan to apply the proposed approach in more scenarios such as animation and movie. We also would like to extend our method to handle other types of objects and effects such as plants, streets, sky and shadows.

Acknowledgments: We gratefully thank the reviewers for their constructive comments. This research is supported by NSFC Grants (61702363, 51978441) 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.

References

[1] T. W. Schaller, *The art of architectural drawing: imagination and technique*. Wiley. com, 1997.
[2] Corel, *Painter 12*. www.corel.com, 2011.
[3] A. Bousseau, M. Kaplan, J. Thollot, and F. X. Sillion, "Interactive watercolor rendering with temporal coherence and abstraction," in *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, ser. NPAR '06, 2006.
[4] C. J. Curtis, S. E. Anderson, J. E. Seims, K. W. Fleischer, and D. H. Salesin, "Computer-generated watercolor," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '97, 1997.
[5] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2414–2423.
[6] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.
[7] T. Van Laerhoven and F. Van Reeth, "Real-time simulation of watery paint: Natural phenomena and special effects," *Comput. Animat. Virtual Worlds*, vol. 16, no. 3-4, Jul. 2005.
[8] N. S.-H. Chu and C.-L. Tai, "Moxi: real-time ink dispersion in absorbent paper," *ACM Trans. Graph.*, vol. 24, no. 3, Jul. 2005.
[9] S. I. E. Lei and C.-F. Chang, "Real-time rendering of watercolor effects for virtual environments," in *Proceedings of the 5th Pacific Rim conference on Advances in Multimedia Information Processing - Volume Part III*, ser. PCM'04, 2004.
[10] A. Hertzmann, "Tutorial: A survey of stroke-based rendering," *IEEE Comput. Graph. Appl.*, vol. 23, no. 4, Jul. 2003.
[11] T. Luft and O. Deussen, "Real-time watercolor illustrations of plants using a blurred depth test," in *Proceedings of the 4th international symposium on Non-photorealistic animation and rendering*, ser. NPAR '06, 2006.
[12] T. Luft, F. Kobs, W. Zinser, and O. Deussen, "Watercolor illustrations of cad data," in *Proceedings of the Fourth Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging*, ser. Computational Aesthetics'08, 2008.
[13] Capcom, "Street fighter iv," 2008.
[14] D. DeCarlo, A. Finkelstein, S. Rusinkiewicz, and A. Santella, "Suggestive contours for conveying shape," *ACM Transactions on Graphics*, vol. 22, pp. 848–855, 2003.
[15] Y. Ohtake, A. Belyaev, and H. Seidel, "Ridge-valley lines on meshes via implicit surface fitting," in *Proceedings of ACM SIGGRAPH*, 2004, pp. 609–612.
[16] T. Judd, F. Durand, and E. H. Adelson, "Apparent ridges for line drawing," *ACM Transactions on Graphics*, vol. 26, no. 3, p. 19, 2007.
[17] M. Kolomenkin, I. Shimshoni, and A. Tal, "On edge detection on surfaces," in *CVPR'09*, 2009, pp. 2767–2774.
[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.
[19] J. Liao, Y. Yao, L. Yuan, G. Hua, and S. B. Kang, "Visual attribute transfer through deep image analogy," *arXiv preprint arXiv:1705.01088*, 2017.
[20] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*. Springer, 2016, pp. 694–711.

[21] D. Chen, L. Yuan, J. Liao, N. Yu, and G. Hua, "Stylebank: An explicit representation for neural image style transfer," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1897–1906.

[22] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua, "Coherent online video style transfer," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1105–1114.

[23] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," in *Proceedings of the International Conference on Neural Information Processing Systems*, 2014, p. 2672–2680.

[24] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman, "Toward multimodal image-to-image translation," in *Advances in Neural Information Processing Systems*, 2017, pp. 465–476.

[25] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[26] Z. Yi, H. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2849–2857.

[27] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1857–1865.

[28] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in neural information processing systems*, 2017, pp. 700–708.

[29] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 172–189.

[30] K. Cao, J. Liao, and L. Yuan, "Carigans: unpaired photo-to-caricature translation," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–14, 2018.

[31] H. Kim, H. Y. Jhoo, E. Park, and S. Yoo, "Tag2pix: Line art colorization using text tag with secat and changing loss," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9056–9065.

ArchGANs: stylized colorization prototyping for architectural line drawing

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

[1]College of Intelligence and Computing, Tianjin University, Tianjin, China [2]Fraunhofer Singapore, Singapore

Supplemental Material: more results with different models and color schemas

*qian.sun@tju.edu.cn