Learning to Shadow Hand-drawn Sketches

Qingyuan Zheng*
University of Maryland,
Baltimore County
qing3@umbc.edu

Zhuoru Li*
Project HAT
hatsuame@gmail.com

Adam Bargteil
University of Maryland,
Baltimore County
adamb@umbc.edu

Abstract

We present a fully automatic method to generate detailed and accurate artistic shadows from pairs of line drawing sketches and lighting directions. We also contribute a new dataset of one thousand examples of pairs of line drawings and shadows that are tagged with lighting directions. Remarkably, the generated shadows quickly communicate the underlying 3D structure of the sketched scene. Consequently, the shadows generated by our approach can be used directly or as an excellent starting point for artists. We demonstrate that the deep learning network we propose takes a hand-drawn sketch, builds a 3D model in latent space, and renders the resulting shadows. The generated shadows respect the hand-drawn lines and underlying 3D space and contain sophisticated and accurate details, such as self-shadowing effects. Moreover, the generated shadows contain artistic effects, such as rim lighting or halos appearing from back lighting, that would be achievable with traditional 3D rendering methods.

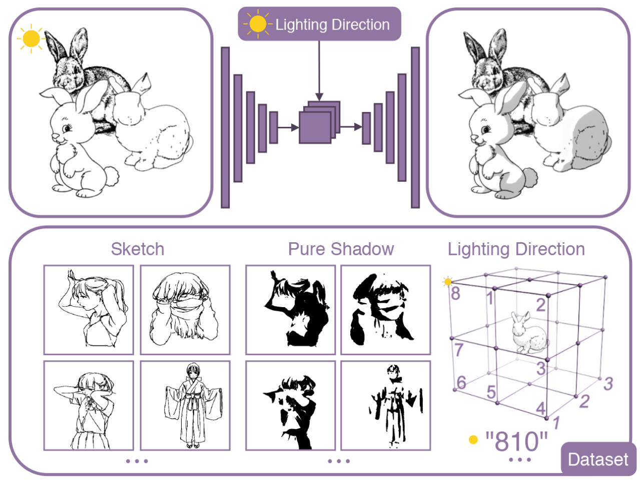


Figure 1: Top: our shadowing system takes in a line drawing and a lighting direction label, and outputs the shadow. Bottom: our training set includes triplets of hand-drawn sketches, shadows, and lighting directions. Pairs of sketches and shadow images are taken from artists' websites and manually tagged with lighting directions with the help of professional artists. The cube shows how we denote the 26 lighting directions (see Section 3.1). ©Toshi, Clement Sauve

1. Introduction

Shadows are an essential element in both traditional and digital painting. Across artistic media and formats, most paintings are first sketched with lines and shadows before applying color. In both the Impressionism and Neoclassicism era, artists would paint oil paintings after they rapidly drew shadowed sketches of their subjects. They recorded what they saw and expressed their vision in sketches and shadows and used these as direct references for their paintings [1].

In the modern painting era, particularly for digital illustration and cel animation, shadows play an important role in depicting objects' shapes and the relationships between 2D lines and 3D space, thereby affecting the audience's recognition of the scene as whole. Illustration is a time-consuming process; illustrators frequently spend several hours drawing an appealing picture, iteratively adjusting the form and structure of the characters many times. In addition to this work, the illustrators also need to iteratively adjust and refine the shadows, either after completing the sketch or while iterating the sketching process. Drawing shadows is particularly challenging for 2D sketches that cannot be observed in the real world, because there is no 3D reference model to reason about; only the artist's imagination. In principal, the more details the structural lines contain, the more difficult it is to draw the resulting shadows. Hence adjusting the shadows can be time consuming, especially for inexperienced illustrators.

In this paper, we describe a real-time method to generate plausible shadows from an input sketch and specified lighting direction. These shadows can be used directly, or

*Equal contribution

if higher quality is desired can be used as a starting point for the artists to modify. Notably, our approach does not generate shadowed sketches directly; instead it generates a separate image of the shadow that may be composited with the sketch. This feature is important as the artist can load the sketch and the shadow into separate image layers and edit them independently.

Our work uses the deep learning methodology to learn a non-linear function which "understands" the 3D spatial relationships implied by a 2D sketch and render the binary shadows (Figure 1 top). The raw output from our neural network is binary shadows, which may be modified by artists in a separate layer independent of line drawings. There is no additional post-processing and the images in our paper are simple composites of the raw network outputs and the input line drawings. If soft shadows are desired, artists may use the second intermediate output from our network (Figure 2 s_2). Our network also produces consistent shadows from continuously varying lighting directions (Section 4.3), even though we train from a discrete set of lighting directions.

Given a line drawing and a lighting direction, our model automatically generates an image where the line drawing is enhanced with detailed and accurate hard shadows; no additional user input is required. We focus on 2D animation style images (*e.g.* Japanese comic, Inker [37]) and the training data is composed of artistic hand-drawn line drawing in the shape of animation characters, mecha, and mechanical objects. We also demonstrate that our model generalizes to line drawing of different objects such as buildings, clothes, and animals.

The term "artistic shadow" in our work refers to binary shadows that largely obey physics but also have artistic features such as less shadowing of characters' faces and rim lighting when characters are back lit.

The main contributions of our work:

- We created a new dataset that contains 1,160 cases of hand-drawn line drawings and shadows tagged with lighting directions.
- We propose a network that "understands" the structure and 3D spatial relationships implied by line drawings and produces highly-detailed and accurate shadows.
- An end-to-end application that can generate binary or soft shadows from arbitrary lighting directions given a 2D line drawing and designated lighting direction.

In Section 3, we will decribe the design of our generative and discriminator networks, and our loss functions. In Section 4, we compare our results quantitatively and qualitatively to baseline network architectures pix2pix [16] and U-net [28]. We also compare to the related approaches Sketch2Normal [32] and DeepNormal [14] applied to our shadow generation problem. Our comparisons include a small user study to assess the perceptual accuracy of our approach. Finally, we demonstrate the necessity of each part of our proposed network through an ablation study and metrics analysis. [1]

2. Related Work

Non-photorealistic rendering in Computer Graphics. The previous work on stylized shadows [26, 3] for cel animation highlights that shadows play an important role in human perception of cel animation. In particular, shadows provide a sense of depth to the various layers of character, foreground, and background. Lumo [17] approximates surface normals directly from line drawings for cel animation to incorprate subtle environmental illumination. Todo *et al.* [35, 36] proposed a method to generate artistic shadows in 3D scenes that mimics the aesthetics of Japanese 2D animation. Ink-and-Ray [34] combined a hand-drawn character with a small set of simple annotations to generate bas-relief sculptures of stylized shadows. Recently, Hudon *et al.* [13] proposed a semi-automatic method of cel shading that produces binary shadows based on hand-drawn objects without 3D reconstruction.

Image translation and colorization. In recent years, the research on Generative Adversarial Networks (GANs) [7, 24] in image translation [16] has generated impressive synthetic images that were perceived to be the same as the originals. Pix2pix [16] deployed the U-net [28] architecture in their Generator network and demonstrated that for the application of image translation U-net's performance is improved when skip connections are included. CycleGAN [44] introduced a method to learn the mapping from an input image to a stylized output image in the absence of paired examples. Reearch on colorizing realistic gray scale images [2, 42, 15, 43] demonstrated the feasibility of colorizing images using GANs and U-net [28] architectures.

Deep learning in line drawings. Researcher that considers line drawings include line drawing colorization [39, 19, 41, 5, 4], sketch simplification [31, 29], smart inker [30], line extraction [21], line stylization [22] and computing normal maps from sketches [32, 14]. Tag2Pix [19] seeks to use GANs that concatenate Squeeze and Excitation [12] to colorize line drawing. Sketch simplification [31, 29] cleans up draft sketches, through such operations as removing dual lines and connecting intermittent lines. Smart inker [30] improves on sketch simplification by including additional user input. Users can draw strokes indicating where they would like to add or erase lines, then the neural network will output a simplified sketch in real-time. Line extraction [21] extracts pure lines from manga (comics) and demonstrates that simple downscaling and upscaling residual blocks with

[1] Project page is at `https://cal.cs.umbc.edu/Papers/Zheng-2020-Shade/`.

skip connections have superior performance. Kalogerakis *et al*. [18] proposed a machine learning method to create hatch-shading style illustrations. Li *et al*. [22] proposed a two-branch deep learning model to transform the line drawings and photo to pencil drawings.

Relighting. Deep learning has also been applied to relighting realistic scenes. Xu *et al*. [38] proposed a method for relighting from an arbitrary directional light given images from five different directional light sources. Sun *et al*. [33] proposed a method for relighting portraits given a single input, such as a selfie. The training datasets are captured by a multi-camera rig. This work differs from ours in that they focus on relighting realistic images while we focus on artistic shadowing of hand-drawn sketches.

Line drawings to normal maps. Sketch2normal [32] and DeepNormal [14] use deep learning to compute normal maps from line drawings. Their training datasets are rendered from 3D models with realistic rendering. Sketch2Normal trains on line drawings of four-legged animals with some annotations. DeepNormal takes as input line drawings with a mask for the object. They solve a different, arguably harder, problem. However, the computed normal maps can be used to render shadows and we compare this approach to our direct shadow computation in Section 4. Given color input images, Gao and colleagues [6] predict normal maps and then generate shadows.

3. Learning Where to Draw Shadows

In this section we describe our data preparation, our representation of the lighting directions, the design of our generator and discriminator networks, and our loss functions.

3.1. Data Preparation

We collect our (sketch, shadow) pairs from website posts by artists. With help from professional artists, each (sketch, shadow) pair is manually tagged with a lighting direction. After pre-processing the sketches with thresholding and morphological anti-aliasing, the line drawings are normalized to obtain a consistent line width of 0.3 px in *cairosvg* standard [27]. To standardize the hand-drawn sketch to the same line width, we use a small deep learning model similar to smart inker [30] to pre-process input data. Our dataset contains 1,160 cases of hand-drawn line drawings. Each line drawing matches one specific hand-drawn shadow as ground truth and one lighting direction.

In contrast to 3D computer animation, which contains many light sources and realistic light transport, 2D animation tends to have a single lighting direction and include some non-physical shadows in a scene.

We observed that artists tend to choose from a relatively small set of specific lighting directions, especially in comics and 2D animation. For this reason, we define 26 lighting directions formed by the 2×2 cube in Figure 1. We found that it was intuitive to allow users to choose from eight lighting directions clockwise around the 2D object and one of three depths (in-front, in-plane, and behind) to specify the light source. We also allow the user to choose two special locations: directly in front and directly behind. This results in $8 \times 3 + 2 = 26$ lighting directions. The user specifies the light position with a three-digit string. The first digit corresponds to the lighting direction (1-8), the second to the plane (1-3), and the third is '0' except for the special directions, which are "001" (in-front) and "002" (behind).

While users found this numbering scheme intuitive, we obtained better training results by first converting these strings to 26 integer triples on the cube from $[-1, 1]^3$ ($(0, 0, 0)$ is not valid as that is the location of the object). For example, "610" is mapped to $(-1, -1, -1)$, "230" is mapped to $(1, 1, 1)$, and "210" is mapped to $(1, 1, -1)$.

3.2. Network Architecture

Our generator incorporates the following modules: residual blocks [9] [10], FiLM [25] residual blocks, and Squeeze-and-Excitation (SE) blocks [12]. The general architecture of our generator follows the architecture of U-net with skip connections [28, 16]. Our Discriminator uses residual blocks. Details are shown in Figure 2.

3.2.1 Generative Network

We propose a novel non-linear model with two parts - *ShapeNet*, which encodes the underlying 3D structure from 2D sketches, and *RenderNet*, which renders artistic shadows based on the encoded structure.

ShapeNet encodes a line drawing of an object into a high dimensional latent space and represents the object's 3D geometric information. We concatenate 2D coordinate channels [23] to the line drawings to assist *ShapeNet* in encoding 3D spatial information.

RenderNet performs reasoning about 3D shadows. Starting from the bottle neck, we input the embedded lighting direction using the normalization method from FiLM residual blocks [25]. The model then starts to learn the relationship between the lighting direction and the various high dimensional features. We repeatedly add the lighting direction into each stage of the *RenderNet* to enhance the reasoning of decoding. In the bottom of each stage in *RenderNet*, a Self-attention [40] layer complements the connection of holistic features.

The shadowing problem involves holistic visual reasoning because shadows can be cast by distant geometry. For this reason we deploy Self-attention layers [40] and FiLM residual blocks [25] to enhance the visual reasoning; networks that consist of only residual blocks have limited receptive fields and are ill-suited to holistic visual reasoning. The SE [12] blocks filter out unnecessary features imported

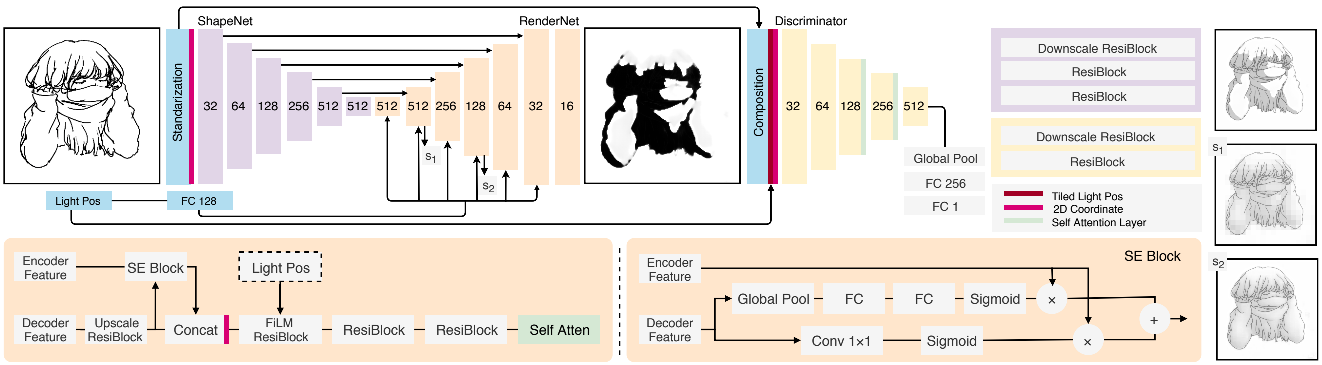


Figure 2: Our GANs architecture. The line drawings are standardized first (same as in Section 3.1) before being inputted into the *ShapeNet*. Lighting directions are repeatedly added into the FiLM residual block in each stage in *RenderNet*. s_1 and s_2 are the up-sampled intermediate outputs from the second and the forth stage in *RenderNet*. In the training process, the line drawings and pure shadows are inverted from black-on-white to white-on-black. More details are in supplementary material.

from the skipped encoder output.

We also extract two supervision intermediate outputs, s_1 and s_2, to facilitate backpropagation. Early stages of our *RenderNet* generate continuous, soft shadow images. In the final stage, the network transforms these images to binary shadows. The quality of the soft shadows in the intermediate outputs, s_1 and s_2, is shown in Figure 2. We note again that our output does not require any post processing to generate binary shadows; the images in this paper result directly from compositing the output our generator with the input sketch.

3.2.2 Discriminator Network

The basic modules of our discriminator include downscaling residual blocks and residual blocks. Since many local features of different shadows are similar to one another, we deploy Self-attention layers to make our discriminator sensitive to the distant features. In Figure 2, the last of the discriminator consists of global average pooling, dropout with 0.3 probabilities, and a fully connected layer with 256 filters. Because generating shadows is more difficult than discriminating between fake and real shadows, a simple discriminator is sufficient and simplifies training.

3.3. Loss Function

The adversarial loss of our Generative Adversarial Network can be expressed as

$$\begin{aligned} \mathcal{L}_{cGAN}\left(G,D\right) = \mathbb{E}_{x,y,z}\left[logD\left(C\left(x,y\right),z\right)\right] \\ +\mathbb{E}_{x,z}\left[log\left(1-D\left(C\left(x,G\left(x,z\right)\right),z\right)\right)\right], \end{aligned} \tag{1}$$

where x is the sketch, y is the ground truth shadow, and z is the lighting direction. $C(\cdot)$ is a function that composite the ground truth shadow and the input sketch as a "real" image, and composite the generated shadow and the input sketch as a "fake" image.

The generator G aims to minimize the loss value, and the discriminator D aims to maximize the loss value. For the loss value of our generator network, we add MSE losses of the two deep supervised outputs, which are the intermediate outputs of the first and third stage in the decoder, to the loss of the generator's final output.

The three losses of the generator network can be expressed as

$$\mathcal{L}_{output}(G) = \mathbb{E}_{x,y,z}\left[\|y - G(x,z)\|_2^2\right] + \xi \cdot \mathrm{TV}\left(G(x,z)\right), \tag{2}$$

where $\mathcal{L}_{output}$ is the loss between generated shadow and the ground truth. $\mathcal{L}_{output}$ consists of a total variation (TV) regularizer and an MSE loss. The TV regularizer, weighted by ξ, encourages smooth details around the boundaries of shadows. We set ξ to 2×10^{-6}, a $5\times$ smaller value than the total number of pixels in the input sketch. We will show how the value of ξ affects the final output in the ablation study. The deep supervised outputs are upsampled and their losses are computed as by MSE loss from ground truth,

$$\mathcal{L}_{s_i}\left(G\right) = \mathbb{E}_{x,y,z}\left[\|y - G_{s_i}(x,z)\|_2^2\right], \ i = 1, 2. \tag{3}$$

Final objective is the sum of $\mathcal{L}_{output}$, $\mathcal{L}_{s_1}$, $\mathcal{L}_{s_2}$, and the $\mathcal{L}_{cGAN}$,

$$\begin{aligned} G^* = arg \min_G \max_D \lambda_1 \mathcal{L}_{cGAN}\left(G,D\right) \\ +\lambda_2 \mathcal{L}_{output}\left(G\right) + \lambda_3 \mathcal{L}_{s_1}\left(G\right) + \lambda_4 \mathcal{L}_{s_2}\left(G\right). \end{aligned} \tag{4}$$

In our experiments, the four losses are weighted by $\lambda_1 = 0.4$, $\lambda_2 = 0.5$, $\lambda_3 = 0.2$, and $\lambda_4 = 0.2$.

4. Experiments and Evaluation

In this section, we evaluate the performance of our shadowing model. In particular, we discuss implementation details, provide comparisons with the baseline pix2pix [16]

Figure 3: Shadows for lighting depth "1" - in front of the plane (front lighting), compared with previous work DeepNormal [14] and Sketch2Normal [32]. The little sun denotes the lighting direction. ©Derori-san, Imomushi-san, Eric ou

and U-net [28] and the previous work DeepNormal [14] and Sketch2Normal [32], describe a small user study, and detail our ablation study.

4.1. Implementation Details

All the lines of sketch images in our dataset are normalized and thinned to produce a standard data representation. If the user input sketch is not normalized and thinned, we apply a pre-trained line normalization model modified from [30] to preprocess the user input.

In the training process, the line drawings are first inverted from black-on-white to white-on-black and input to the network. The final output and the intermediate outputs s_1 and s_2 from the generator are similarly white shadows on black backgrounds. Inverting the images causes the network to converge faster. The generated shadows are composited with the line drawings as the "fake" image input to the discriminator. Similarly we composite the sketch and pure shadow in our dataset as the "real" image input to discriminator.

Because of limited size of our dataset of sketch/shadow pairs with annotated lighting direction we used the entire dataset for training—we did not reserve any of our training dataset for testing. We trained for $80,000$ iterations with Adam optimizer [20]. The optimizer parameters are set to $\text{learning rate} = 0.0002$, $\beta_1 = 0$, and $\beta_2 = 0.9$. The network is trained using one 12G Titan Xp with a batch size of 8 and 320 × 320 input image size.

We shift, zoom in/out, and rotate to augment our dataset. When we rotate our line drawing input by each of {0, 45, 90, 135, 180, 225, 270, 315} degrees, we also rotate the ground truth shadow images and modify the lighting direction labels, by adding 1 to the first digit for every 45 degrees of rotation. Shifting and zooming does not affect the lighting direction.

4.2. Comparison with Prior Work

In this subsection, we qualitatively compare our approach to DeepNormal [14] and Sketch2Normal [32]. Also, we compare our network to two baselines, Pix2pix [16] and U-net [28]. The evaluation dataset is not included in training. The line drawings (without shadows) used for evaluation are collected from other artists and prior work to which we compare.

We generated the output from DeepNormal and Sketch2Normal using their source codes and trained models, unmodified. We use the scripts provided by DeepNormal to render shadows from normal maps. All normal maps are rendered under the same settings in this paper. To generate binary shadows, we threshold the continuous shadings at 0.5. We note that DeepNormal additionally requires a mask to eliminate space outside the object; Sketch2Normal and our work do not require this mask. We provide a hand-drawn mask as input to DeepNormal. Our

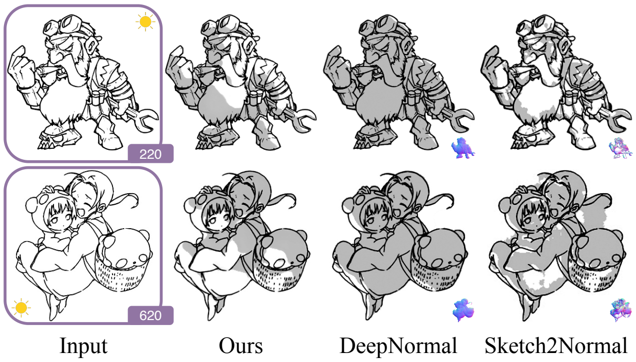


Figure 4: Comparisons with previous works DeepNormal [14] and Sketch2Normal [32] with lighting depth "2" - in the plane (side lighting). The little sun denotes the lighting direction.

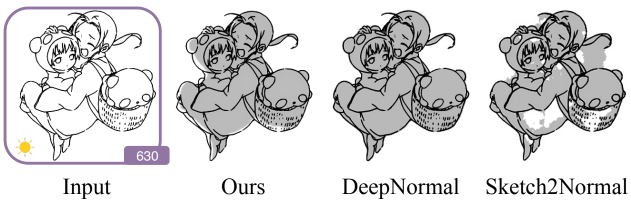


Figure 5: Comparison with DeepNormal [14] and Sketch2Normal [32] when the light's depth is "3" - behind the plane (back lighting). Our approach demonstrates rim lighting.

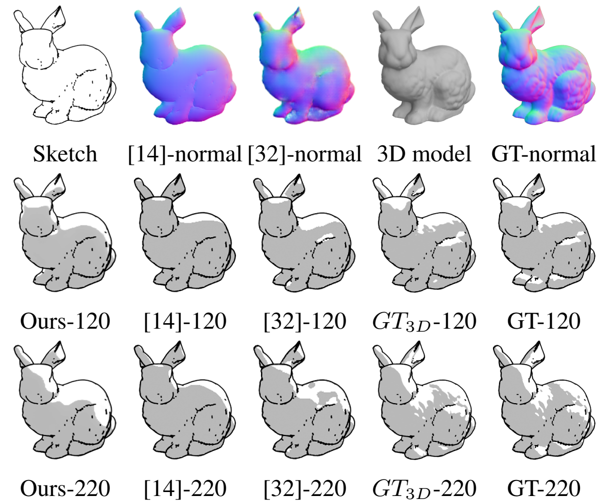


Figure 6: Comparisons between ground truth (GT), our approach, DeepNormal [14], and Sketch2Normal [32] rendered with a 3D bunny, with lighting depth "2". "120": top, side lighting. "220": upper right, side lighting. "GT_{3D}": rendered from commercial 3D software. "GT": rendered from its normal map.

method and DeepNormal are predicted from 320×320 inputs and Sketch2Normal is predicted from 256×256 inputs. Since DeepNormal claims their results are consistent in various size of input, and Sketch2Normal experiment in 256×256 inputs.

As shown in Figure 3, 4, 6, 5, 7, our work performs favorably. For example, on the two-people and multiple-people line drawings (Figure 3 second row), our work is able to shadow each character, however, DeepNormal and Sketch2Normal treat multiple people as one object. Notably, our work is superior in generating highly detailed shadows, such as in girl's hair and skirt. In terms of the complexity of sketch, though our training datasets have a moderate level of detail, our network performs well on complex sketches as shown in Figure 3. We also perform well beyond the object's boundary without requiring a mask.

Moreover, our work produces more precise details when the light source changes depth. As we can see in Figure 4, the shadows from DeepNormal [14] cover almost the entire image, so that it seems as though the light is behind the object. However, in these images, the light source is in the same plane as the object, resulting in side lighting. In Figure 6, we explain why DeepNormal [14] underperforms when the light is in the object's plane by comparing with a 3D test model. In particular, using our technique the shadows on the bunny's head and leg are closer to the ground truth and demonstrate self-shadowing. As highlighted in Figure 4, DeepNormal's normal maps have low variance due to multiple average of 256×256 tiles (refer to section 3.4 of DeepNormal). This low variance results in front lighting appearing to be side lighting and side lighting appearing to be back lighting. Some images generated by Sketch2Normal have some artifacts because the predicted normal maps have some blank areas. Because it is trained on simple sketches, Sketch2Normal struggles with complex sketches. Finally, we note that our approach produces artistic rim highlights from back lighting. Please refer to the supplementary material for the normal maps in Figure 3 and more comparison figures.

Our architecture also performs favorable when qualitatively compared to Pix2pix and U-net trained on our dataset (Figure 7). Generally, U-net generates inaccurate soft shadows that are far from our goal of binary shadows. Pix2pix generates shadows far outside the object's boundary and ignores the geometric information in the sketch. In our early research, we used a residual block autoencoder with skip connections, which generated soft shadows. To achieve our goal of binary shadows, we added a discriminator and adopted a deeper *RenderNet*. If the artist desires soft shadows, the intermediate output s_2 can be used.

4.3. Artistic Control

Though our network is trained with a discrete set of 26 lighting directions, the lighting direction is inputted to the network using floating point values in $[-1, 1]^3$, allowing for the generation of shadows from arbitrary light locations. In-

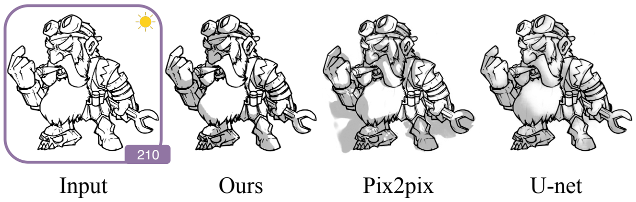


Figure 7: Comparison with Pix2pix [16] and U-net [28] architectures trained on our dataset. Light depth is "1".

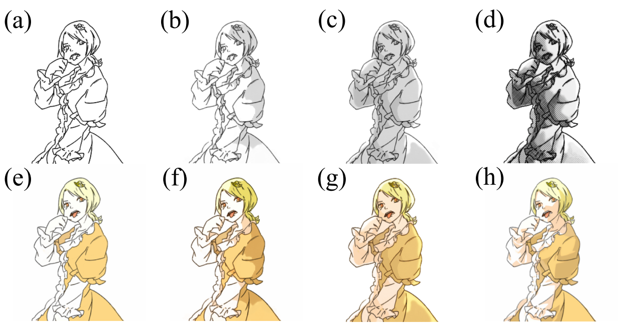


Figure 8: Combining our shadows with color. (a) Input sketch. (b) Our shadow with lighting direction "710". (c) Our shadows in complex lighting conditions created by compositing shadows from "001", "730", and "210". (d) Our shadows composited from lighting directions "001", "210", "220" with dots and soft shadow to produce a manga style. (e) Colorized sketch with commercial software. (f) Composite of (e) and (b). (g) Composite of (e) and (c). (h) Original artist's image. ©nico-opendata

Methods	GT	Ours	[14]	[32]	[16]	[28]
Turing	68%	**69%**	51%	11%	23%	19%
Scores	6.37	**6.70**	5.78	3.35	3.77	3.06
Methods	GT	Ours	[14]	[32]	[16]	[28]
Turing	**70%**	65%	45%	10%	25%	17%
Scores	6.50	**6.66**	5.69	3.44	3.91	3.03

Table 1: Results of user study comparing Ground truth (GT) in our datasets, Ours, DeepNormal [14], Sketch2Normal [32], Pix2pix [16] baseline and U-net [28] baseline. First row: percentage that pass the Turing test. Second row: average scores. 9 is the best score. Top: total results. Bottom: results of people with drawing experience.

tuitively, our network learns a continuous representation of lighting direction from the discrete set of examples. Furthermore, when a series of light locations are chosen the shadows move smoothly over the scene as in time-lapse video footage. Please refer to the supplementary material for gifs demonstrating moving shadows.

Although the final output of our network is binary shadows, if an artist desires soft shadows, the intermediate output, s_2, can be used, as shown in Figure 2.

Our work is complementary to prior work on automatic colorization of sketches [39, 19, 41, 5, 4]. Figure 8 demonstrates that our shadows can be combined with these colorization approaches. While most prior work on colorization combines shading and shadowing effects, it would be interesting to separate these effects into indpendent image layers for further artistic editing.

4.4. User Study

To evaluate our approach we conducted a small user study. We generated shadows using six different techniques and asked users to evaluate the results. We train the users with some samples from our dataset at the beginning. Our user study had two stages: a "Turing" test that asked the simple question "*Do you think this shadow was drawn by a human? Yes or no?*" and another stage where the user is shown an image and asked to rate the quality of the shadow with the prompt "*Under this lighting direction, evaluate the appearance of this shadow*" on a Likert scale from 1 to 9 (9 being best). In each stage the user was shown 36 images generated from six input sketches and each of six shadow generation methods: ground truth shadows created by artists, Ours, DeepNormal [14], Sketch2Normal [32], Pix2pix [16], and U-net [28]. For the synthetic shadows, the lighting directions were chosen randomly (Ours exclude the directions in ground truth), with the restriction that we did not use back lighting. We only used front lighting for DeepNormal for the reasons described in Section 4.2. For the quality rating, lighting directions were described with text, e.g. "upper right, front lighting." For the Turing test, no lighting directions were given. Users were shown one image at a time, but could use the "back" and "forward" buttons.

Users received a brief training that displayed 15 ground truth shadowed sketches from our dataset and highlighted the differences between front lighting and side lighting. We also asked the users to rate their drawing experience as "professional", "average", "beginner" or "0 experience". We distributed the survey online and received 60 results. Forty participants had drawing experience: 13 were professional artists, 11 were average level, and 16 were beginners. The results are shown in Table 1. Our approach performs favorably, almost matching the ground truth shadows created by artists. We ran a one-way ANOVA to analyze the Likert scores. The results confirmed that our results were quantitatively similar to ground truth ($p = 0.24$) and better than the other methods ($p < 0.05$ for all of the comparisons). Please refer to the supplementary material for more details of the statistical significance report of our user study.

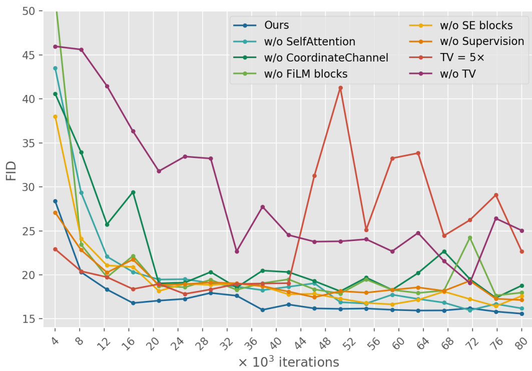
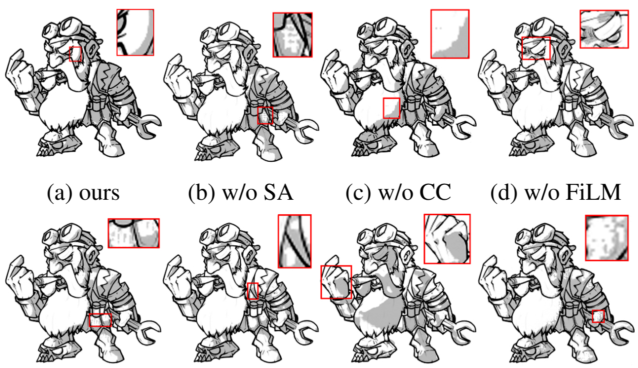


Figure 9: FID scores of ours and ablation studies. Our model's line is on the most bottom.



(a) ours (b) w/o SA (c) w/o CC (d) w/o FiLM

(e) w/o SE (f) w/o s_1, s_2 (g) $TV = 5\times$ (h) w/o TV

Figure 10: Ablation studies. (b) removing the Self-attention (SA) layers, (c) removing the Coordinate Channel (CC), (d) removing the FiLM residual blocks, (e) removing the SE blocks, (f) removing the two deep supervised outputs (s_1, s_2), (g) increasing the TV loss weights to e^{-5}, (h) removing the total variant (TV) regularizer.

4.5. Ablation Study

We performed seven ablation studies as shown in Figures 10 and 9. For quantitative comparison, we calculated the Fréchet Inception Distance (FID) [11] per 4000 iterations of our work and the ablation studies using the entire dataset. Figure 9 shows that our work has the lowest and most stable FID. This demonstrate that each feature we propose is essential and that the total variation regularizer was critically important.

Figure 10 qualitatively demonstrates that without the elements we propose, the networks performance is degraded: boundaries become aliased and artifacts appear in shadows. Among all ablation studies, "w/o Self-attention" has the least influence, as the shown Figure 10 (b) and the FID in Figure 9. Setting the coefficient of the total variation regularizer $5\times$ larger or removing the regularizer has the most influence on the overall performance and ruins the smoothness of shadow. The corresponding FID also highlight the importance of the total variant regularizer.

In Figure 10, all of the images use the same lighting direction "810". Generally, when the Self-attention layers are removed, the network performs poorly with details and there are tiny artifacts within the shadow block; without the Coordinate Channel or FiLM residual block, the output will have unrealistic shadow boundaries and shadows outside the object's boundary; without SE blocks, there will be shadow "acne" and the overall appearance looks messy; without the two deep supervised outputs ($\lambda_1 = .4$, $\lambda_2 = .9$, $\lambda_3 = \lambda_4 = 0$), the output will have dot artifacts in a grid pattern and lower accuracy; if the network has $5\times$ higher weight for the TV regularizer or is missing the TV regularizer, the network will converge too fast and trap in a local minimum.

5. Future Work

The network performance is not invariant on different sizes of input images. Mostly the 320×320 inputs have the best performance, because our network is trained on 320×320 size inputs. 480×480 input images also have good performance. Though we almost match the ground truth in user study, our generated shadows are not so much detailed as ground truth, especially on hard surface object (*e.g.* desk, laptop). Also, if inputting a local part of the line drawing, the network is not able to reason the correct shadows. As future work, we will develop a network that can output various image sizes to meet the high resolution requirements of painting.

6. Conclusion

Our conditional Generative Adversarial Network learns a non-photorealistic renderer that can automatically generate shadows from hand-drawn sketches. We are the first to attempt to directly generate shadows from sketches through deep learning. Our results compare favorably to prior art that renders normal maps from sketches on both simple and sophisticated images. We also demonstrate that our network architecture can "understand" the 3D spatial relationships implied by 2D line drawings well enough to generate detailed and accurate shadows.

Acknowledgements: The authors wish to thank Yuchen Ma and Kejun Liu for annotating the dataset and the reviewers, Tiantian Xie, and Lvmin Zhang for many suggestions.

References

[1] Rudolf Arnheim. *Art and visual perception: A psychology of the creative eye*. Univ of California Press, 1965.

[2] Zezhou Cheng, Qingxiong Yang, and Bin Sheng. Deep colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 415–423, 2015.

[3] Christopher DeCoro, Forrester Cole, Adam Finkelstein, and Szymon Rusinkiewicz. Stylized shadows. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 77–83. ACM, 2007.

[4] Kevin Frans. Outline colorization through tandem adversarial networks. *arXiv preprint arXiv:1704.08834*, 2017.

[5] Chie Furusawa, Kazuyuki Hiroshiba, Keisuke Ogaki, and Yuri Odagiri. Comicolorization: semi-automatic manga colorization. In *SIGGRAPH Asia 2017 Technical Briefs*, page 12. ACM, 2017.

[6] Zhengyan Gao, Taizan Yonetsuji, Tatsuya Takamura, Toru Matsuoka, and Jason Naradowsky. Automatic illumination effects for 2d characters. In *NIPS Workshop on Machine Learning for Creativity and Design*, 2018.

[7] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[8] Todd Goodwin, Ian Vollick, and Aaron Hertzmann. Isophote distance: a shading approach to artistic stroke thickness. In *Proceedings of the 5th international symposium on Non-photorealistic animation and rendering*, pages 53–62. ACM, 2007.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

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

[12] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[13] Matis Hudon, Rafael Pagés, Mairéad Grogan, Jan Ondřej, and Aljoša Smolić. 2d shading for cel animation. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, page 15. ACM, 2018.

[14] Matis Hudon, Rafael Pagés, Mairéad Grogan, and Aljosa Smolić. Deep normal estimation for automatic shading of hand-drawn characters. In *ECCV Workshops*, 2018.

[15] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification. *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)*, 35(4):110:1–110:11, 2016.

[16] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5967–5976. IEEE, 2017.

[17] Scott F Johnston. Lumo: illumination for cel animation. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, pages 45–ff, 2002.

[18] Evangelos Kalogerakis, Derek Nowrouzezahrai, Simon Breslav, and Aaron Hertzmann. Learning hatching for pen-and-ink illustration of surfaces. *ACM Transactions on Graphics (TOG)*, 31(1):1, 2012.

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

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Chengze Li, Xueting Liu, and Tien-Tsin Wong. Deep extraction of manga structural lines. *ACM Transactions on Graphics (TOG)*, 36(4):117, 2017.

[22] Yijun Li, Chen Fang, Aaron Hertzmann, Eli Shechtman, and Ming-Hsuan Yang. Im2pencil: Controllable pencil illustration from photographs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1525–1534, 2019.

[23] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*, pages 9628–9639, 2018.

[24] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[25] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[26] Lena Petrović, Brian Fujito, Lance Williams, and Adam Finkelstein. Shadows for cel animation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 511–516. ACM Press/Addison-Wesley Publishing Co., 2000.

[27] Python. Cairosvg, 2019. `https://cairosvg.org/`.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[29] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. Mastering sketching: adversarial augmentation for structured prediction. *ACM Transactions on Graphics (TOG)*, 37(1):11, 2018.

[30] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. Real-time data-driven interactive rough sketch inking. *ACM Transactions on Graphics (TOG)*, 37(4):98, 2018.

[31] Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. Learning to simplify: fully convolutional networks for rough sketch cleanup. *ACM Transactions on Graphics (TOG)*, 35(4):121, 2016.

[32] Wanchao Su, Dong Du, Xin Yang, Shizhe Zhou, and Hongbo Fu. Interactive sketch-based normal map generation with

deep neural networks. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 1(1):22, 2018.

[33] Tiancheng Sun, Jonathan T Barron, Yun-Ta Tsai, Zexiang Xu, Xueming Yu, Graham Fyffe, Christoph Rhemann, Jay Busch, Paul Debevec, and Ravi Ramamoorthi. Single image portrait relighting. *ACM Transactions on Graphics (TOG)*, 38(4):79, 2019.

[34] Daniel Sỳkora, Ladislav Kavan, Martin Čadík, Ondřej Jamriška, Alec Jacobson, Brian Whited, Maryann Simmons, and Olga Sorkine-Hornung. Ink-and-ray: Bas-relief meshes for adding global illumination effects to hand-drawn characters. *ACM Transactions on Graphics (TOG)*, 33(2):16, 2014.

[35] Hideki Todo, Ken Anjyo, and Shunichi Yokoyama. Lit-sphere extension for artistic rendering. *The Visual Computer*, 29(6-8):473–480, 2013.

[36] Hideki Todo, Ken-ichi Anjyo, William Baxter, and Takeo Igarashi. Locally controllable stylized shading. *ACM Trans. Graph.*, 26(3):17, 2007.

[37] Wikipedia. Inker, 2019. `https://en.wikipedia.org/wiki/Inker`.

[38] Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, and Ravi Ramamoorthi. Deep image-based relighting from optimal sparse samples. *ACM Transactions on Graphics (TOG)*, 37(4):126, 2018.

[39] Taizan Yonetsuji. Paintschainer, 2017. https://paintschainer.preferred.tech/.

[40] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[41] Lvmin Zhang, Chengze Li, Tien-Tsin Wong, Yi Ji, and Chunping Liu. Two-stage sketch colorization. In *SIGGRAPH Asia 2018 Technical Papers*, page 261. ACM, 2018.

[42] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European Conference on Computer Vision*, pages 649–666. Springer, 2016.

[43] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics (TOG)*, 9(4), 2017.

[44] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

A. Lighting Directions

We found '810' numbering scheme to be more intuitive for the users than the other two methods ($[-1, 0, 1]^3$ or an integer between 1 and 26). Therefore, we use '810' scheme as the user inputs, then transfer the '810' scheme (first column of Table 2) to the $[-1, 0, 1]^3$ scheme (third column of Table 2) in programming.

Label	Direction	Position
001	rear center	[0,0,1]
002	front center	[0,0,-1]
110	center top, front lighting	[0,1,-1]
120	center top, side lighting	[0,1,0]
130	center top, back lighting	[0,1,1]
210	upper right, front lighting	[1,1,-1]
220	upper right, side lighting	[1,1,0]
230	upper right, back lighting	[1,1,1]
310	center right, front lighting	[1,0,-1]
320	center right, side lighting	[1,0,0]
330	center right, back lighting	[1,0,1]
410	lower right, front lighting	[1,-1,-1]
420	lower right, side lighting	[1,-1,0]
430	lower right, back lighting	[1,-1,1]
510	bottom, front lighting	[0,-1,-1]
520	bottom, side lighting	[0,-1,0]
530	bottom, back lighting	[0,-1,1]
610	lower left, front lighting	[-1,-1,-1]
620	lower left, side lighting	[-1,-1,0]
630	lower left, back lighting	[-1,-1,1]
710	center left, front lighting	[-1,0,-1]
720	center left, side lighting	[-1,0,0]
730	center left, back lighting	[-1,0,1]
810	upper left, front lighting	[-1,1,-1]
820	upper left, side lighting	[-1,1,0]
830	upper left, back lighting	[-1,1,1]

Table 2: A lookup table of our 26 lighting direction labels, the actual lighting directions, and $[-1, 0, 1]^3$ style positions in programming.

B. Pre-processing:

The pre-processing is a light neural network modified by smart inker [30]. We re-trained the network with synthetic data (0.2–2 px, cairosvg standard, various darkness). This pre-processing network was sufficiently robust for the Japanese and Disney style images we have tested, which had line widths in the range 1–6px.

Figures 23, 24, 25, and 26 give some indication of how this pre-processing network performs "in the wild on a wide range of line styles."

C. Network Architecture

More details of the network architecture are in Table 3, 4, 5, 6, 7, 8, 9 and Figure 11, 12. Please refer to the main body of our paper for the network architecture figure.

'ResiBlock': Residual Blocks. 'DownResiBlock': Downscale Residual Blocks. 'UpResiBlock': Upscale Residual Blocks. 'ShapeNet': the encoder of Generator. 'RenderNet': the decoder of Generator.

The first stage of RenderNet does not contain UpResiBlock nor SE block, and has four ResiBlocks. The second to the last stage of RenderNet does not contain SelfAttention. The last stage of RenderNet consists of just three ResiBlocks.

We use a fully connected layer (Table 3) to embed the $[-1, 1]^3$ lighting positions. We repeatedly input the embedded lighting position into each stage of RenderNet where a FiLMResiBlock exists. However, we only input the lighting direction once without embedding at the beginning of Discriminator.

The inputs of the Generator are the line drawing, pure shadow (ground truth), and the lighting direction. The outputs of the Generator are the final output (pure binary shadow), s_1 and s_2. The inputs of the Discriminator are the composition of line drawing and pure shadow (ground truth and the final output of Generator), and the lighting direction.

Layer	Filter	Output Size
Linear	128	128
Tanh()	-	128

Table 3: **Light Position embedding**

F(x)
BatchNorm
LeakyReLU()
Conv2D(1×1)
BatchNorm
LeakyRuLU()
Conv2D(3×3)
BatchNorm
LeakyReLU()
Conv2D(1×1)
Shortcut Branch
Conv2D(1×1)

Table 4: **ResiBlock**

F(x)
BatchNorm
LeakyReLU()
Conv2D(1×1)
BatchNorm
LeakyRuLU()
Conv2D(3×3, strides=2)
BatchNorm
LeakyReLU()
Conv2D(1×1)
Shortcut Branch
Conv2D(1×1, strides=2)

Table 5: **DownResiBlock**

F(x)
BatchNorm
LeakyReLU()
Conv2D(1×1)
BatchNorm
LeakyRuLU()
SubPixelConv2D(3×3, strides=2)
BatchNorm
LeakyReLU()
Conv2D(1×1)
Shortcut Branch
SubPixelConv2D(1×1, strides=2)

Table 6: **UpResiBlock**. Dropout(0.1) was added after the residual addition.

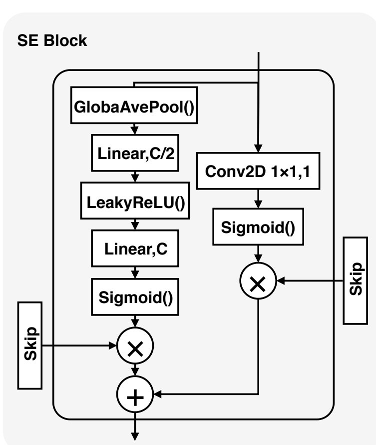


Figure 11: **SE (SE Block)**. 'C' is filter size.

Layer	Filter	Output Size
Concat(Coord)	-	$3 \times 320 \times 320$
ResiBlock	8, 8, 32	$32 \times 320 \times 320$
ResiBlock	8, 8, 32	$32 \times 320 \times 320$
DownResiBlock	16, 16, 64	$64 \times 160 \times 160$
ResiBlock	16, 16, 64	$64 \times 160 \times 160$
ResiBlock	16, 16, 64	$64 \times 160 \times 160$
DownResiBlock	32, 32, 128	$128 \times 80 \times 80$
ResiBlock	32, 32, 128	$128 \times 80 \times 80$
ResiBlock	32, 32, 128	$128 \times 80 \times 80$
DownResiBlock	64, 64, 256	$256 \times 40 \times 40$
ResiBlock	64, 64, 256	$256 \times 40 \times 40$
ResiBlock	64, 64, 256	$256 \times 40 \times 40$
DownResiBlock	64, 64, 256	$256 \times 20 \times 20$
ResiBlock	64, 64, 256	$256 \times 20 \times 20$
ResiBlock	64, 64, 256	$256 \times 20 \times 20$
DownResiBlock	128, 128, 512	$512 \times 10 \times 10$
ResiBlock	128, 128, 512	$512 \times 10 \times 10$
ResiBlock	128, 128, 512	$512 \times 10 \times 10$

Table 7: **ShapeNet**

Layer	Filter	Output Size
Concat(Pos, Coord)	-	$6 \times 320 \times 320$
DownResiBlock	8, 8, 32	$32 \times 160 \times 160$
ResiBlock	8, 8, 32	$32 \times 160 \times 160$
DownResiBlock	16, 16, 64	$64 \times 80 \times 80$
ResiBlock	16, 16, 64	$64 \times 80 \times 80$
DownResiBlock	32, 32, 128	$128 \times 40 \times 40$
ResiBlock	32, 32, 128	$128 \times 40 \times 40$
SelfAttention	-	$128 \times 40 \times 40$
DownResiBlock	64, 64, 256	$256 \times 20 \times 20$
ResiBlock	64, 64, 256	$256 \times 20 \times 20$
SelfAttention	-	$256 \times 20 \times 20$
DownResiBlock	128, 128, 512	$512 \times 10 \times 10$
ResiBlock	128, 128, 512	$512 \times 10 \times 10$
GlobalAvgPool()	-	512
Dropout(0.3)	-	512
Linear	256	256
Linear	1	1
Sigmoid()	-	1

Table 8: **Discriminator**

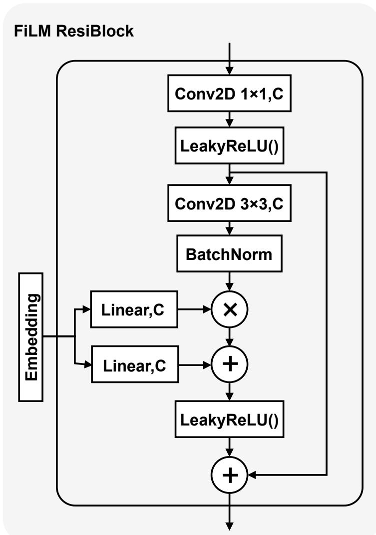


Figure 12: **FiLMResiBlock**[25]. 'C' is filter size.

D. Compositing sketches and shadows

Our result images (I) are composited by a simple weighted sum of the output shadow (S) and original line drawing (L)

$$I = 0.2S + 0.8L, \tag{5}$$

where all images are grayscale in $[0, 1]$.

In our training process, both line drawings and shadows are processed. The line drawings are inverted, $L' = 1 - L$, to achieve white lines on a black background. The shadow images are inverted, then scaled and shifted to the interval $[-1, 1]$, $S' = (1 - S) \times 2 - 1$. The inverse transform is applied to output from the Generator before compositing as described above.

Additionally, simply concatenating the line drawing and shadow for input to the Discriminator produced poor results; instead we composite these images with another weighted sum,

$$I' = L' + 0.25(S' + 1). \tag{6}$$

This compositing is applied to both ground truth and Generator shadows.

E. More Results

Figure 13, 14 : zoom in the Fig3 in our main body of paper. Figure 15, 16, 17, 18, 19 : more comparisons with related work. Figure 20, 21, 22 : our results in more lighting directions. Figure 23, 24, 25: examples of our shadowing system applied to artistic line drawings. Figure 26 : our results with and without pre-processing, and the robustness of our results in the wild. Figure 27: generalization ability. Figure 28 : Failure cases.

Layer	Filter	Output Size
Concat(Coord)	-	$514 \times 10 \times 10$
FiLMResiBlock(x, e)	512	$512 \times 10 \times 10$
ResiBlock	128, 128, 512	$512 \times 10 \times 10$
ResiBlock	128, 128, 512	$512 \times 10 \times 10$
ResiBlock	128, 128, 512	$512 \times 10 \times 10$
ResiBlock	128, 128, 512	$512 \times 10 \times 10$
SelfAttention	-	$512 \times 10 \times 10$
UpResiBlock	64, 64, 256	$256 \times 20 \times 20$
Concat(SE(x, s), Coord)	-	$514 \times 20 \times 20$
FiLMResiBlock(x, e)	256	$256 \times 20 \times 20$
ResiBlock	64, 64, 256	$256 \times 20 \times 20$
ResiBlock	64, 64, 256	$256 \times 20 \times 20$
SelfAttention	-	$256 \times 20 \times 20$
UpResiBlock	64, 64, 256	$256 \times 40 \times 40$
Concat(SE(x, s), Coord)	-	$514 \times 40 \times 40$
FiLMResiBlock(x, e)	256	$256 \times 40 \times 40$
ResiBlock	64, 64, 256	$256 \times 40 \times 40$
ResiBlock	64, 64, 256	$256 \times 40 \times 40$
SelfAttention	-	$256 \times 40 \times 40$
UpResiBlock	32, 32, 128	$128 \times 80 \times 80$
Concat(SE(x, s), Coord)	-	$258 \times 80 \times 80$
FiLMResiBlock(x, e)	128	$128 \times 80 \times 80$
ResiBlock	32, 32, 128	$128 \times 80 \times 80$
ResiBlock	32, 32, 128	$128 \times 80 \times 80$
SelfAttention	-	$128 \times 80 \times 80$
UpResiBlock	16, 16, 64	$64 \times 160 \times 160$
Concat(SE(x, s), Coord)	-	$130 \times 160 \times 160$
FiLMResiBlock(x, e)	64	$64 \times 160 \times 160$
ResiBlock	16, 16, 64	$64 \times 160 \times 160$
ResiBlock	16, 16, 64	$64 \times 160 \times 160$
SelfAttention	-	$64 \times 160 \times 160$
UpResiBlock	8, 8, 32	$32 \times 320 \times 320$
Concat(SE(x, s), Coord)	-	$66 \times 320 \times 320$
FiLMResiBlock(x, e)	32	$32 \times 320 \times 320$
ResiBlock	8, 8, 32	$32 \times 320 \times 320$
ResiBlock	8, 8, 32	$32 \times 320 \times 320$
ResiBlock	4, 4, 16	$16 \times 320 \times 320$
ResiBlock	4, 4, 16	$16 \times 320 \times 320$
ResiBlock	4, 4, 16	$16 \times 320 \times 320$
Conv2D(1×1)	1	$1 \times 320 \times 320$
Tanh()	-	$1 \times 320 \times 320$

Table 9: **RenderNet**. 'e' is lighting position embedding. 's' is skip connection from ShapeNet.

Table 10 and 11 shows the statistically significance report of our user study in Likert scores. We deploy levene's test for the equality of variance, and Fisher's Least Significant Difference (LSD) for the analysis of variance.

	GT	Our	[14]	[32]	pix2pix	U-net
Mean	6.37	6.70	5.78	3.35	3.78	3.06
SD	1.55	1.33	1.27	1.30	1.31	2.11
Mean	6.50	6.66	5.69	3.44	3.91	3.03
SD	1.40	1.15	1.02	1.12	1.10	2.10

Table 10: Mean and standard deviation of user study (on Likert score). Top: total results. Bottom: results of people with drawing experience.

	MeanDiff	**T-value**	**P-value**	**Alpha**
Our / GT	0.33	1.18	0.24	0.05
[14] / GT	-0.59	-2.16	0.03	0.05
[14] / Our	-0.92	-3.34	$9.23E^{-4}$	0.05
[32] / GT	-3.02	-10.98	$2.41E^{-24}$	0.05
[32] / Our	-3.35	-12.16	$1.11E^{-28}$	0.05
[32] / [14]	-2.43	-8.82	$5.21E^{-17}$	0.05
pix2pix / GT	-2.60	-9.45	$4.83E^{-19}$	0.05
pix2pix / Our	-2.93	-10.63	$4.36E^{-23}$	0.05
pix2pix / [14]	-2.00	-7.29	$2.07E^{-12}$	0.05
pix2pix / [32]	0.42	1.53	0.126	0.05
U-net / GT	-3.31	-12.02	$3.76E^{-28}$	0.05
U-net / Our	-3.63	-13.20	$1.22E^{-32}$	0.05
U-net / [14]	-2.71	-9.87	$1.98E^{-20}$	0.05
U-net / [32]	-0.29	-1.04	0.30	0.05
U-net/pix2pix	-0.71	-2.57	0.01	0.05
Our / GT	0.16	0.52	0.60	0.05
[14] / GT	-0.82	-2.67	$8.07E^{-3}$	0.05
[14] / Our	-0.98	-3.19	$1.62E^{-3}$	0.05
[32] / GT	-3.06	-10.02	$6.79E^{-20}$	0.05
[32] / Our	-3.22	-10.54	$1.67E^{-21}$	0.05
[32] / [14]	-2.25	-7.34	$3.33E^{-12}$	0.05
pix2pix / GT	-2.60	-8.49	$2.35E^{-15}$	0.05
pix2pix / Our	-2.75	-9.01	$7.39E^{-17}$	0.05
pix2pix / [14]	-1.78	-5.82	$1.91E^{-8}$	0.05
pix2pix / [32]	0.47	1.53	0.128	0.05
U-net / GT	-3.48	-11.37	$3.87E^{-24}$	0.05
U-net / Our	-3.63	-11.89	$8.28E^{-26}$	0.05
U-net / [14]	-2.65	-8.70	$6.06E^{-16}$	0.05
U-net / [32]	-0.41	-1.35	0.178	0.05
U-net/pix2pix	-0.87	-2.88	$4.40E^{-3}$	0.05

Table 11: Statistical significance report of user study (on Likert score). Top: total results. Bottom: results of people with drawing experience.

F. Dataset Samples

Figure 29 shows $\{$sketch, light direction, shadow, mask$\}$ sample pairs from our dataset. Our dataset comprises 1,160 sketch/shadow pairs and includes a variety of lighting directions and subjects. Specifically, 372 front-lighting, 506 side-lighting, 111 back-lighting, 85 center-back, and 86 center-front. With regard to subjects there are 867 single-person, 56 multi-person, 177 body-part, and 60 mecha.

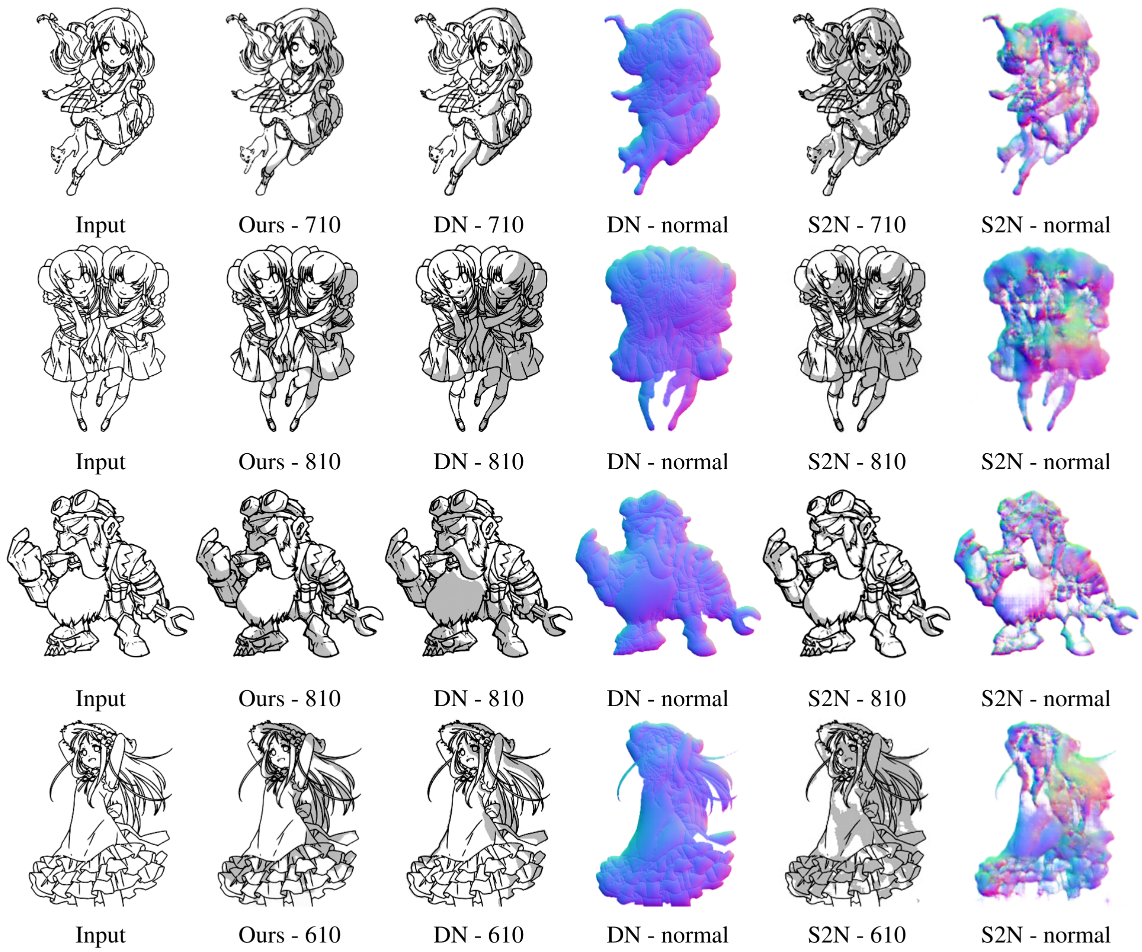


Figure 13: Comparisons with previous work DeepNormal (DN) [14] and Sketch2Normal (S2N) [32] in front lighting. (Part 1)

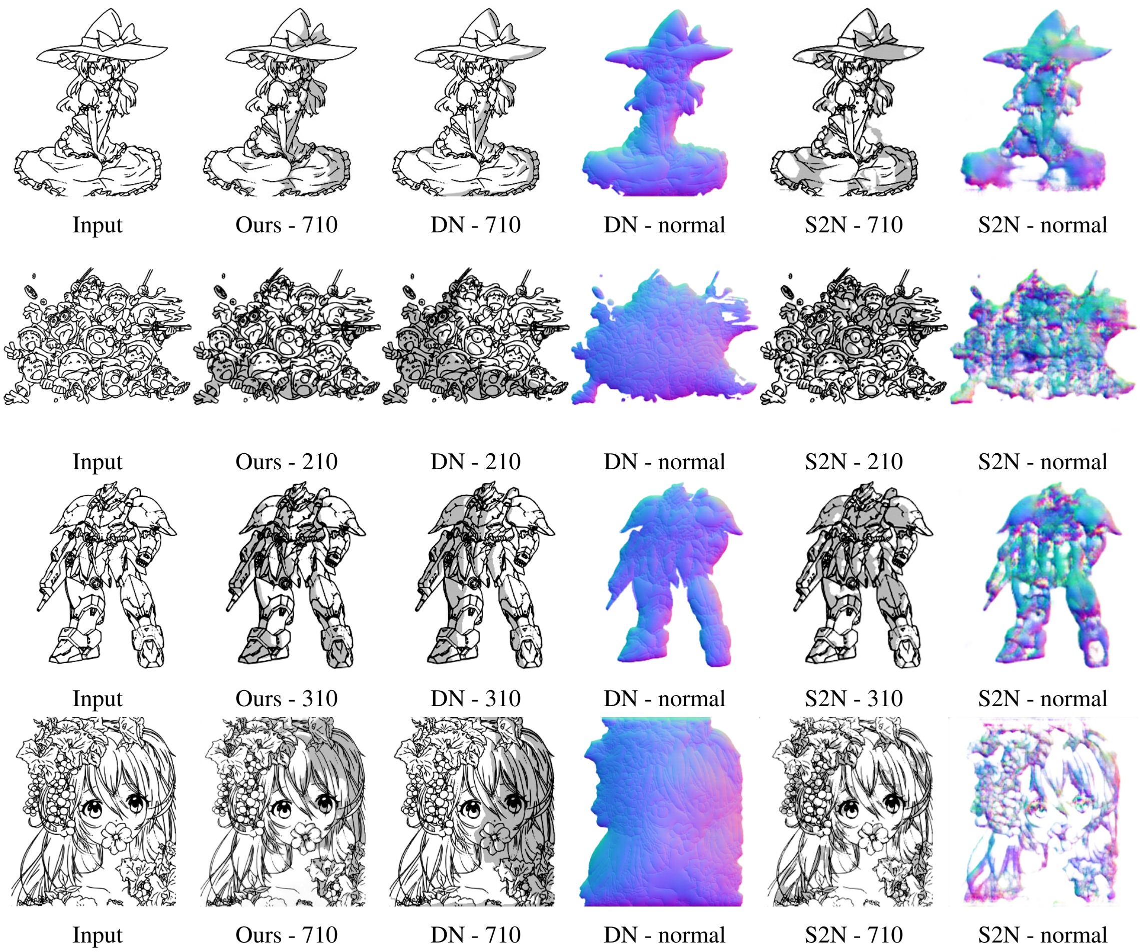


Figure 14: Comparisons with previous work DeepNormal (DN) [14] and Sketch2Normal (S2N) [32] in front lighting. (Part 2)

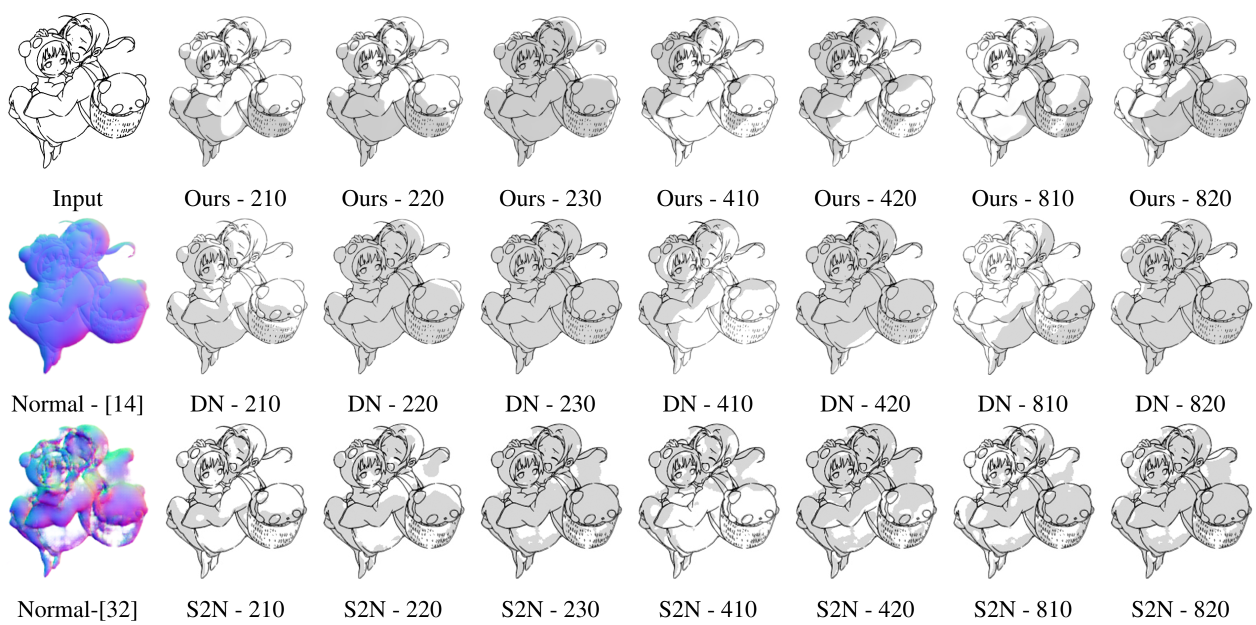


Figure 15: Comparisons with previous work DeepNormal (DN) [14] and Sketch2Normal (S2N) [32] when the light source changes depth. First row is ours. The second row is DeepNormal's. The third row is Sketch2Normal's.

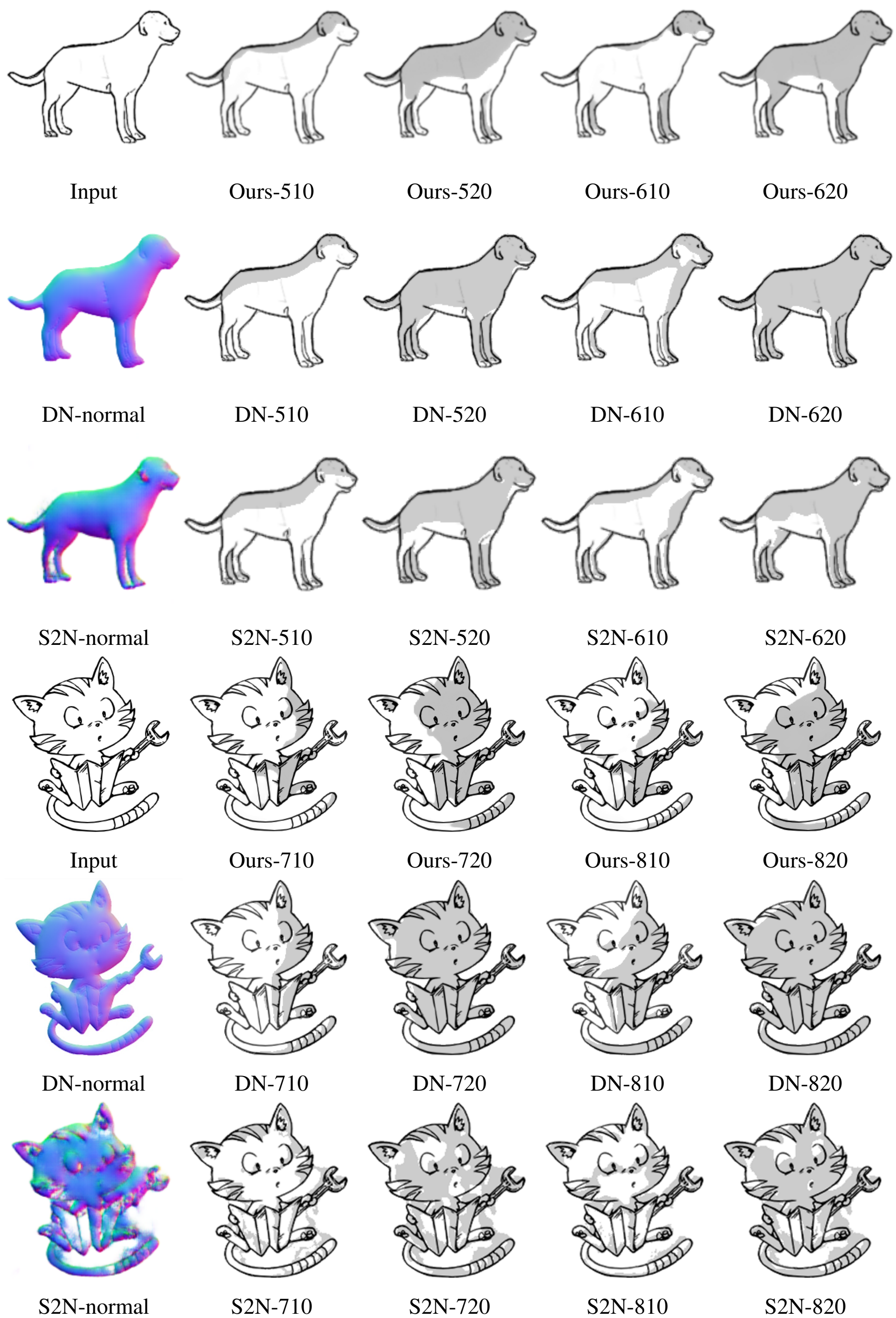


Figure 16: Comparisons with previous work DeepNormal (DN) [14] and Sketch2Normal (S2N) [32] using the line drawings from their papers. Dog image is from Sketch2Normal [32]. Cat image is from DeepNormal [14].

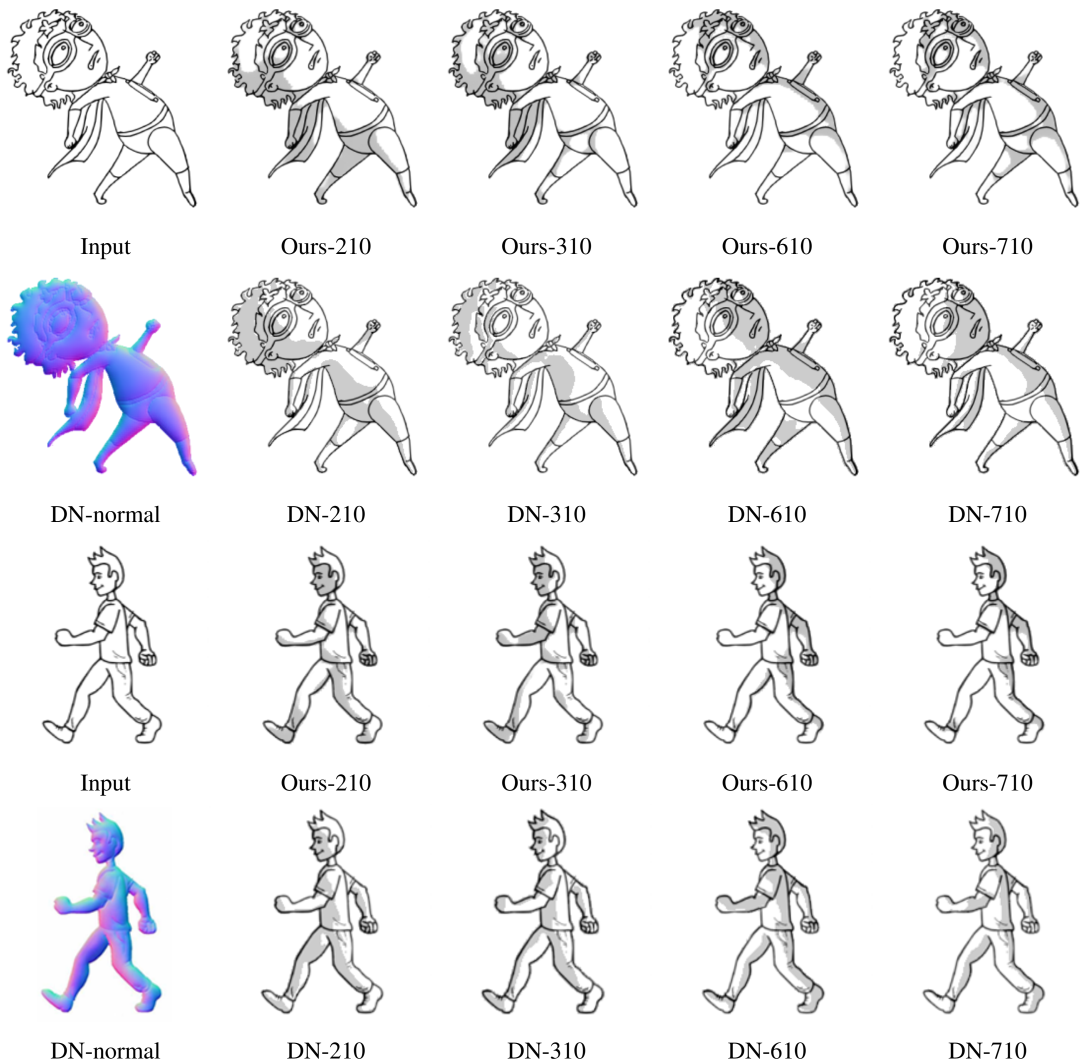
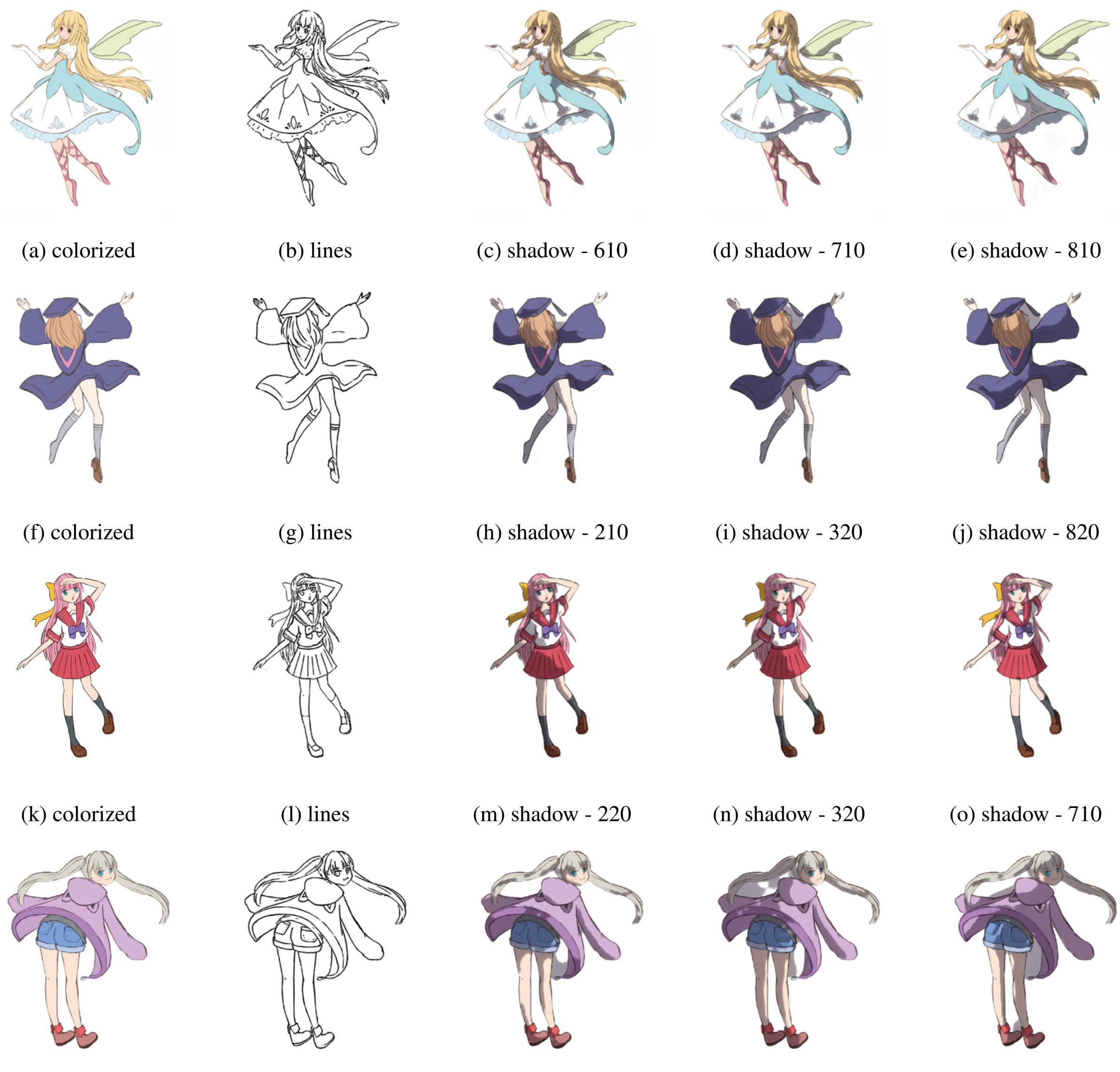


Figure 17: Comparisons with DeepNormal (DN) [14] using the line drawings and normal maps in the paper.



(a) colorized (b) lines (c) shadow - 610 (d) shadow - 710 (e) shadow - 810

(f) colorized (g) lines (h) shadow - 210 (i) shadow - 320 (j) shadow - 820

(k) colorized (l) lines (m) shadow - 220 (n) shadow - 320 (o) shadow - 710

(p) colorized (q) lines (r) shadow - 210 (s) shadow - 320 (t) shadow - 820

Figure 18: Our results using [6]'s images. [6] solves similar problems, inputting colorized images to predict the normal maps midway then generate the binary shadows. The colorized images (a), (f), (k), (p) are from [6]. (b), (g), (l), (q) are the lines that we subtract from the colorized images. Our system uses (b), (g), (l), (q) as the inputs to predict the pure shadows, then composite the shadows with the colorized images. For each line drawing, we show our results in three lighting directions. Please refer to [6] for the their results in similar lighting directions.

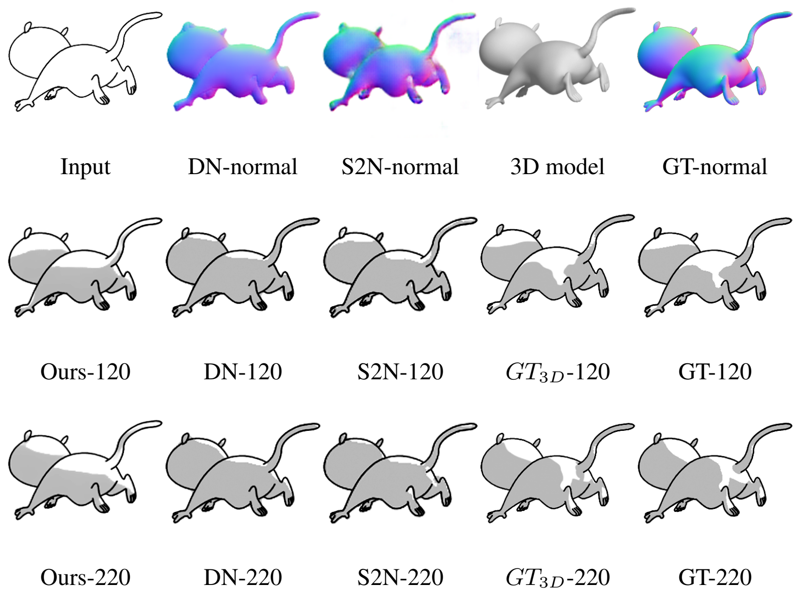


Figure 19: Comparisons on 3D test model [8] with DeepNormal (DN) [14], Sketch2Normal (S2N) [32] and Ground Truth (GT). GT_{3D}: rendered directly from 3D model with commercial software. GT: rendered from its normal map. All of the normal maps, including ground truth, are rendered with the same settings as the paper (use the renderer scripts provided by DeepNormal and threshold the continuous shadings at 0.5). Along with the bunny 3D test model in paper, our shadows are most close to the ground truth.

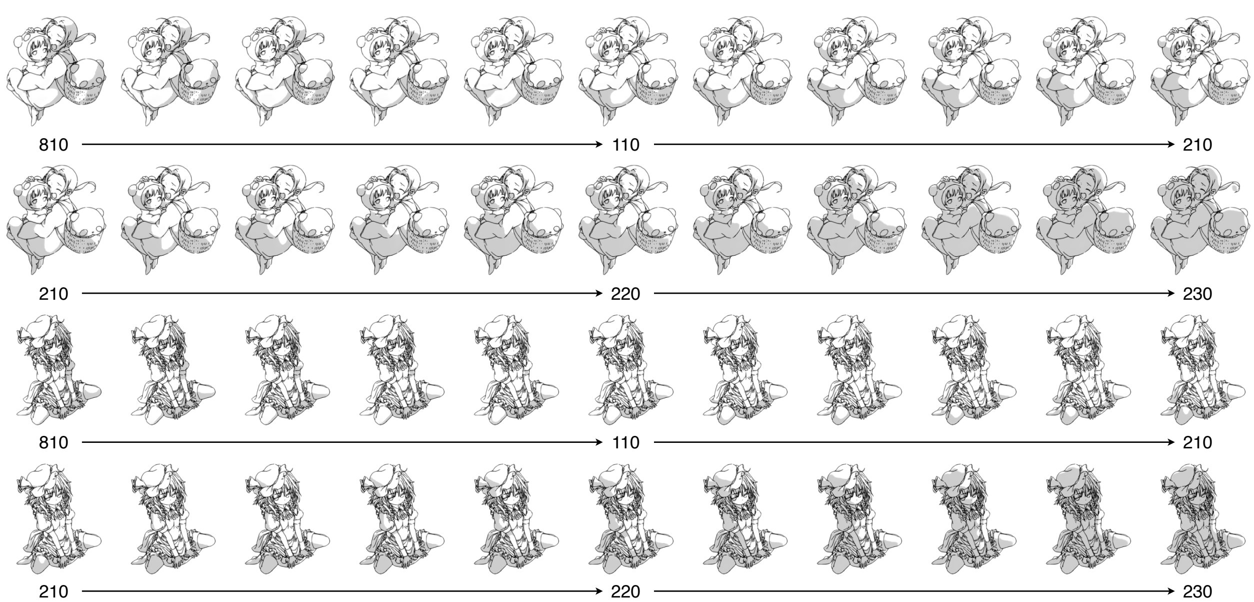


Figure 20: Examples of our continuous shadows between the discrete lighting source.

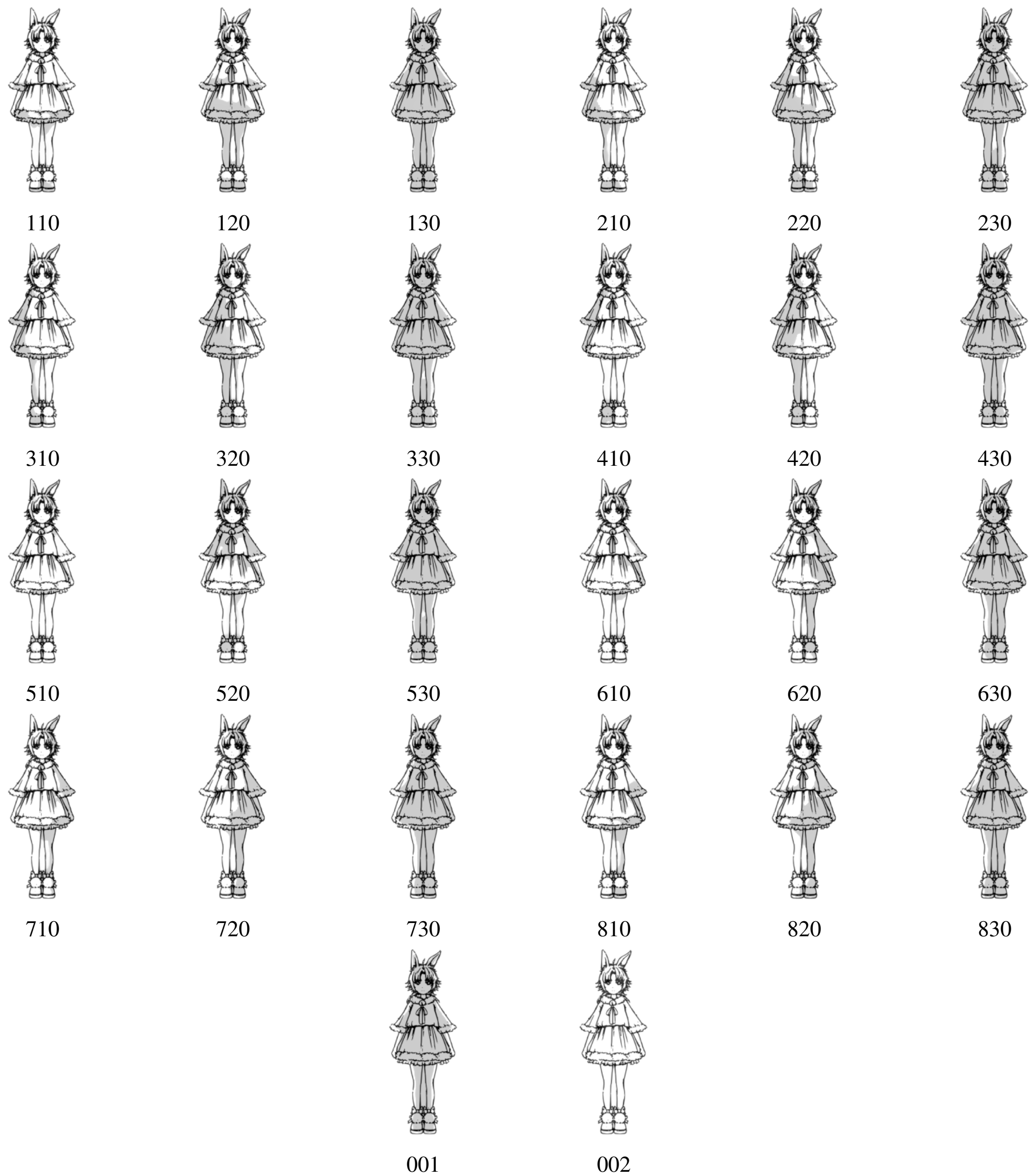


Figure 21: Evaluations of our work in all 26 directions.

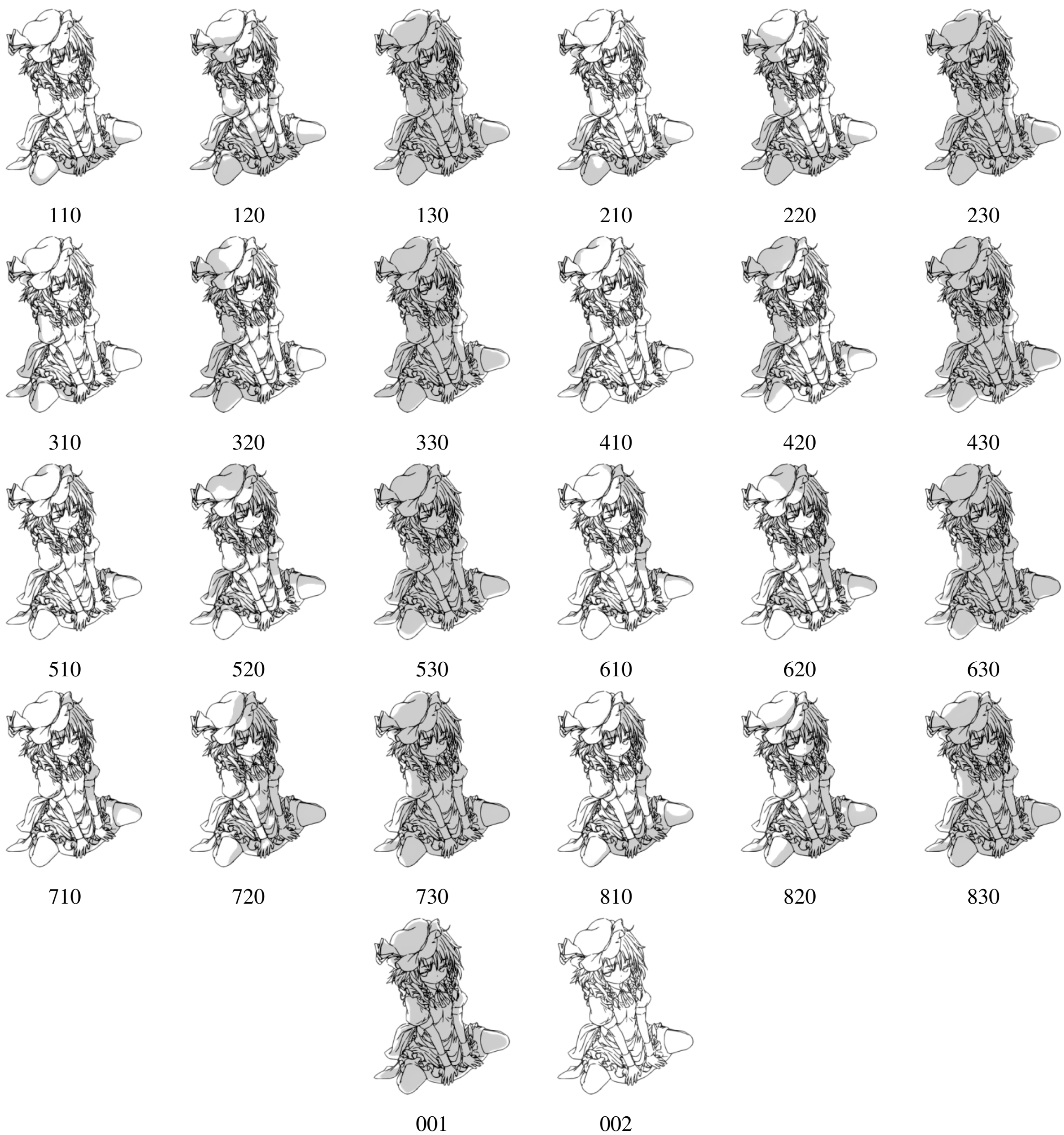


Figure 22: Evaluations of our work in all 26 directions.

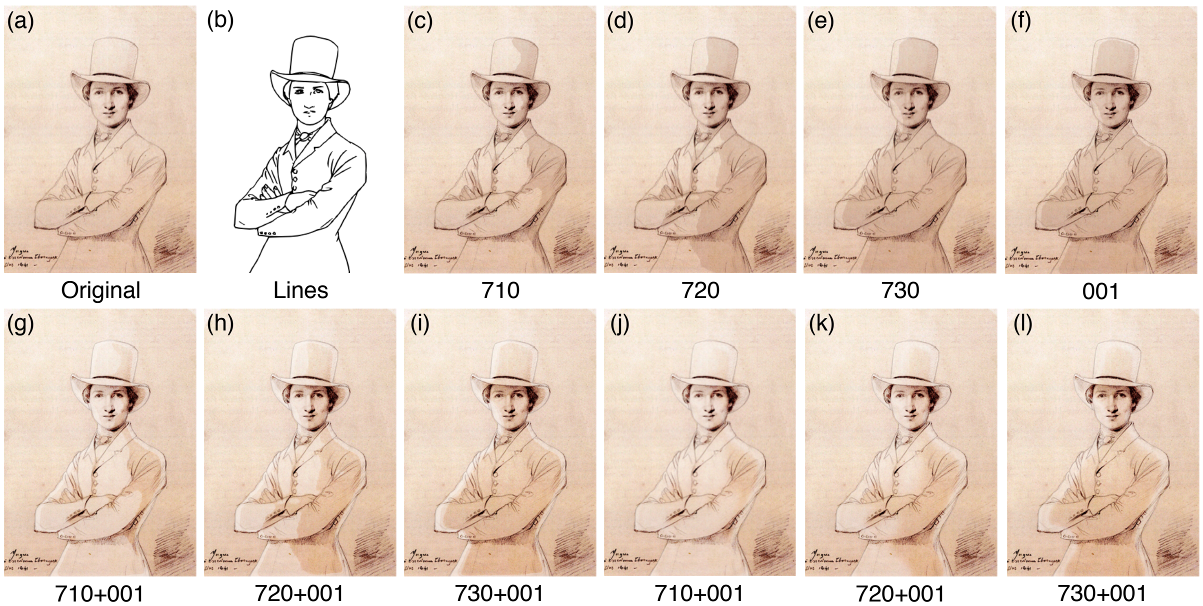


Figure 23: Examples of our shadowing system applying to artistic line drawing (Antoine Thomeguex, drawn by Jean Auguste Dominique Ingres. Public domain.). (a): original sketch. (b): extract lines from (a). (c)-(f): binary shadows in 710, 720, 730 and 001 lighting directions. (g)-(i): composites of binary shadows in dual lighting directions. (j)-(l): soft shadows in dual lighting directions. The results show that our shadowing system can give artists hints or a starting point to study shadows in different lighting sources.

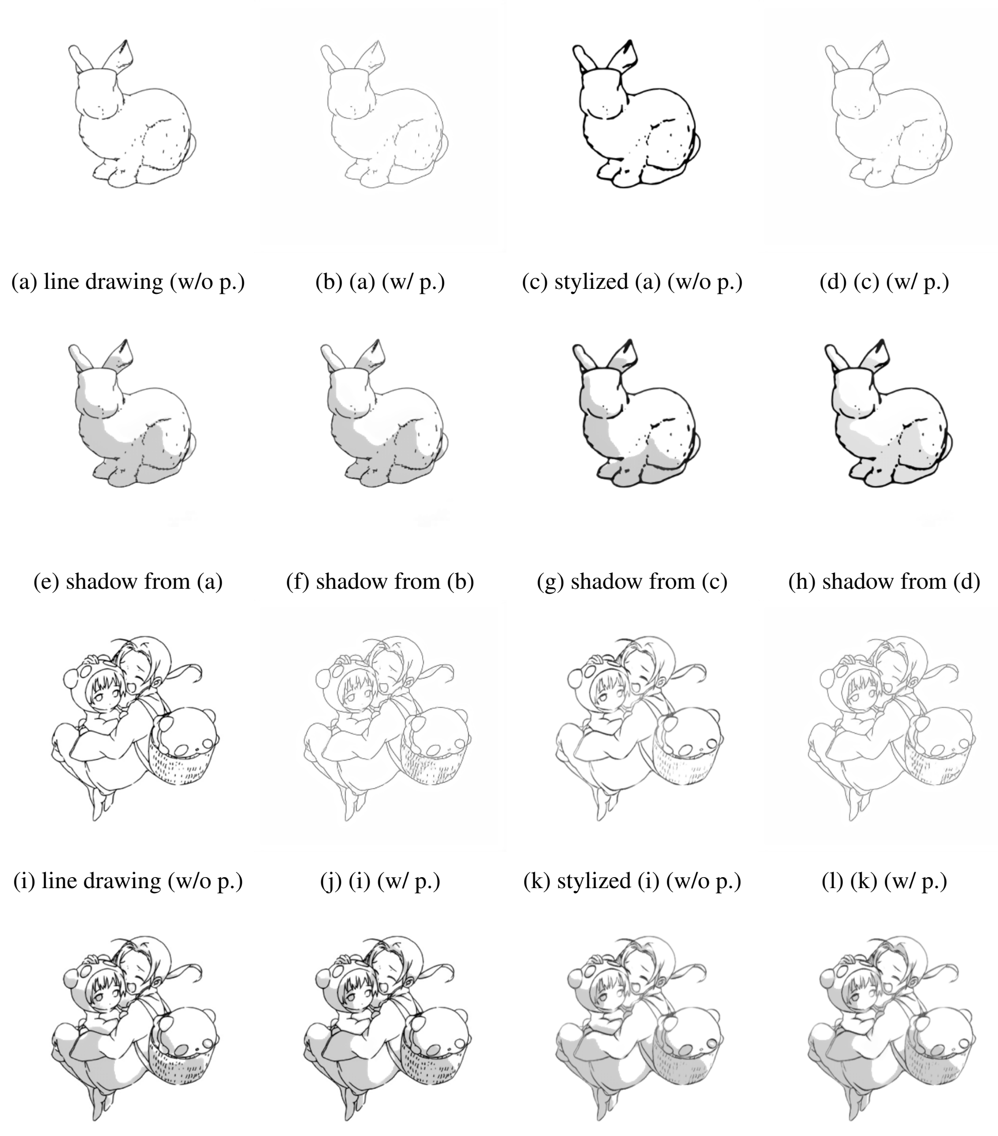
Figure 24: Examples of our shadowing system applying to Ukiyo-e (Kabuki Actor Segawa Kikunoj III as the Shirabyshi Hisakata Disguised as Yamato Manzai, by Toshusai Sharaku. Public domain.). Composite our shadows with pure colorized artwork.

(a) Original (b) w/o shadows (c) Lines (d) 510 (e) 520 (f) 530

(g) 210 (h) 220 (i) 230 (j) 810 (k) 820 (l) 830

(m) 110 (n) 120 (o) 130 (p) 710 (q) 720 (r) 730

Figure 25: Examples of our shadowing system applying to poster (Jardin de Paris, Fłte de Nuit Bal, illustrated by Jules Cheret. Public domain.). (a): original poster. (b): remove the shadows in the human. (c): extract line drawing from (a). (d)-(r): composites our shadows in various lighting directions with (b). Assuming the artists draw artwork with digital tools, they can rapidly try different shadows with our shadowing system.



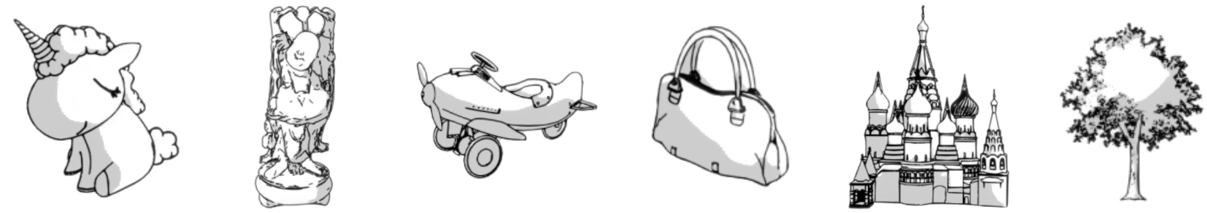
(a) line drawing (w/o p.) (b) (a) (w/ p.) (c) stylized (a) (w/o p.) (d) (c) (w/ p.)

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

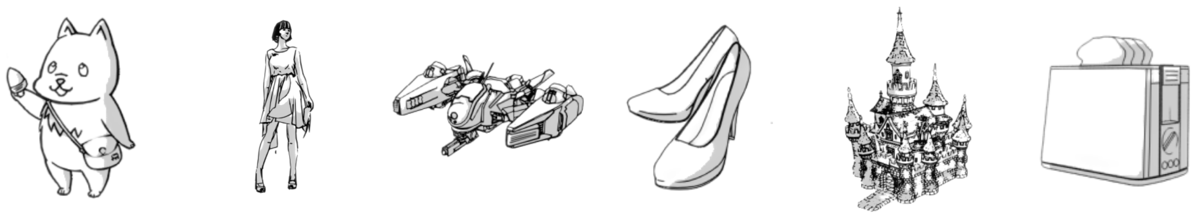
(i) line drawing (w/o p.) (j) (i) (w/ p.) (k) stylized (i) (w/o p.) (l) (k) (w/ p.)

(m) shadow from (i) (n) shadow from (j) (o) shadow from (k) (p) shadow from (l)

Figure 26: The comparisons of our shadowing system with and without pre-processing (denoted as w/ p. and w/o p.). (a), (c), (i), (k) are line drawings without our pre-processing. (b), (d), (j), (l) are line drawings after our pre-processing. We test the robustness of our pre-processing system with stylized lines (c) and (k) which have different line width, line transparency, and line strokes.

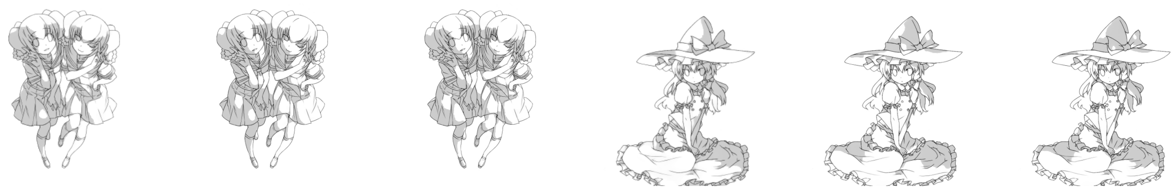


310 810 220 210 220 820

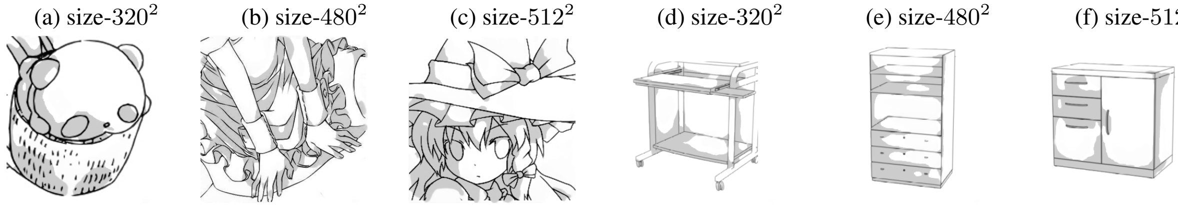


710 120 820 810 320 810

Figure 27: Evaluations on various categories of sketch (e.g. sculpture, bags, shoes, toys, sketchy cloth, buildings and etc.). This demonstrates that our work has generalization ability.



(a) size-320^2 (b) size-480^2 (c) size-512^2 (d) size-320^2 (e) size-480^2 (f) size-512^2



(g) 210 (h) 710 (i) 310 (j) 210 (k) 210 (l) 210

Figure 28: Limitation examples of the Future Work section. (a)-(c) and (d)-(f): invariant performance of shadows in different input size under the same lighting direction. (g)-(i): results of the local parts of line drawings being inputted. (j)-(l): unrealistic shadows in complex hard surface object.

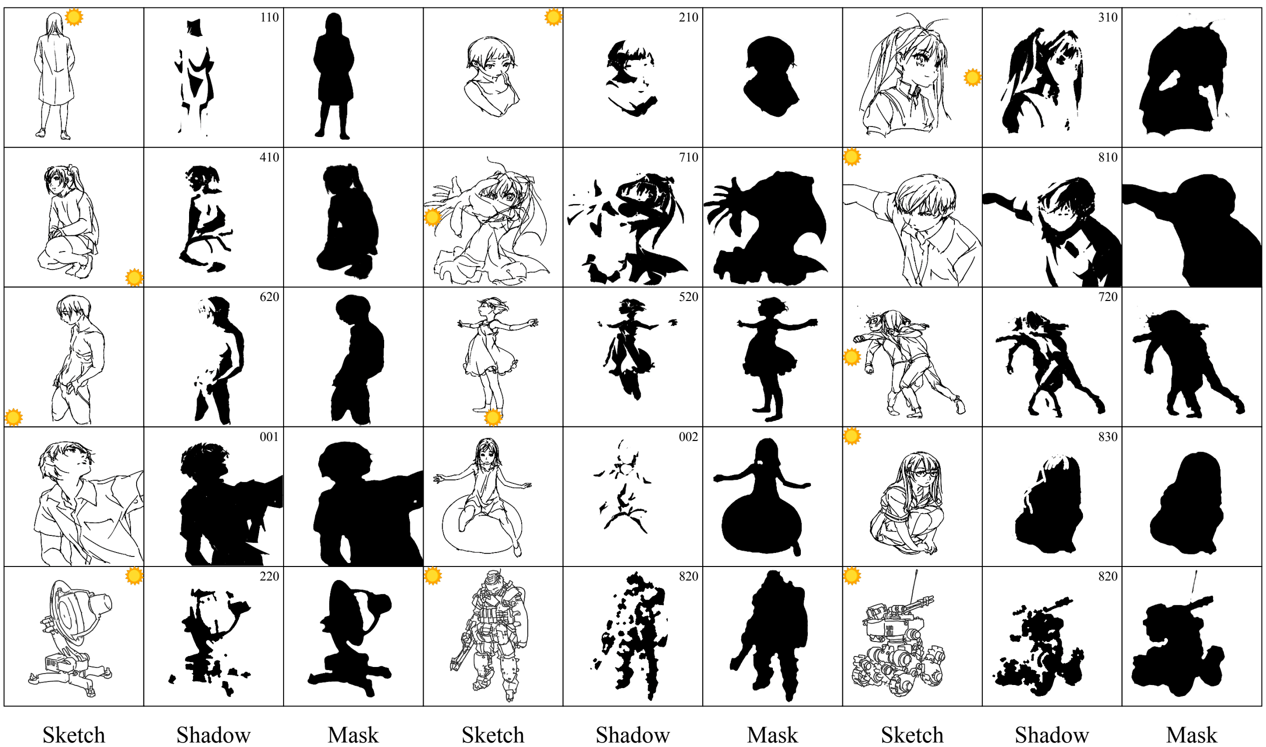


Figure 29: Sketch/shadow/mask pairs from our dataset. Our dataset contains alpha masks for the line drawings, but we did not need to use these masks in this paper.